

كتاب البرمجة بلغة بايثون

الجزء الثالث

تأليف

مصطفى صادق لطيف

2018



لتنزيل الجزء الاول يرجى زيارة الرابط التالي ([الجزء الأول](#)).

لتنزيل الجزء الثاني يرجى زيارة الرابط التالي ([الجزء الثاني](#)).

المقدمة:

السلام عليكم ورحمة الله وبركاته...

اخوتي الكرام اخواتي الفاضلات...

بدأنا معكم منذ مدة كورس اساسيات البرمجة بلغة بايثون وشرحنا عنه 19 درس تم نشرها في الجزئين الأول والثاني من هذا الكتاب.

وبعد أن أكملنا الحديث عن الاساسيات، انتقلنا إلى الحديث عن نوع خاص من مترجمات بايثون وهي مترجمات بايثون التفاعلية (Interactive Python Compilers) والتي نستطيع الوصول إليها عن طريق برنامج ذو واجهة ويب ونستطيع برمجة الكثير من التطبيقات التفاعلية بواسطتها. ولأن لها الكثير من التطبيقات فقد ارتأينا تخصيص الكلام عن تطبيقات بايثون في علم البيانات (تحليل البيانات، تصنيف وتبويب البيانات، معالجة البيانات الضخمة والخروج باستنتاجات حولها ... الخ). وبعد ان شرحنا عدة مكتبات خاصة بهذه المواضيع انتقلنا الى شرح مشروع متكامل لبناء نماذج التخمين (predictive models) لتطبيق إقراض المواطنين وشرحنا كل ذلك في 12 درس وها نحن نقوم بجمعها لكم في ملف واحد أسميناه (الجزء الثالث) من كتاب البرمجة بلغة بايثون. اتمنى ان يكون الكتاب بكل أجزائه مفيداً للجميع وانتظرونا في اجزاء قادمة ان شاء الله وحول مجالات أخرى للغة بايثون التي تنمو بشكل سريع جداً وسيكون لها مستقبل رائع في شتى المجالات.

تحياتي للجميع و نستقبل ارائكم واستفساراتكم حول الكتاب ومحتواه والأجزاء السابقة والقادمة منه (ان شاء الله) على موقعنا على الرابط التالي:

مدونة مصطفى صادق العلمية (www.mustafasadiq0.com)

او صفحتنا على الفيس بوك: صفحة مصطفى صادق العلمية:

(www.facebook.com/mustafasadiq01).

اخوكم

مصطفى صادق لطيف

بايثون-20: البداية مع علم البيانات وتنصيب البرمجيات الضرورية

السلام عليكم

اليوم ان شاء الله نبدأ برحلة طويلة لشرح كيفية التعامل مع البيانات بأحترافية والطريق الى احتراف علم البيانات Data Science باستخدام لغة بايثون.

بداية وقبل البدء يفترض أن يكون الشخص القاريء لهذا الدرس قد أكمل المتطلبات التالية:

1- الاطلاع على الدروس السابقة (الجزء الأول والثاني من الكتاب) وفهمها بشكل جيد

2- ان يكون بالفعل قد وجد اجابة السؤال التالي (لماذا ادرس لغة بايثون. وما هي فائدتها؟).

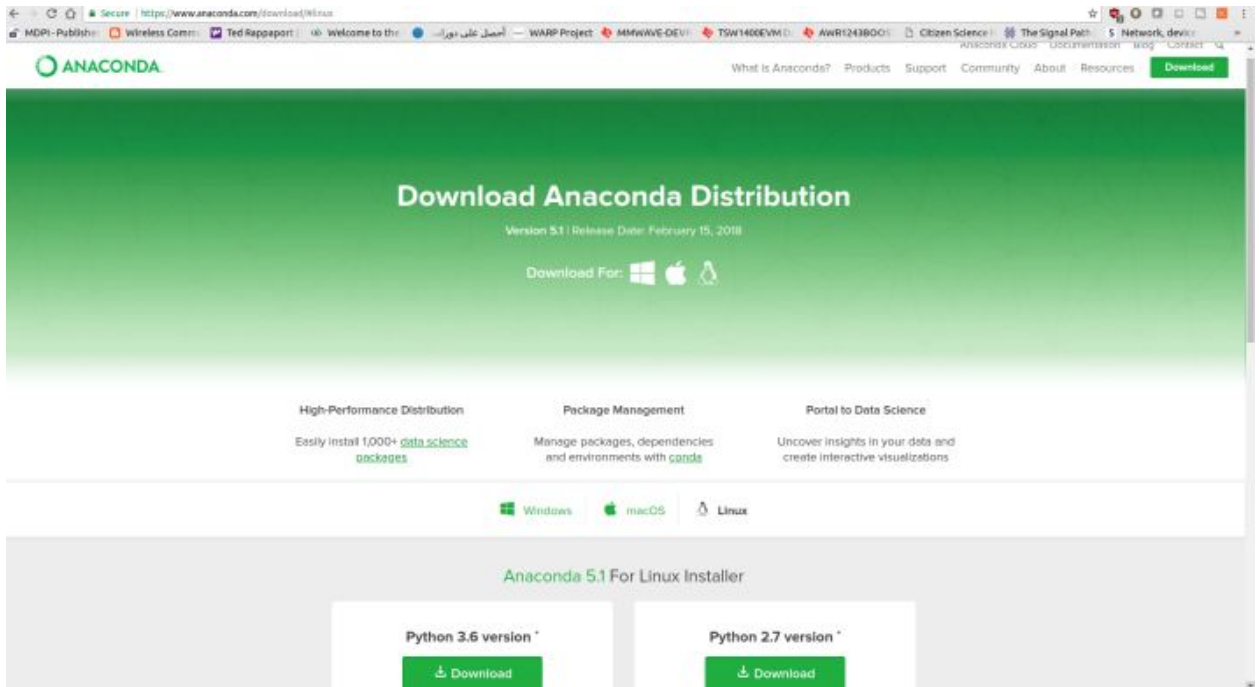
3- يجب ان يكون المهتم والقارئ لهذه السطور عارفاً ان لغة بايثون هي لغة عامة الأغراض يمكن ان تستخدم في أي مجال من مجالات البرمجة الكثيرة ولكنها تتميز اليوم في عدة مجالات كأسرع لغة برمجة من ناحية المكتبات الملحقة بها ودعمها عالمياً وعدد المستخدمين لها ومن هذه المجالات علم البيانات الذي نحن بصدد الحديث عنه حيث تتشارك لغة بايثون مع لغة (R) الريادة لهذا المجال كأفضل لغات البرمجة للتعامل مع البيانات من ناحية العرض والترشيح (الفلتر) والرسم وتحويل البيانات المبعثرة الى نتائج ذات قيمة.

الان وبعد ان عرفنا كل هذه الامور دعونا نبدأ على بركة الله.

الخطوة الاولى كالعادة في تعلم اي لغة برمجة هو تنصيب المنصة البرمجية الخاصة بها في حواسيبنا الشخصية وبخلاف الدروس السابقة والتي تعاملنا فيها مع لغة بايثون باستخدام محرر اللغة الأساسي python IDE

سنقوم في هذه الدروس بالعمل على منصة متكاملة للغة بايثون وهي منصة اناكوندا (وهو اسم لنوع اخر من الافاعي حيث يبدو ان القائمين على هذه اللغة وعشاقها من المحبين للأفاعي بمختلف أنواعها :)) التي يمكن تنزيلها من الرابط التالي: [Download Anaconda](https://www.anaconda.com/download/linux)

والآن بمجرد النقر على الرابط اعلاه ستفتح لك نافذة تحتوي على روابط التنزيل المناسبة لنظامك سواء أكان ويندوز او لينكس او ماك (حيث يتم التعرف على جهازك بشكل تلقائي) وبالنسبة لي ولأنني أعمل على نظام لينكس فقد ظهرت لي النافذة التالية:



وبعد النقر على زر التنزيل لنسخة البرنامج المناسبة لنظام تشغيلك سيتم تنزيل الحزمة على شكل ملف مضغوط في ملف التنزيل وبعدها يمكن اتباع الخطوات التالية لكل نظام تشغيل:

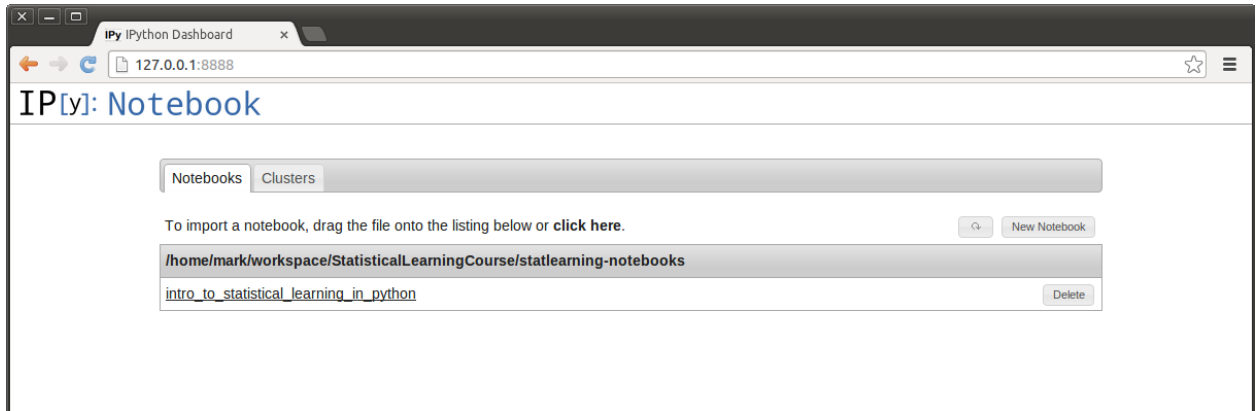
لنظام الماك (Mac OS) يمكن اتباع الخطوات التالية:

1- بعد تنزيل الحزمة نقوم بفتح الضغط للملف المضغوط ثم نفتح المجلد ونبحث عن الملف الذي امتداده (.pkg) وننقر عليه ثم نتبع تعليمات التنصيب في الشاشة وحين يطلب منا الاختيار لنوع التنصيب يجب ان نختار: Install for me only

2- سنتظهر نافذة التطبيق في سطح المكتب وننقر عليها نقرتين لتفتح الواجهة ومنها نختار lunch ليبدأ البرنامج.

3- إذا لم يبدأ البرنامج بشكل تلقائي بأخذك الى المتصفح فقم بفتح متصفح الويب والدخول الى الرابط التالي: <http://127.0.0.1:8888> او <http://127.0.0.1:8889> او <http://127.0.0.1:8890>

وبعدها ستظهر لك واجهة مشابهة للتالي:



4- بعدها تنقر على New Notebook الموجود في يمين الصفحة

5- نقوم بإعادة تسمية الملف untitled0 بأي اسم تريده

6- والان يفترض ان ترى هذين القوسين [] فننقر الى يمين هذين القوسين ونبدأ بكتابة البرامج التي سنتعلمها في الدروس القادمة ان شاء الله.

لنظام لينكس اوبونتو (Linux Ubuntu) نقوم باتباع الخطوات التالية:

1- بعد تنزيل الملف (والذي حجمه تقريباً 500 ميغابايت) يكون بامتداد (.sh) وهو ملف غير قابل للتنفيذ مبدئياً.

2- لتحويل الملف الى ملف تنفيذي نقوم بالدخول الى سطر الاوامر (terminal) ونذهب الى مجلد التنزيل (downloads) وهناك نكتب الايعاز التالي:

```
chmod +x Anaconda-<version>.sh
```

ومعنى هذا الايعاز هو الاتي: قم بأضافة (+) خاصية التنفيذ (x) الى الملف الذي اسمه (Anaconda....) وعبارة (<version>) هي النسخة الخاصة بالملف المنزل و(.sh) هو الامتداد.

3- بعد تنفيذ الايعاز اعلاه نقوم بتنفيذ هذا الايعاز

```
./Anaconda-<version>.sh
```

4- نقوم باتباع التعليمات على الشاشة حتى تنتهي عملية التنصيب وكما في النافذة التالية:

```
mustafa@mustafapc: ~/Downloads
mustafa@mustafapc:~$ cd Downloads/
mustafa@mustafapc:~/Downloads$ chmod +x Anaconda3-5.1.0-Linux-x86_64.sh
mustafa@mustafapc:~/Downloads$ ./Anaconda3-5.1.0-Linux-x86_64.sh

Welcome to Anaconda3 5.1.0

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
=====
Anaconda End User License Agreement
=====

Copyright 2015, Anaconda, Inc.

All rights reserved under the 3-clause BSD License:

Redistribution and use in source and binary forms, with or without modification,
are permitted provided that the following conditions are met:

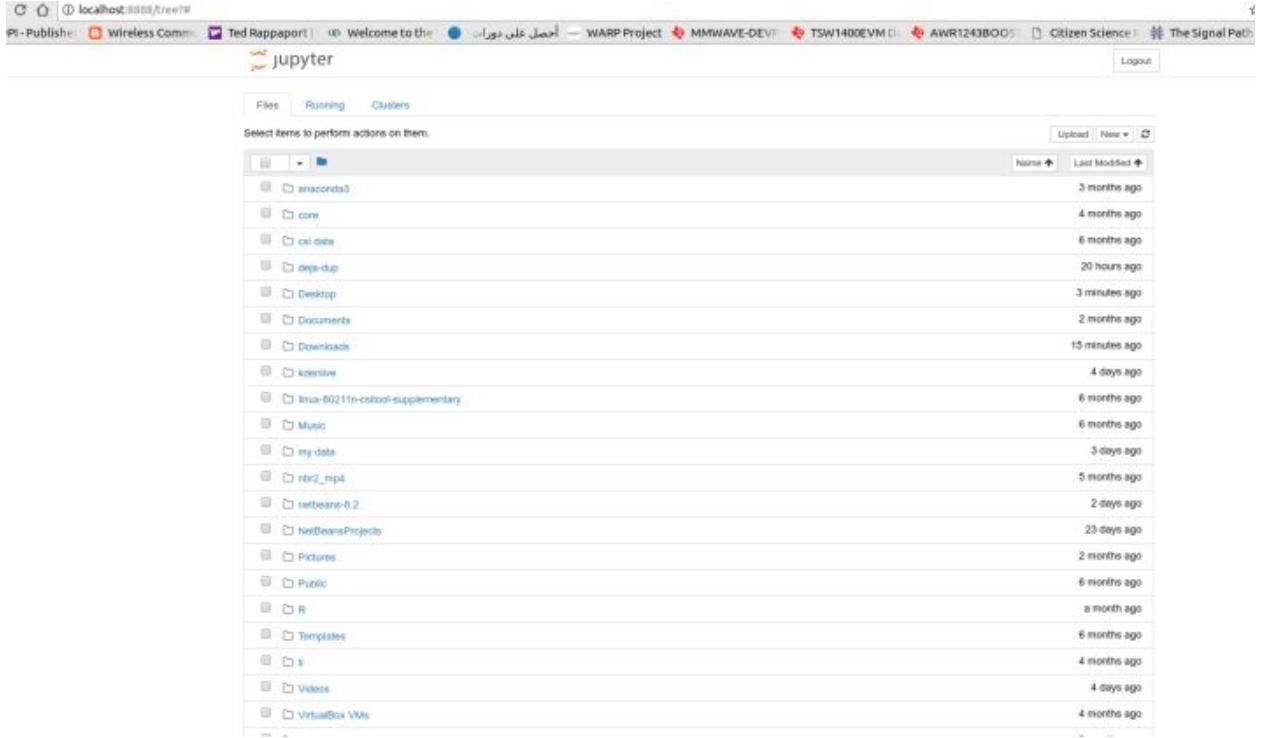
  * Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.
  * Redistributions in binary form must reproduce the above copyright notice, th
  is list of conditions and the following disclaimer in the documentation and/or o
  ther materials provided with the distribution.
  * Neither the name of Anaconda, Inc. ("Anaconda, Inc.") nor the names of its c
  ontributors may be used to endorse or promote products derived from this softwar
  e without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WA
RRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.
```

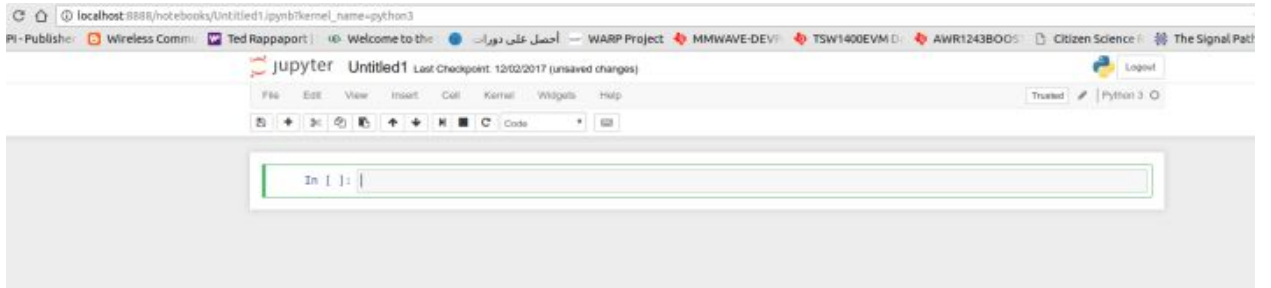
5- لتشغيل ال (IPython Notebook server) نقوم بالدخول مرة اخرى الى سطر الاوامر (terminal) ونكتب اليعاز التالي:

ipython notebook

مباشرة ستفتح لنا نافذة كالآتي:



نقوم بالنقر على زر (new) في الزاوية العليا اليمنى ونختار منه (python) لفتح نافذة اخرى في المتصفح كما في ادناه:



هذه النافذة هي الواجهة الرئيسية التي سنتعامل معها في كتابة تطبيقات بايثون لعلم البيانات والتي سنتعرف على تفاصيلها في الدروس القادمة ان شاء الله.

أما بخصوص نظام الويندوز Windows فيمكن تنصيب حزمة اناكوندا باتباع الخطوات التالية:

1- بعد تنزيل النسخة المناسبة لنا نذهب الى مجلد التنزيل download

2- نعر على ملف اناكوندا بامتداد (.exe) فننقر عليه ونقوم بتنصيبه تنصيباً طبيعياً كأى برنامج ويندوز اخر.

3- بعد اكتمال التنصيب تظهر ايقونة البرنامج في سطح المكتب فنقوم بالنقر عليها نقرتين فتفتح لنا المتصفح كما في النافذة اعلاه وبذلك يكون التنصيب قد اكتمل واصبحنا مستعدين للبدء باستخدام هذه الحزمة الرائعة التي ستحتوي على اغلب ما سنحتاجه للعمل على بايثون لتطبيقات علم البيانات.

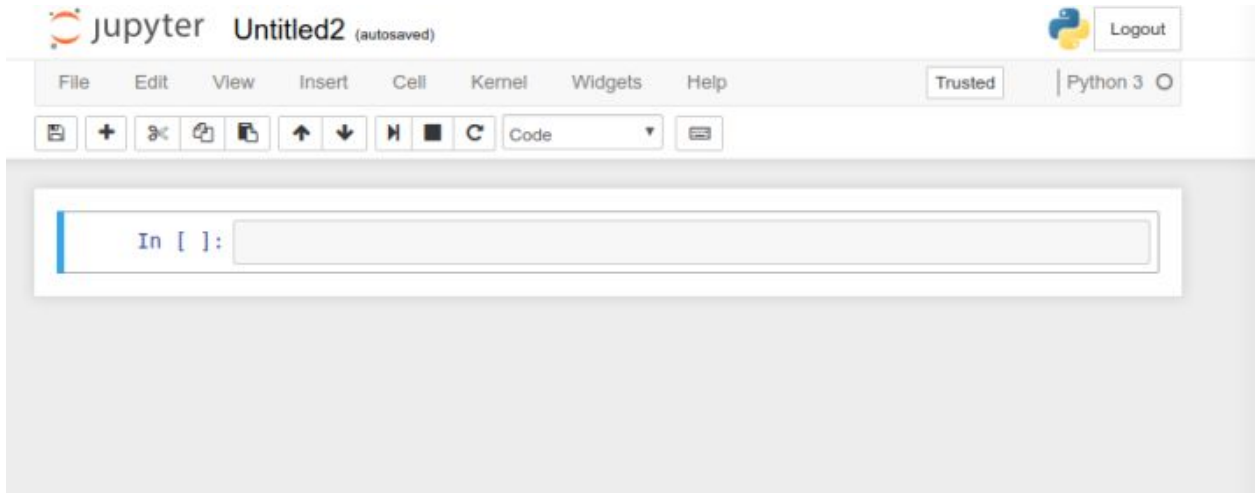
الى هنا تنتهي عملية تنصيب حزمة اناكوندا والتي سيتم شرحها بالتفصيل في الدروس القادمة ان شاء الله فانتظرونا

بايثون-21: التعرف على واجهة Jupyter وكيفية استخدامها للبرمجة

السلام عليكم

في هذا الدرس سنتعرف على واجهة الويب الخاصة بحزمة (Jupyter) وهو المشروع المفتوح المصدر الذي يستهدف بناء التطبيقات مفتوحة المصدر التفاعلية للعديد من لغات البرمجة والمكتوب بلغة بايثون ويمكن معرفة المزيد عنه من الرابط التالي (<http://jupyter.org/index.html>). بعد ان قمنا بتنصيب هذه الحزمة في الدرس السابق و تعرفنا على كيفية فتحها في انظمة التشغيل المختلفة (ويندوز، ماك، ولينكس اوبونتو).

الواجهة الاولى التي ستظهر لنا حين فتح التطبيق هي المبينة في الصورة أدناه:



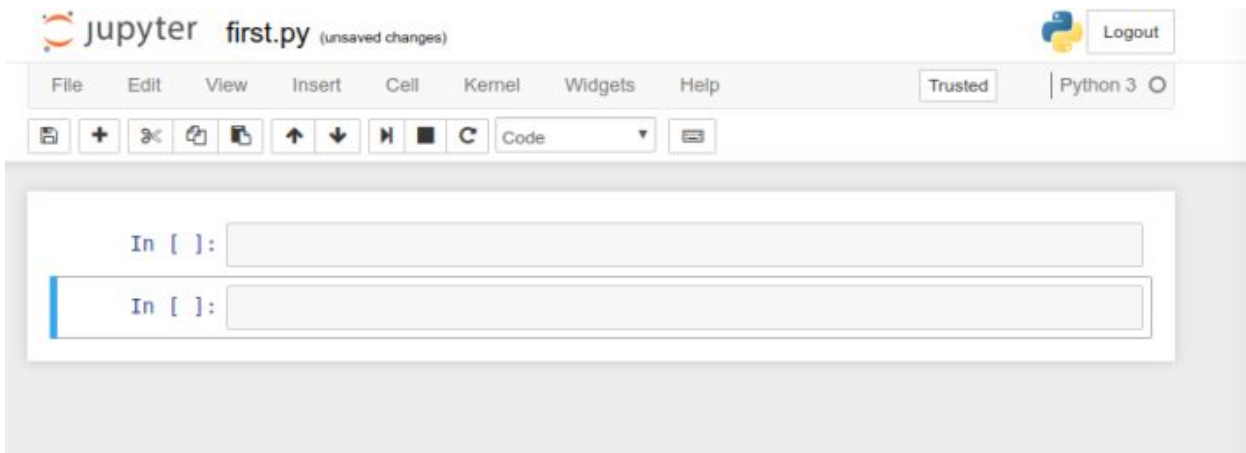
وهنا تظهر لنا الأدوات التي نستطيع من خلالها التحكم بالبرنامج المكتوب وتنفيذه والتلاعب به وكما يلي: في اعلى النافذة والى اليمين من ايقونة البرنامج (Jupyter) نرى اسم الملف الذي نعمل عليه الان وهو في هذه الحالة (Untitled2) والذي يمكن تغييره بالنقر عليه لتظهر النافذة المبينة ادناه:



حيث قمنا بتسمية الملف (first.py).

والان ننتقل الى قائمة الادوات ونبدأ بها من اليسار الى اليمين:

- رمز ال(floppy disk): وهو رمز الحفظ المعروف.
- رمز (+): ويستخدم لأضافة خلية جديدة اسفل الخلية الحالية وعند النقر عليه نلاحظ ظهور



التالي:

- رمز المقص: ويستخدم لقص (او حذف) خلية معينة بعد النقر عليها.
- رمز الورقتين: لنسخ الخلية التي نعمل عليها الان.
- رمز اللصق المعروف: للصق خلية معينة في مكان ما اسفل الخلية التي نعمل عليها.
- رمز السهم الى الاعلى: لتحريك الخلية المحددة الى الاعلى.

- رمز السهم الى الاسفل: تحريك الخلية المحددة الى الاسفل.
- رمز السهم الجانبي: لتنفيذ الكود المكتوب في الخلية المحددة حالياً (الخلية التي يظهر خط ملون الى جانبها الايسر).
- رمز المربع الاسود: لأيقاف التنفيذ.
- رمز السهم المعقوف: لإعادة تشغيل الواجهة في بيئة الويب.

هذه تقريباً هي اغلب الادوات التي سنتعامل معها في كتابة الكود وفي حالة الحاجة الى ادوات اخرى فسيتم شرحها قبل استخدامها ان شاء الله.

ملاحظة: في هذه المرحلة وكما نوهنا سابقاً المفروض لمن يرغبون بمتابعة هذه الدروس ان يكونوا قد قاموا بمراجعة الدروس السابقة والتعرف على اساسيات لغة بايثون ليكونوا مستعدين للعمل على البرامج المتقدمة بلغة بايثون التي سنشرحها في الدروس القادمة ان شاء الله.

كذلك اضافة الى المراجعة للدروس السابقة، هذه مجموعة من الامور التي يجب اتقانها قبل اكمال بقية اجزاء الكورس:

1- التعابير القياسية (والتي تم شرحها في الدروس 18 و 19) وتلخصها الصورة التالية:

Python Regex Cheatsheet

Quick Filter

Regular Expression Basics	Regular Expression Character Classes	Regular Expression Flags
.	[ab-d] One character of: a, b, c, d	l Ignore case
a	[^ab-d] One character except: a, b, c, d	m ^ and \$ match start and end of line
ab	[b] Backspace character	s . matches newline as well
a b	\d One digit	x Allow spaces and comments
a*	\D One non-digit	L Locale character classes
\	\s One whitespace	u Unicode character classes
	\S One non-whitespace	(?iLmsux) Set flags within regex
	\w One word character	
	\W One non-word character	
Regular Expression Quantifiers	Regular Expression Assertions	Regular Expression Special Characters
*	^ Start of string	\n Newline
+	\A Start of string, ignores m flag	\r Carriage return
?	\$ End of string	\t Tab
{2}	\Z End of string, ignores m flag	\YYY Octal character YYY
{2, 5}	\b Word boundary	\xYY Hexadecimal character YY
{2,}	\B Non-word boundary	
{,5}	(?=...) Positive lookahead	Regular Expression Replacement
	(?!...) Negative lookahead	\g<0> Insert entire match
	(?<=...) Positive lookbehind	\g<Y> Insert match Y (name or number)
	(?<!...) Negative lookbehind	\Y Insert group numbered Y
	(?()) Conditional	
Regular Expression Groups		
(...)		
(?P<Y>...)		
(?:...)		
\Y		
(?P=Y)		
(?#...)		

New to Debuggex? Check out the [regex tester!](#)

وهذه بعض التمارين لأختبار مستواك في فهم التعبيرات القياسية قبل الاستمرار في بقية مكونات الكورس: <https://developers.google.com/edu/python/exercises/baby-names>

<https://www.analyticsvidhya.com/blog/2014/11/text-data-cleaning-steps-python>

2- تنصيب حزمة اناكوندا وجوبيتر وكيفية التعامل مع واجهاتها (الدرس 20 و 21).

واخيراً وقبل الانتهاء من هذا الدرس التحضيري اليكم مثال بسيط عن كيفية كتابة وتنفيذ الكودات في واجهة جوبيتر:

```
In [8]: 4+5
Out[8]: 9

In [12]: a=5
         b=4
         c=(a*b)**2 # multiply a and b and raise the result to the power of 2

In [13]: print(c)
         400

In [14]: first=int(input('Enter the first number' ))
         second=int(input('Enter the second number'))

         #addition
         print(first+second)

Enter the first number33
Enter the second number55
88

In [ ]:
```

الى هنا ينتهي درسنا لهذا اليوم وفي الدرس القادم ان شاء الله سنبدأ الحديث عن مكتبات بايثون الخاصة بمعالجة وتحليل البيانات (علم البيانات عموماً) فأحرصوا على اكمال المتطلبات المذكورة أعلاه قبل الدرس القادم وبالتوفيق للجميع ان شاء الله

بايثون-22: التعامل مع مكتبة NumPy الجزء الاول

السلام عليكم

بعد ان تناولنا كيفية تنصيب حزمة اناكوندا وتعرفنا على واجهة جوبيتر في الدرسين الماضيين،
نبدأ اليوم على بركة الله التعرف على مكتبة بايثون العلمية المسماة NumPy

هذه المكتبة تختص بشكل رئيسي بالتعامل مع الكميات الكبيرة من البيانات (المصفوفات متعددة
الابعادة بشكل عام) وتأتي منصبه تلقائياً مع حزمة اناكوندا ولكن ان كنت قد اخترت متابعة
الكورس والعمل على منصة اخرى فيمكن تنصيب هذه المكتبة من الرابط التالي:

[/http://www.numpy.org](http://www.numpy.org)

بعد تنصيب هذه المكتبة يمكن استخدامها في اي كود بعد استيرادها كما في ادناه

```
from numpy import *
```

والان لنبدأ التعامل مع ادوات هذه المكتبة بالتفصيل:

العنصر الرئيسي الذي تتعامل معه ادوات هذه المكتبة هو المصفوفات المتجانسة التي تتكون
عموماً من جداول تحتوي ارقام كلها لها نفس النوع (اعداد صحيحة او عشرية) ويتم فهرستها
بأستخدام قيم موجية صحيحة وكل بعد من ابعاد هذه المصفوفات يسمى محور (axis) وعدد
المحاور يسمى الرتبة (rank) للمصفوفة.

وكمثال على هذه المصفوفات هو احداثيات نقطة في فضاء ثلاثي الابعاد حيث يمكن التعبير عنها
بالشكل التالي:

```
In [1]: coordinates=[1,2,3]
```

```
In [2]: print(coordinates)
[1, 2, 3]
```

والآن يجب التفريق بين المصفوفة المعرفة كمصفوفة (NumPy) والتي يكون نوعها (ndarray) وبين مصفوفات لغة بايثون الطبيعية التي تكون احادية البعد فقط والتي يكون نوعها (array) ومن ابرز الادوات لمكتبة مصفوفات ال (NumPy) هي:

1- (ndarray.ndim) وتستخدم لإعطاء عدد المحاور او الابعاد الخاص بمصفوفة ما وهو ايضاً يمثل رتبة (rank) المصفوفة وكما في المثال أدناه:

```
In [1]: from numpy import * # استيراد مكتبة NumPy
a = arange(15).reshape(3, 5) # تعريف مصفوفة تحتوي العناصر من صفر الى 14
```

```
In [2]: a
```

```
Out[2]: array([[ 0,  1,  2,  3,  4],
               [ 5,  6,  7,  8,  9],
               [10, 11, 12, 13, 14]])
```

```
In [4]: a.ndim
```

```
Out[4]: 2
```

وهنا كما نرى قمنا بالبداية باستيراد المكتبة (numpy) في السطر الاول ثم قمنا بتعريف مصفوفة اسمها (a) وتتكون من الارقام من صفر الى 14 باستخدام دالة (arange) ثم قمنا بتحويلها الى مصفوفة تتكون من بعدين بحيث يحتوي كل سطر (البعد الأول) على 5 عناصر والبعد الثاني (الاعمدة) على 3 عناصر باستخدام دالة (reshape).

بعد ذلك قمنا بكتابة (a) فقط ثم نقرنا على زر التنفيذ فتمت طباعة قيمة المتغير (a) وهذا يشبه دالة

```
print(a)
```


واخيراً قمنا بطلب معرفة عدد أبعاد المصفوفة (a) باستخدام الاداة (ndim) فكانت النتيجة 2 اي أنها تتكون من بعدين (صفوف وأعمدة).

-2 (ndarray.shape) وتستخدم لمعرفة عدد الصفوف والاعمدة وبقية الابعاد الاخرى للمصفوفة وفي حالتنا هذه سيكون الجواب 3 للصفوف و 5 للأعمدة وكما في ادناه:

```
In [3]: a.shape
```

```
Out[3]: (3, 5)
```

-3 (ndarray.size) وتستخدم لطباعة العدد الكلي لعناصر المصفوفة وهو عادة حاصل ضرب القيم الناتجة من الدالة السابقة وكما في المثال التالي:

```
In [3]: a.shape
```

```
Out[3]: (3, 5)
```

```
In [7]: a.size
```

```
Out[7]: 15
```

-4 (ndarray.dtype) وتستخدم لطباعة نوع المتغيرات او القيم في داخل المصفوفة وكما في المثال ادناه:

```
In [15]: a.dtype
```

```
Out[15]: dtype('int64')
```

```
In [16]: a.dtype.name
```

```
Out[16]: 'int64'
```

5- (ndarray.itemsize) ويستخدم لطباعة عدد البايتات التي يشغلها كل عنصر من عناصر المصفوفة وكمثال لو كان نوع المتغير (float64) فهذا يعني انه من نوع المتغيرات التي تشغل 64 بت والتي تساوي 8 بايت لأننا نعرف ان كل بايت يساوي 8 بت وهكذا وكما بينه المثال ادناه:

```
In [16]: a.dtype.name
```

```
Out[16]: 'int64'
```

```
In [6]: a.itemsize
```

```
Out[6]: 8
```

```
In [17]: b=zeros([2,3,4])  
b
```

```
Out[17]: array([[ [ 0.,  0.,  0.,  0.],  
                 [ 0.,  0.,  0.,  0.],  
                 [ 0.,  0.,  0.,  0.]],  
               [[ 0.,  0.,  0.,  0.],  
                 [ 0.,  0.,  0.,  0.],  
                 [ 0.,  0.,  0.,  0.]])
```

```
In [18]: b.dtype
```

```
Out[18]: dtype('float64')
```

```
In [19]: b.itemsize
```

```
Out[19]: 8
```

6- (ndarray.data) ويستخدم هذا اليعاز لعرض موقع الذاكرة الخاص بحفظ قيم المصفوفة ولا يتم استخدامه عادة لأنه فقط مفيد لأدارة الذاكرة وليس للاستخدام الطبيعي وكما يوضحه المثال التالي:

```
In [17]: b=zeros([2,3,4])
b
Out[17]: array([[[ 0.,  0.,  0.,  0.],
                 [ 0.,  0.,  0.,  0.],
                 [ 0.,  0.,  0.,  0.]],
                [[ 0.,  0.,  0.,  0.],
                 [ 0.,  0.,  0.,  0.],
                 [ 0.,  0.,  0.,  0.]])

In [18]: b.dtype
Out[18]: dtype('float64')

In [19]: b.itemsize
Out[19]: 8

In [20]: b.data
Out[20]: <memory at 0x7f16941f1408>
```

الى هنا ينتهي الجزء الأول من شرح كيفية التعامل مع مكتبة بايثون الرقمية NumPy وسنكمل معكم في الأجزاء القادمة ان شاء الله فتابعوا معنا.

بايثون-23: مكتبة NumPy الجزء الثاني

السلام عليكم

سنتناول في هذا الدرس المواضيع التالية:

- إنشاء المصفوفات في مكتبة NumPy
- طباعة المصفوفات
- العمليات على المصفوفات

إنشاء المصفوفات

هناك عدة طرق لإنشاء المصفوفات بلغة بايثون ومنها ما تطرقنا له في الدرس السابق من تعريف مكتبة NumPy ثم تعريف متغير على انه مصفوفة وذكر العناصر مباشرة وكما في المثال التالي:

```
In [1]: from numpy import *  
a=array([1,2,3,4,5])  
a
```

```
Out[1]: array([1, 2, 3, 4, 5])
```

```
In [2]: a.dtype
```

```
Out[2]: dtype('int64')
```

```
In [3]: b=array ([1.2,3.4,5.6,6.8])  
b
```

```
Out[3]: array([ 1.2,  3.4,  5.6,  6.8])
```

```
In [4]: b.dtype
```

```
Out[4]: dtype('float64')
```

كما هو واضح في الصورة أعلاه فقد قمنا بتعريف مصفوفتين احدهما نوعها صحيحة والاخرى عشرية وكلاهما تتكون من بعد واحد (صف واحد او عمود واحد من البيانات) ولتعريف مصفوفات متعددة الابعاد فإن السلسلة الواحدة تعتبر صف واحد وسلسلة السلاسل تعتبر صفوف واعمدة وكذلك سلسلة السلاسل المتكونة من سلاسل تعتبر مصفوفة ثلاثية الابعاد وهكذا:

```
In [9]: c=array([[1,2,3],[4,5,6],[7,8,9],zeros(3)])  
c
```

```
Out[9]: array([[ 1.,  2.,  3.],  
              [ 4.,  5.,  6.],  
              [ 7.,  8.,  9.],  
              [ 0.,  0.,  0.]])
```

```
In [11]: d=array([[1,2],[3,4],[5,6]], dtype=complex)  
d
```

```
Out[11]: array([[ 1.+0.j,  2.+0.j],  
              [ 3.+0.j,  4.+0.j],  
              [ 5.+0.j,  6.+0.j]])
```

الصورة اعلاه تبين مصفوفتين الاول ابعادها 3*4

والسطر الاخير منها يتكون من اصفار فقط قمنا بتعريفها باستخدام دالة

`zeros()`

والمصفوفة الثانية تتكون من ثلاث صفوف وعمودين ونوع عناصرها مركبة حيث قمنا بتعريف نوع العناصر باستخدام دالة

`dtype=complex`

بعد الانتهاء من تعداد عناصر المصفوفة.

كما هو معروف في لغات البرمجة فإن المصفوفات قد تكون عناصرها غير معروفة ولكن حجمها يجب ان يتم تعريفه من البداية تجنباً للأرباك الذي قد يحصل فيما بعد حيث ان الذاكرة المخصصة للمصفوفة يجب ان تحجز مسبقاً عند تعريفها (ويشذ عن هذه القاعدة المتجهات vectors في لغة سي بلس بلس حيث يمكن تعريف متجه (بعد واحد) او عدة متجهات (مصفوفة متعددة الابعاد) بدون تحديد الحجم)

وتوفر لغة بايثون عدة طرق لتعريف مصفوفة بدون ذكر عناصرها الحقيقية في البداية باستخدام دوال

`zeros(n,m) :`

لتعريف مصفوفة تتكون من اصفار عدد صفوفها (n) وعدد اعمدتها (m)

`ones(n,m)`

نفس الدالة السابقة ولكنها تستخدم لملء المصفوفة بوحدات كقيم اولية يمكن تغييرها فيما بعد.

`empty(n,m)`

وتعمل نفس الدوال السابقة ولكنها تملأ المصفوفة بقيم عشوائية نوعها (float64) وكما في الامثلة التالية:

```
In [13]: e=zeros( (3,4) )
e
```

```
Out[13]: array([[ 0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.],
                [ 0.,  0.,  0.,  0.]])
```

```
In [14]: f=ones( (2,3,4),dtype=int16)      # مصفوفة ثلاثية الابعاد
f
```

```
Out[14]: array([[[1, 1, 1, 1],
                 [1, 1, 1, 1],
                 [1, 1, 1, 1]],

                [[1, 1, 1, 1],
                 [1, 1, 1, 1],
                 [1, 1, 1, 1]]], dtype=int16)
```

```
In [15]: g=empty((2,2,2))
g
```

```
Out[15]: array([[[ 6.94981335e-310,  4.67495645e-310],
                 [ 0.00000000e+000,  0.00000000e+000]],

                [[ 0.00000000e+000,  0.00000000e+000],
                 [ 0.00000000e+000,  0.00000000e+000]])
```

لأنشاء مصفوفة تتكون من ارقام متسلسلة بين قيمتين نستطيع استخدام دالة (arange) التي سبق ان رأيناها في الدرس السابق والتي يشرحها المثال التالي:

```

In [18]: h=arange(2,4,0.1) # مصفوفة تبدأ بالرقم 2 وتنتهي بالرقم 4 وترداد 0.1
          h
Out[18]: array([ 2. ,  2.1,  2.2,  2.3,  2.4,  2.5,  2.6,  2.7,  2.8,  2.9,  3.
                ,  3.1,  3.2,  3.3,  3.4,  3.5,  3.6,  3.7,  3.8,  3.9])

In [19]: i=arange(15)
          i
Out[19]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

In [20]: j=arange(1,15)
          j
Out[20]: array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])

In [21]: h=arange(1,16,2)
          h
Out[21]: array([ 1,  3,  5,  7,  9, 11, 13, 15])

```

الدالة الاخرى المستخدمة لتوليد ارقام بين قيمتين والفارق بينهما قيمة ثابتة هي دالة

(linspace) وهي نفس الدالة المستخدمة في الماتلاب وبنفس المواصفات وكما في الامثلة التالية:


```

In [22]: k=linspace(0,2,10) # array from 0 to 2 with 10 elements
          k
Out[22]: array([ 0.          ,  0.22222222,  0.44444444,  0.66666667,  0.88888889,
                1.11111111,  1.33333333,  1.55555556,  1.77777778,  2.
                ])

In [23]: l=linspace(0,2*pi, 100) pi=3.14
          l
Out[23]: array([ 0.          ,  0.06346652,  0.12693304,  0.19039955,  0.25386607,
                0.31733259,  0.38079911,  0.44426563,  0.50773215,  0.57119866,
                0.63466518,  0.6981317 ,  0.76159822,  0.82506474,  0.88853126,
                0.95199777,  1.01546429,  1.07893081,  1.14239733,  1.20586385,
                1.26933037,  1.33279688,  1.3962634 ,  1.45972992,  1.52319644,
                1.58666296,  1.65012947,  1.71359599,  1.77706251,  1.84052903,
                1.90399555,  1.96746207,  2.03092858,  2.0943951 ,  2.15786162,
                2.22132814,  2.28479466,  2.34826118,  2.41172769,  2.47519421,
                2.53866073,  2.60212725,  2.66559377,  2.72906028,  2.7925268 ,
                2.85599332,  2.91945984,  2.98292636,  3.04639288,  3.10985939,
                3.17332591,  3.23679243,  3.30025895,  3.36372547,  3.42719199,
                3.4906585 ,  3.55412502,  3.61759154,  3.68105806,  3.74452458,
                3.8079911 ,  3.87145761,  3.93492413,  3.99839065,  4.06185717,
                4.12532369,  4.1887902 ,  4.25225672,  4.31572324,  4.37918976,
                4.44265628,  4.5061228 ,  4.56958931,  4.63305583,  4.69652235,
                4.75998887,  4.82345539,  4.88692191,  4.95038842,  5.01385494,
                5.07732146,  5.14078798,  5.2042545 ,  5.26772102,  5.33118753,
                5.39465405,  5.45812057,  5.52158709,  5.58505361,  5.64852012,
                5.71198664,  5.77545316,  5.83891968,  5.9023862 ,  5.96585272,
                6.02931923,  6.09278575,  6.15625227,  6.21971879,  6.2831853
                ])
1))

```

رغم ان هناك الكثير من الادوات الاخرى لأنشاء المصفوفات الا اننا سنكتفي بما شرحناه ونوضح البقية اذا احتجناها أثناء شرح بقية مكونات الكورس.

للأطلاع على كل أدوات مكتبة NumPy يمكن زيارة الرابط التالي:

[انقر هنا لزيارة صفحة NumPy على ال GitHub](#)

طباعة المصفوفات

وتتم بعدة طرق بناءً على نوع المصفوفة (أبعادها) وطريقة العرض التي نريدها (بأستخدام دالة reshape التي ذكرناها سابقاً):

```
In [24]: a=arange(5) # one dimensional array  
a
```

```
Out[24]: array([0, 1, 2, 3, 4])
```

```
In [26]: b=arange(16).reshape(4,4) # two dimensional array  
b
```

```
Out[26]: array([[ 0,  1,  2,  3],  
                [ 4,  5,  6,  7],  
                [ 8,  9, 10, 11],  
                [12, 13, 14, 15]])
```

```
In [27]: c=arange(24).reshape(2,3,4) # three dimentional array  
c
```

```
Out[27]: array([[[ 0,  1,  2,  3],  
                 [ 4,  5,  6,  7],  
                 [ 8,  9, 10, 11]],  
                [[12, 13, 14, 15],  
                 [16, 17, 18, 19],  
                 [20, 21, 22, 23]])
```

لتغيير طريقة عرض المصفوفات يمكن استخدام الابعاز التالي:

```
set_printoptions(threshold='nan')
```

والذي سيجبر مكتبة NumPy على طباعة المصفوفة كاملة

العمليات على المصفوفات

يمكن إجراء جميع العمليات على المصفوفات التي تدعمها لغات البرمجة الأخرى وكما موضح في الأمثلة التالية:

```
In [29]: a=array([[1,2,3,4],[5,6,7,8],[9,10,11,12],[13,14,15,16]])  
a
```

```
Out[29]: array([[ 1,  2,  3,  4],  
               [ 5,  6,  7,  8],  
               [ 9, 10, 11, 12],  
               [13, 14, 15, 16]])
```

```
In [31]: b=ones((4,4))  
b
```

```
Out[31]: array([[ 1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.]])
```

```
In [32]: c=a+b # array addition  
c
```

```
Out[32]: array([[ 2.,  3.,  4.,  5.],  
               [ 6.,  7.,  8.,  9.],  
               [10., 11., 12., 13.],  
               [14., 15., 16., 17.]])
```

```
In [33]: d=a-b #array subtraction  
d
```

```
Out[33]: array([[ 0.,  1.,  2.,  3.],  
               [ 4.,  5.,  6.,  7.],  
               [ 8.,  9., 10., 11.],  
               [12., 13., 14., 15.]])
```

```
In [35]: e=a*b #array multiplication  
e
```

```
Out[35]: array([[ 1.,  2.,  3.,  4.],  
               [ 5.,  6.,  7.,  8.],  
               [ 9., 10., 11., 12.],  
               [13., 14., 15., 16.]])
```

وكما هو واضح في المثال السابق فإن علامة الضرب (*) تستخدم لضرب كل عنصر من المصفوفة الأولى في مقابله من المصفوفة الثانية أو ما يسمى element-wise multiplication

ولإجراء الضرب الاعتيادي بين مصفوفتين نستخدم دالة

dot(a,b)

وهذه مجموعة اخرى من الامثلة لتوضيح الفكرة:

```
In [36]: b=b**2 # ضرب كل عنصر في نفسه او رفعه للأس 2  
b
```

```
Out[36]: array([[ 1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.],  
               [ 1.,  1.,  1.,  1.]])
```

```
In [37]: b=b*3 # ضرب كل عنصر في 3  
b
```

```
Out[37]: array([[ 3.,  3.,  3.,  3.],  
               [ 3.,  3.,  3.,  3.],  
               [ 3.,  3.,  3.,  3.],  
               [ 3.,  3.,  3.,  3.]])
```

```
In [38]: a=10*cos(a)  
a
```

```
Out[38]: array([[ 5.40302306, -4.16146837, -9.89992497, -6.53643621],  
               [ 2.83662185,  9.60170287,  7.53902254, -1.45500034],  
               [-9.11130262, -8.39071529,  0.04425698,  8.43853959],  
               [ 9.07446781,  1.36737218, -7.59687913, -9.5765948 ]])
```

```
In [39]: a=a<0 # True العنصر الذي يحقق الشرط يكون  
a
```

```
Out[39]: array([[False,  True,  True,  True],  
               [False, False, False,  True],  
               [ True,  True, False, False],  
               [False, False,  True,  True]], dtype=bool)
```

```
In [40]: c=dot(a,b) #matrix multiplication  
c
```

```
Out[40]: array([[ 9.,  9.,  9.,  9.],  
               [ 3.,  3.,  3.,  3.],  
               [ 6.,  6.,  6.,  6.],  
               [ 6.,  6.,  6.,  6.]])
```

مجموعة من الدوال الاخرى التي تعمل على المصفوفات تشمل

$$a+=3 \implies a=a+3$$

$$a-=2 \implies a=a-2$$

$$b*=a \implies b=b*a$$

وهكذا

ملاحظة: حين جمع او ضرب او طرح او اجراء اي عملية بين مصفوفتين مختلفة في النوع الناتج يأخذ النوع الاكبر (او الاكثر شمولاً) وكمثال اذا ضربنا مصفوفة اعداد صحيحة بمصفوفة اعداد مركبة يكون الناتج مصفوفة اعداد مركبة وهكذا.

a.sum()

تعطي ناتج جمع عناصر المصفوفة (a) كلها في رقم واحد

a.min()

ترجع قيمة أصغر عنصر في المصفوفة

a.max()

ترجع قيمة أكبر عنصر في المصفوفة

كل هذه العمليات تتم على المصفوفة ككل ولكن اذا اردنا ان نقوم بتنفيذها على صفوف فقط او أعمدة فقط فنستخدم دالة (axis) وكما في الامثلة التالية:

```
In [41]: k=a.sum()  
k
```

```
Out[41]: 8
```

```
In [42]: t=c.min()  
t
```

```
Out[42]: 3.0
```

```
In [44]: b=arange(12).reshape(3,4)  
b
```

```
Out[44]: array([[ 0,  1,  2,  3],  
               [ 4,  5,  6,  7],  
               [ 8,  9, 10, 11]])
```

```
In [45]: b.sum(axis=0)  #sum of each column
```

```
Out[45]: array([12, 15, 18, 21])
```

```
In [46]: b.min(axis=1)  #min of each row
```

```
Out[46]: array([0, 4, 8])
```

```
In [47]: b.cumsum(axis=0)  #commulative sum along each column
```

```
Out[47]: array([[ 0,  1,  2,  3],  
               [ 4,  6,  8, 10],  
               [12, 15, 18, 21]])
```

ملاحظة أخيرة: المجموعة التراكمي (cumulative sum) لأي صف او عمود معناه ان العنصر الاول يبقى نفسه والعنصر الثاني يكون حاصل جمع العنصر الأول والثاني والعنصر الثالث يكون حاصل جمع العنصر الأول والثاني والثالث وهكذا

بايثون-24: مكتبة NumPy الجزء الثالث

السلام عليكم

في هذا الدرس سنتحدث عن المواضيع التالية:

- الدوال الرياضية العامة التي توفرها مكتبة NumPy
- تجزئة وتشرح المصفوفات والدورات داخل المصفوفات

الدوال الرياضية العامة

إضافة إلى ما سبق ذكر من دوال الرياضيات الأساسية التي توفرها مكتبة NumPy للتعامل مع الأرقام والمصفوفات فإنها توفر مجموعة أخرى منها والتي توضحها الأمثلة التالية:

```
In [2]: from numpy import *  
a=arange(4)  
a
```

```
Out[2]: array([0, 1, 2, 3])
```

```
In [3]: b=exp(a)  
b
```

```
Out[3]: array([ 1.          ,  2.71828183,  7.3890561 , 20.08553692])
```

```
In [4]: c=sqrt(a)  
c
```

```
Out[4]: array([ 0.          ,  1.          ,  1.41421356,  1.73205081])
```

```
In [5]: d=add(b,c)  
d
```

```
Out[5]: array([ 1.          ,  3.71828183,  8.80326966, 21.81758773])
```

توضيح للصورة أعلاه:

بداية قمنا باستيراد المكتبة في السطر الأول ثم قمنا بتعريف مصفوفة تتكون من 4 عناصر في السطر الثاني

`exp()`

هي دالة الأس المعروفة

`sqrt()`

هي دالة الجذر التربيعي

`add(a,b)`

جمع مصفوفتين

واما بقية الدوال فيمكن إيجادها مع امثلة عنها في الرابط التالي:

[دليل الدوال الرياضية التي توفرها مكتبة NumPy](#)

تجزئة المصفوفات وتشرحها

بالنسبة للمصفوفات ذات البعد الواحد فيمكن الوصول إلى أي عنصر فيها من خلال الفهرس (index) الخاص به وكما في الامثلة التالية:


```

In [6]: e=arange(10)**3 # مصفوفة للعناصر بين الصفر والتسعة مرفوعة للأس الثالث
e
Out[6]: array([ 0,  1,  8, 27, 64, 125, 216, 343, 512, 729])

In [7]: e[3] # العنصر الرابع
Out[7]: 27

In [8]: e[3:6] # العناصر الرابع والخامس والسادس
Out[8]: array([ 27,  64, 125])

In [9]: e[0:4] # طباعة العناصر من الاول الى الخامس
Out[9]: array([ 0,  1,  8, 27])

In [10]: e[:4] # مثل الابعار السابق تماماً
Out[10]: array([ 0,  1,  8, 27])

In [12]: e[0:6:2]=55 # هذا الابعار معناه اجعل قيمة عناصر المصفوفة من الاول الى
# السادس 55 ولكن قم بزيادة الفهرس بمقدار 2 اي الاول والثالث والخامس وهكذا
e
Out[12]: array([ 55,  1,  55, 27,  55, 125, 216, 343, 512, 729])

```

لعكس المصفوفة نستخدم الدالة التالية:

`a[::-1]`

للتنقل بين عناصر المصفوفة يمكن استخدام (for) ايضاً وكما في المثال التالي:

```
In [15]: for i in e: # طباعة الجذر التكعيبي لعناصر المصفوفة
         print (i**(1/3.))
```

```
3.80295246076
1.0
3.80295246076
3.0
3.80295246076
5.0
6.0
7.0
8.0
9.0
```

```
In [21]: for j in e[:4]: # طباعة مربع اول 4 عناصر من المصفوفة
         print(j*j)
```

```
3025
1
3025
729
```

```
In [20]: e
```

```
Out[20]: array([ 55,  1,  55,  27,  55, 125, 216, 343, 512, 729])
```

اما بالنسبة للمصفوفات متعددة الأبعاد فيمكن استخدام نفس الخصائص أعلاه مع فرق واحد وهو ان الابعاد المتعددة تضاف الى اسم المصفوفة مفصولة بنقطتين وكما في المثال أدناه

a[3,4]

هو العنصر في الصف الرابع العمود الخامس من المصفوفة (a) وهكذا

مزيد من الامثلة في ادناه:

```
In [22]: def f(a,b): # تعريف دالة لمتغيرين
          return 5*(a+b)

          t=fromfunction (f,(3,4), dtype=int) # تعريف مصفوفة من دالة
          t
```

```
Out[22]: array([[ 0,  5, 10, 15],
                [ 5, 10, 15, 20],
                [10, 15, 20, 25]])
```

```
In [24]: t[2,3]
```

```
Out[24]: 25
```

```
In [26]: t[0:3,1:3] # طباعة العناصر للمصفوف من 2-0 والاعمدة من 2-1
```

```
Out[26]: array([[ 5, 10],
                [10, 15],
                [15, 20]])
```

```
In [27]: t[1:3,:] # طباعة كل الاعمدة للصفين الثاني والثالث
```

```
Out[27]: array([[ 5, 10, 15, 20],
                [10, 15, 20, 25]])
```

إذا تم تزويد عدد من الأبعاد أقل من الأبعاد المصفوفة فبقية الأبعاد سيتم اعتبارها كاملة (كل الصفوف أو كل الأعمدة) وكما في المثال أدناه:

t[5]

معناها الصف السادس وكل الأعمدة

t[:,3]

معناها كل الصفوف والعمود الرابع

وهكذا

كذلك للتعبير عن مصفوفة ذات ابعاد اكثر من 2 فيمكن استخدام التعبير التالي

```
a[2,2,:,:] == a[2,2,...]
```

```
b[...2] == b[:, :, :, 2]
```

أي أنه يمكن استخدام النقاط الثلاث (...) بدل اي عدد من الابعاد التي نريد التعبير عنها بدون ذكر رقم محدد.

الامثلة التالية توضح المقصود:

```
In [28]: c = array( [ [ [ 0, 1, 2], # مصفوفة ثلاثية الابعاد وهي عبارة عن
...                [ 10, 12, 13]], # مصفوفتين ثنائيتين الابعاد فوق بعضها
...
...                [[100,101,102],
...                [110,112,113]] ] )
```

```
In [29]: c.shape
```

```
Out[29]: (2, 2, 3)
```

```
In [31]: c[1,...] # نفس c[1,:,:]
```

```
Out[31]: array([[100, 101, 102],
                [110, 112, 113]])
```

```
In [32]: c[...2] # نفس c[:, :, 2]
```

```
Out[32]: array([[ 2, 13],
                [102, 113]])
```

وللتنقل وبين محتويات المصفوفة متعددة الابعاد يمكن استخدام (for) والمحور الاول وكما في المثال التالي:

```
In [33]: for row in c:  
         print(row)
```

```
[[ 0  1  2]  
 [10 12 13]]  
[[100 101 102]  
 [110 112 113]]
```

```
In [34]: for element in c.flat:  
         print(element)
```

```
0  
1  
2  
10  
12  
13  
100  
101  
102  
110  
112  
113
```

حيث نلاحظ ان المثال الأول مكننا من طباعة صفوف المصفوفة باستخدام (for) في حين اذا اردنا طباعة كل عناصر المصفوفة متعددة الابعاد فيمكن الاستعانة بدالة (flat) وكما في المثال الثاني.

واخيراً لمعرفة المزيد عن الادوات التي توفرها مكتبة NumPy للتلاعب بالمصفوفات يمكن الاستعانة بالرابط اعلاه والذي يحتوي كل الدوال للتعامل مع المصفوفات (التي شرحناها والتي لم نشرحها).

بايثون-25: مكتبة NumPy الجزء الرابع

السلام عليكم

يتناول درس اليوم المواضيع التالية:

- التلاعب بشكل المصفوفة (تغيير الشكل، دمج مصفوفتين، تجزئة مصفوفة واحدة الى عدة مصفوفات، النسخ وانواع العرض للمصفوفات).
- مجموعة من الدوال العامة للتلاعب بالمصفوفات

تغيير شكل المصفوفة:

تستخدم دالة (shape) لعرض ابعاد المصفوفة (عدد الصفوف والاعمدة والابعاد الاخرى ان وجدت) وتستخدم كما في المثال أدناه:

```
In [4]: from numpy import *
        a=(random.random((3,4))) # لتوليد مصفوفة ارقام عشوائية
        a
Out[4]: array([[ 0.21952788,  0.85474983,  0.14431023,  0.86009309],
               [ 0.83035431,  0.0472052 ,  0.4532913 ,  0.66694021],
               [ 0.37475105,  0.18329479,  0.96364733,  0.16649516]])

In [5]: a.shape
Out[5]: (3, 4)
```

والان ليتم تحويل المصفوفة الى بعد واحد نستطيع استخدام دالة (ravel) وكما في المثال التالي:

```

In [6]: a.ravel() # لتحويل المصفوفة الى بعد واحد
Out[6]: array([ 0.21952788,  0.85474983,  0.14431023,  0.86009309,  0.83035431,
                0.0472052 ,  0.4532913 ,  0.66694021,  0.37475105,  0.18329479,
                0.96364733,  0.16649516])

In [7]: a.transpose() # لقلب الصفوف اعمدة والاعمدة صفوف
Out[7]: array([[ 0.21952788,  0.83035431,  0.37475105],
               [ 0.85474983,  0.0472052 ,  0.18329479],
               [ 0.14431023,  0.4532913 ,  0.96364733],
               [ 0.86009309,  0.66694021,  0.16649516]])

In [9]: a.reshape((2,6)) # لتغيير شكل المصفوفة
Out[9]: array([[ 0.21952788,  0.85474983,  0.14431023,  0.86009309,  0.8303543
1,
                0.0472052 ],
               [ 0.4532913 ,  0.66694021,  0.37475105,  0.18329479,  0.9636473
3,
                0.16649516]])

In [12]: a.resize((6,2)) # لتغيير حجم المصفوفة
a
Out[12]: array([[ 0.21952788,  0.85474983],
                [ 0.14431023,  0.86009309],
                [ 0.83035431,  0.0472052 ],
                [ 0.4532913 ,  0.66694021],
                [ 0.37475105,  0.18329479],
                [ 0.96364733,  0.16649516]])

```

الصورة اعلاه تبين عدة أمثلة للدوال التالية:

transpose لعرض المصفوفة مع قلب الصفوف اعمدة والاعمدة صفوف

reshape لعرض شكل جديد للمصفوفة (بدون تغيير الشكل الاصلي)

resize لتغيير شكل المصفوفة الاصلية بشكل نهائي

ملاحظة: اذا استخدمنا ابعاد بأرقام سالبة فمعناها البدء بالحساب من اليمين الى اليسار (اي قلب الصفوف وقلب الاعمدة) وكما في الامثلة التالية:

```
In [15]: a=ceil(10*random.random((2,3))) # لتوليد مصفوفة عشوائية 2*3  
a
```

```
Out[15]: array([[ 1.,  6.,  4.],  
               [ 5.,  6.,  3.]])
```

```
In [18]: a[-1,-1] # this is equal to a[2,3]
```

```
Out[18]: 3.0
```

```
In [19]: a.reshape(6,-1) # تعريف احد الابعاد سالب واحد يعنى نجعل بايثون تحسبه لنا
```

```
Out[19]: array([[ 1.],  
               [ 6.],  
               [ 4.],  
               [ 5.],  
               [ 6.],  
               [ 3.]])
```

دمج عدة مصفوفات سوية:

نستطيع استخدام الأدوات التالية:

`vstack((a,b))`

وضع المصفوفة الثانية اسفل المصفوفة الاولى

`hstack((a,b))`

وضع المصفوفة الثانية الى يمين المصفوفة الاولى

`column_stack((a,b))`

يستخدم فقط للمصفوفات احادية البعد ويضع المصفوفة الثانية اسفل المصفوفة الاولى

```
row_stack((a,b))
```

يستخدم فقط للمصفوفات احادية البعد ويضع المصفوفة الثانية يمين المصفوفة الاولى

الامثلة التالية توضح ذلك:

```
In [20]: a = floor(10*random.random((2,2)))  
a
```

```
Out[20]: array([[ 6.,  3.],  
               [ 5.,  0.]])
```

```
In [21]: b = floor(10*random.random((2,2)))  
b
```

```
Out[21]: array([[ 7.,  0.],  
               [ 4.,  2.]])
```

```
In [22]: c=vstack((a,b))  
c
```

```
Out[22]: array([[ 6.,  3.],  
               [ 5.,  0.],  
               [ 7.,  0.],  
               [ 4.,  2.]])
```

```
In [23]: d=hstack((a,b))  
d
```

```
Out[23]: array([[ 6.,  3.,  7.,  0.],  
               [ 5.,  0.,  4.,  2.]])
```

```
In [25]: a=[1,2,3,4,5]  
b=[11,22,33,44,55]  
c=column_stack((a,b))  
d=row_stack((a,b))  
  
c
```

```
Out[25]: array([[ 1, 11],  
               [ 2, 22],  
               [ 3, 33],  
               [ 4, 44],  
               [ 5, 55]])
```

```
In [26]: d
```

```
Out[26]: array([[ 1,  2,  3,  4,  5],  
               [11, 22, 33, 44, 55]])
```

بالنسبة للمصفوفات ذات الأبعاد أكثر من بعدين:

hstack تقوم برزم المصفوفات بعد البعد الثاني

vstack ترزم المصفوفات بعد البعد الأول

تجزئة المصفوفة الواحدة الى أجزاء أصغر:

ويتم باستخدام دالة `hsplit` وكما في الامثلة التالية:

```
In [27]: a = floor(10*random.random((2,12)))  
a
```

```
Out[27]: array([[ 0.,  4.,  9.,  8.,  3.,  3.,  7.,  4.,  1.,  0.,  8.,  7.],  
               [ 5.,  7.,  2.,  4.,  0.,  6.,  0.,  3.,  7.,  4.,  0.,  7.]])
```

```
In [28]: hsplit(a,3) #تجزئة المصفوفة الى 3 اجزاء
```

```
Out[28]: [array([[ 0.,  4.,  9.,  8.],  
               [ 5.,  7.,  2.,  4.]])], array([[ 3.,  3.,  7.,  4.],  
               [ 0.,  6.,  0.,  3.]])], array([[ 1.,  0.,  8.,  7.],  
               [ 7.,  4.,  0.,  7.]])]
```

```
In [29]: hsplit(a,(3,4)) #تجزئة المصفوفة بعد العمود الثالث والرابع
```

```
Out[29]: [array([[ 0.,  4.,  9.],  
               [ 5.,  7.,  2.]])], array([[ 8.],  
               [ 4.]])], array([[ 3.,  3.,  7.,  4.,  1.,  0.,  8.,  7.],  
               [ 0.,  6.,  0.,  3.,  7.,  4.,  0.,  7.]])]
```

```
In [33]: vsplit(a,2) #تقسيم المصفوفة الى صفين متساويين
```

```
Out[33]: [array([[ 0.,  4.,  9.,  8.,  3.,  3.,  7.,  4.,  1.,  0.,  8.,  7.]])],  
          array([[ 5.,  7.,  2.,  4.,  0.,  6.,  0.,  3.,  7.,  4.,  0.,  7.]])]
```

نسخ المصفوفات ونسخ واجهات المصفوفات

حين نتلاعب بالمصفوفات فبعض الاحيان نقوم بنسخ المصفوفة الى مصفوفة اخرى وفي احيان اخرى يتم نسخ واجهة المصفوفة (نسخ سطحي) فقط وليست المصفوفة كعنصر كامل وفي الامثلة التالية تتضح الامور اكثر:

في حالة قول $(a=b)$ فهذا لا يعني نسخ المصفوفة كعنصر او كبيانات وانما كواجهة فقط وكما في ادناه:

```
In [34]: a=arange(12)
a
```

```
Out[34]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
In [35]: b=a # لن يتم انشاء عنصر جديد هنا وانما فقط اعطاء المصفوفة اسم ثاني
b
```

```
Out[35]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
```

```
In [36]: b is a # اختبار هل المصفوفتين هما نفس العنصر
```

```
Out[36]: True
```

```
In [37]: b.shape=3,4 # تغيير شكل المصفوفة
a.shape
```

```
Out[37]: (3, 4)
```

إذا اردنا ان ننسخ بيانات مصفوفة في مصفوفة اخرى بدون إنشاء عنصر جديد نستخدم دالة view وكما في ادناه:

```

In [38]: c=a.view()           # نسخ واجهة المصفوفة في مصفوفة اخرى
Out[38]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])

In [39]: c is a
Out[39]: False

In [40]: c.base is a        #c and a share the same view only
Out[40]: True

In [41]: c.flags.owndata    #c does not have data
Out[41]: False

In [42]: c.shape=2,6        # تغيير شكل المصفوفة لا يغير شكل المصفوفة الاخرى
Out[42]: array([[ 0,  1,  2,  3],
                [1122,  5,  6,  7],
                [ 8,  9, 10, 11]])

In [43]: a.shape
Out[43]: (3, 4)

In [48]: c[0,4]=1122
Out[48]: array([[ 0,  1,  2,  3],
                [1122,  5,  6,  7],
                [ 8,  9, 10, 11]])

In [49]: a
Out[49]: array([[ 0,  1,  2,  3],
                [1122,  5,  6,  7],
                [ 8,  9, 10, 11]])

```

واخيراً لنسخ مصفوفة بشكل كامل الى مصفوفة أخرى نستخدم دالة (copy) وكما في الامثلة التالية:

```
In [50]: d=a.copy()
```

```
In [51]: d
```

```
Out[51]: array([[ 0,  1,  2,  3],
                [1122,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [52]: d is a
```

```
Out[52]: False
```

```
In [53]: d.base is a #d is new array that shares nothing with a
```

```
Out[53]: False
```

```
In [55]: d[0,0] = 3333 #changing d does not change a
a
```

```
Out[55]: array([[ 0,  1,  2,  3],
                [1122,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

طبعاً هذه فقط مجموعة بسيطة من الدوال الكثيرة التي توفرها مكتبة NumPy للتعامل مع المصفوفات للأغراض العلمية وللإطلاع على بقية الدوال يمكن زيارة الروابط التالية:

[لأنشاء المصفوفات](#)

[arange](#), [array](#), [copy](#), [empty](#), [empty_like](#), [eye](#), [fromfile](#), [fromfunction](#), [identity](#), [linspace](#), [logspace](#), [mgrid](#), [ogrid](#), [ones](#), [ones_like](#), [r](#), [zeros](#), [zeros_like](#)

للتحويل بين المصفوفات

[astype](#), [atleast 1d](#), [atleast 2d](#), [atleast 3d](#), [mat](#)

التلاعب بالمصفوفات

array split, column stack, concatenate, diagonal, dsplit, dstack, hsplit, hstack, item,
newaxis, ravel, repeat, reshape, resize, squeeze, swapaxes, take, transpose, vsplit,
vstack

للسؤال عن حالة المصفوفات

all, any, nonzero, where

ترتيب المصفوفة

argmax, argmin, argsort, max, min, ptp, searchsorted, sort

العمليات على المصفوفات

choose, compress, cumprod, cumsum, inner, fill, imag, prod, put, putmask, real, sum

احصائيات اولية لبيانات المصفوفات

cov, mean, std, var

دوال الجبر الخطي للمصفوفات

cross, dot, outer, svd, vdot

بايثون-26: شرح مكتبة NumPy الجزء الخامس

السلام عليكم

شرحنا في الدروس الاربعة الماضية الكثير من الادوات المفيدة التي توفرها مكتبة (NumPy) العلمية لتسهيل التعامل مع المصفوفات بلغة بايثون وارفقنا الروابط للكثير من الادوات الاخرى التي لم يتم شرحها لأن هناك المئات منها ولو اردنا شرحها كلها فسيستغرق ذلك مدة طويلة جداً ولذا سيكون هذا الجزء هو الاخير من اجزاء شرح مكتبة (NumPy) وسنرفق معه روابط لأمثلة تشرح بقية الادوات التي لن نشرحها ليستطيع المهتمون الرجوع لها عند الحاجة كما سنشرح اي اداة لم يتم شرحها في هذه الاجزاء الخمسة حين نصل الى التطبيق العملي ونحتاجها.

سنتناول في هذا الدرس المواضيع التالية:

- فهارس المصفوفات والتلاعب بها واستخدامها بشكل متقدم.
- دالة `ix()`
- دوال الجبر الخطي
- مقارنة الانواع المختلفة للمصفوفات
- دوال متنوعة

الفهارس و خدع استخدامها:

بالإضافة الى فهرسة المصفوفات بالطرق الاعتيادية السابقة فإنه يمكننا استخدام مصفوفة للفهارس وكما في الامثلة التالية:

```
In [1]: from numpy import *
a=arange(12)
a
Out[1]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

In [2]: a[0] # استخدام الفهارس بشكل طبيعي
Out[2]: 0

In [3]: a[5]
Out[3]: 5

In [4]: a[6]+a[2]
Out[4]: 8

In [5]: b=arange(12)**2 # تربيع اول 12 رقم صحيح ابتداءً من الصفر
b
Out[5]: array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81, 100, 121])

In [7]: i=array([1,1,3,4,8,5]) # مصفوفة للفهارس
b[i] # استخدام مصفوفة كفهرس لمصفوفة ثنائية
Out[7]: array([ 1,  1,  9, 16, 64, 25])

In [8]: c=array([ [ 3, 4], [ 9, 7 ] ]) # مصفوفة ثنائية الابعاد
b[c] # يمكن استخدام مصفوفة ثنائية الابعاد كفهرس لمصفوفة احادية
Out[8]: array([[ 9, 16],
               [81, 49]])
```

اما اذا كانت مصفوفة الفهرس متعددة الأبعاد لاستخدامها لفهرسة مصفوفة احادية البعد فإن البعد الاول منها سيستخدم فقط وكما في المثال التالي الذي يبين كيفية تحويل اسماء الالوان في صورة الى الالوان:

```
In [9]: palette = array( [ [0,0,0],          # black
...                       [255,0,0],       # red
...                       [0,255,0],       # green
...                       [0,0,255],       # blue
...                       [255,255,255] ] ) # white
```

```
In [10]: image = array( [ [ 0, 1, 2, 0 ], [ 0, 3, 4, 0 ] ] )
# كل قيمة في هذه الصورة تقابل لون معين في اللوحة اعلاه
```

```
In [11]: palette[image]
```

```
Out[11]: array([[ [ 0,  0,  0],
                  [255,  0,  0],
                  [  0, 255,  0],
                  [  0,  0,  0]],
                [[ [ 0,  0,  0],
                  [  0,  0, 255],
                  [255, 255, 255],
                  [  0,  0,  0]])
```

ملاحظة: لدمج او التعامل مع مصفوفات مختلفة الأبعاد يمكن توحيد اشكالها بأستخدام دالة (reshape) التي تم شرحها سابقاً.

يمكننا ايضاً اعطاء فهرس (indexes) لأكثر من بعد من أبعاد المصفوفة الا ان هذه الفهارس يجب ان تكون بنفس الحجم وكما في الامثلة التالية:

```
In [12]: a = arange(12).reshape(3,4)
a
```

```
Out[12]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [13]: i = array( [ [0,1], [1,2] ] ) #فهارس البعد الاول من المصفوفة
j=array ([[2,1],[3,3]]) #فهارس البعد الثاني للمصفوفة
a[i,j]
```

```
Out[13]: array([[ 2,  5],
                [ 7, 11]])
```

```
In [14]: a[i,2]
```

```
Out[14]: array([[ 2,  6],
                [ 6, 10]])
```

```
In [15]: a[:,j]
```

```
Out[15]: array([[ [ 2,  1],
                  [ 3,  3]],
                [[ 6,  5],
                  [ 7,  7]],
                [[10,  9],
                  [11, 11]])
```

```
In [16]: k=[i,j] #يمكننا ايضاً وضع الابعاد في مصفوفة واحدة ثم استخدامها
a[k]
```

```
Out[16]: array([[ 2,  5],
                [ 7, 11]])
```

يمكن ايضاً استخدام فهارس تتكون من قيم منطقية (صفر او واحد) او (صح او خطأ) وكما في الامثلة التالية:

```
In [17]: a = arange(12).reshape(3,4)
         b = a > 4 # مصفوفة القهارس هي نفس حجم المصفوفة الاصلية ولكن مشروطة منطقياً
```

```
In [18]: a
```

```
Out[18]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [19]: b
```

```
Out[19]: array([[False, False, False, False],
                [False,  True,  True,  True],
                [ True,  True,  True,  True]], dtype=bool)
```

```
In [20]: a[b] # قيم المصفوفة التي توافق شرط القهارس
```

```
Out[20]: array([ 5,  6,  7,  8,  9, 10, 11])
```

```
In [21]: a[b]=0 # نجعل كل قيم المصفوفة التي تحقق شرط القهارس تساوي صفر
         a
```

```
Out[21]: array([[0,  1,  2,  3],
                [4,  0,  0,  0],
                [0,  0,  0,  0]])
```

دالة ix_()

وتستخدم لدمج متجهات مختلفة للحصول على نتيجة تحتوي ازواج مرتبة من مكونات المتجهات المختلفة وكما تبينه الامثلة التالية:

```
In [23]: a = array([2,3,4,5])
         b = array([8,5,4])
         c = array([5,4,6,8,3])
         ax,bx,cx = ix_(a,b,c)
```

```
In [24]: ax
```

```
Out[24]: array([[2],
                [3],
                [4],
                [5]])
```

```
In [25]: bx
```

```
Out[25]: array([[8],
                [5],
                [4]])
```

```
In [26]: cx
```

```
Out[26]: array([[5, 4, 6, 8, 3]])
```

```
In [27]: ax.shape, bx.shape, cx.shape
```

```
Out[27]: ((4, 1, 1), (1, 3, 1), (1, 1, 5))
```

```
In [28]: result = ax+bx*cx      #result of a+b*c
         result
```

```
Out[28]: array([[42, 34, 50, 66, 26],
                [27, 22, 32, 42, 17],
                [22, 18, 26, 34, 14]],

               [[43, 35, 51, 67, 27],
                [28, 23, 33, 43, 18],
                [23, 19, 27, 35, 15]],

               [[44, 36, 52, 68, 28],
                [29, 24, 34, 44, 19],
                [24, 20, 28, 36, 16]],

               [[45, 37, 53, 69, 29],
                [30, 25, 35, 45, 20],
                [25, 21, 29, 37, 17]])
```

النوع الاخير من الفهرسة يتم بأستخدام السلاسل الرمزية ويمكن ايجاد تفاصيل اكثر عنه في [الرابط التالي string indexing](#)

الحبر الخطي (Linear Algebra):

ويتم استخدامها باستدعاء المكتبة الفرعية التالية:

```
from numpy.linalg import *
```

وهذه بعض الامثلة للدوال التي تدعمها هذه المكتبة:

```
a.transpose()
```

لأعطاء ترانسبوز المصفوفة (قلب الصفوف الى اعمدة والاعمدة الى صفوف)

```
inv(a)
```

لإعطاء معكوس المصفوفة inverse

```
eye(2)
```

تعطي مصفوفة التعريف التي تحتوي واحد في القطر الرئيسي واصفار في بقية اجزائها والتي تبدو كما في ادناه:

```
1 0
```

```
0 1
```

dot(j,z)

لضرب المصفوفات ضرب نقطي dot product

solve(a,y)

حل المعادلة الخطية (y) لإيجاد قيمة (a)

trace(a)

مجموعة القيم في القطر الرئيسي للمصفوفة (a)

eig(b)

إرجاع قيمة ال (eigenvalue) للمصفوفة

ويمكن إيجاد تطبيق هذه الدوال في الامثلة التالية:

```

>>> from numpy import *
>>> from numpy.linalg import *

>>> a = array([[1.0, 2.0], [3.0, 4.0]])
>>> print a
[[ 1.  2.]
 [ 3.  4.]]

>>> a.transpose()
array([[ 1.,  3.],
       [ 2.,  4.]])

>>> inv(a)
array([[ -2. ,  1. ],
       [ 1.5, -0.5]])

>>> u = eye(2) # unit 2x2 matrix; "eye" represents "I"
>>> u
array([[ 1.,  0.],
       [ 0.,  1.]])
>>> j = array([[0.0, -1.0], [1.0, 0.0]])

>>> dot(j, j) # matrix product
array([[ -1.,  0.],
       [ 0., -1.]])

>>> trace(u) # trace
2.0

>>> y = array([[5.], [7.]])
>>> solve(a, y)
array([[ -3.],
       [ 4.]])

>>> eig(j)
(array([ 0.+1.j,  0.-1.j]),
array([[ 0.70710678+0.j,  0.70710678+0.j],
       [ 0.00000000-0.70710678j,  0.00000000+0.70710678j]])
Parameters:
  square matrix

Returns
  The eigenvalues, each repeated according to its multiplicity.

  The normalized (unit "length") eigenvectors, such that the
  column ``v[:,i]`` is the eigenvector corresponding to the
  eigenvalue ``w[i]`` .

```

إضافة إلى الهيكل البياني المسمى (array) هناك كلاس آخر ضمن مكتبة (numpy) اسمه (matrix) وله مواصفات مختلفة إلى حد ما عن مواصفات (array) وهذه بعض من أمثلة استخدامه:


```
In [29]: A = matrix('1.0 2.0; 3.0 4.0')
A
```

```
Out[29]: matrix([[ 1.,  2.],
                 [ 3.,  4.]])
```

```
In [30]: type(A)    #المكان الذي خزنت فيه المصفوفة
```

```
Out[30]: numpy.matrixlib.defmatrix.matrix
```

```
In [31]: A.T    #transpose of A
```

```
Out[31]: matrix([[ 1.,  3.],
                 [ 2.,  4.]])
```

```
In [32]: X = matrix('5.0 7.0')
Y = X.T
Y
```

```
Out[32]: matrix([[ 5.],
                 [ 7.]])
```

```
In [33]: A*Y
```

```
Out[33]: matrix([[ 19.],
                 [ 43.]])
```

```
In [34]: A.I    #inverse of matrix
```

```
Out[34]: matrix([[ -2. ,  1. ],
                 [ 1.5, -0.5]])
```

الفروق الرئيسية بين (array) و (matrix) ضمن مكتبة (NumPy):

1- ال (array) هي مصفوفات ذات عدة ابعاد (تتراوح بين بعد واحد واي عدد من الأبعاد) في حين ال (matrix) هي مصفوفات ذات بعدين عموماً.

2- الفهارس للنوعين تعمل بنفس الطريقة ولها نفس الشروط

3- المصفوفات من نوع (matrix) يمكن أن تعمل كفهرس لمصفوفات من نفس النوع ولكن الافضل استخدام (array) او (list) لذلك.

4- كل من النوعين لها فهرسة تبدأ بصفر كبقية فهارس بايثون.

نصائح وخدع عامة لمكتبة NumPy

إعادة التشكيل بشكل اوتوماتيكي

ذكرنا سابقاً ان استخدام الفهرس (-1) يعني استخدم اي عدد بحسب الحاجة وهذا المثال يوضح الفكرة:

إذا كانت لديك مصفوفة من 30 عنصر و اردت ان تجعل لها عدد محدد من الصفوف وعدد محدد من الابعاد ولكنك لا تريد تحديد بقية القيم فيمكن اتباع الطريقة التالية:

```
In [37]: a = arange(30) # مصفوفة فيها 30 عنصر  
>>> a.shape = 2, -1, 3 # قيمة سالبة واحد معناها استخدم الرقم المناسب  
>>> a.shape
```

```
Out[37]: (2, 5, 3)
```

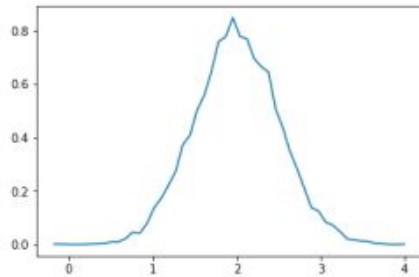
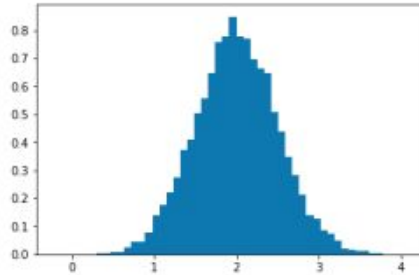
```
In [38]: a
```

```
Out[38]: array([[ 0,  1,  2],  
               [ 3,  4,  5],  
               [ 6,  7,  8],  
               [ 9, 10, 11],  
               [12, 13, 14]),  
              [[15, 16, 17],  
               [18, 19, 20],  
               [21, 22, 23],  
               [24, 25, 26],  
               [27, 28, 29]])
```

واخيراً لرسم البيانات باستخدام مكتبة (numpy) نستخدم دالة ال (histogram) والتي حين تطبيقها على مصفوفة معينة ترجع زوجاً من المتجهات وهي هستوغرام المصفوفة و متجه الفهارس. لاحظ ان مكتبة (matplotlib) تحتوي على هستوغرام ايضاً (اسمه hist مثل الماتلاب) والذي يختلف عن هستوغرام الخاص بال(numpy) والفرق الرئيسي ان دالة هستوغرام الخاصة بمكتبة (numpy) يرجع البيانات فقط في حين هستوغرام ال(matplotlib) يرسم البيانات مباشرة.

وفي المثال ادناه يتبين كيفية استخدام ورسم الهستوغرام:

```
In [39]: import numpy
import pylab
# Build a vector of 10000 normal deviates with variance 0.5^2 and mean 2
mu, sigma = 2, 0.5
v = numpy.random.normal(mu, sigma, 10000)
# Plot a normalized histogram with 50 bins
pylab.hist(v, bins=50, normed=1) # matplotlib version (plot)
pylab.show()
# Compute the histogram with numpy and then plot it
(n, bins) = numpy.histogram(v, bins=50, normed=True) # NumPy version (no plot)
pylab.plot(.5*(bins[1:]+bins[:-1]), n)
pylab.show()
```



الى هنا ينتهي الحديث عن مكتبة (NumPy) في دورتنا المستمرة ان شاء الله وفي الدرس القادم سنبدأ الحديث عن مكتبة اخرى من مكاتب بايثون العلمية والتي سنستخدمها في أمثلتنا المستقبلية عن علم البيانات وهي مكتبة (SciPy) والتي يمكن إيجاد كافة التفاصيل عنها في موقعها الرسمي (scipy.org)

بايثون-27: اساسيات مكتبة (SciPy) الخاصة بلغة بايثون

السلام عليكم

في المحاضرات السابقة شرحنا مكتبة (NumPy) وبيننا الكثير من فوائدها في مجال البرمجة العلمية والتعامل مع البيانات الكثيرة وخصوصاً المصفوفات واليوم وكما وعدناكم نبدأ على بركة الله شرح مكتبة علمية اخرى للغة بايثون الا وهي مكتبة (SciPy).

بداية قد يرد السؤال عن ما هي فائدة هذه المكتبات؟

والجواب ببساطة ان هذه المكتبات توفر ادوات ودوال تسهل وتسرع عملية التعامل مع البيانات الكثيرة التي ننوي العمل عليها في الامثلة العملية التي سيتم شرحها بعد الانتهاء من شرح الادوات لعلم البيانات بلغة بايثون.

وبصورة عامة فأن التعامل مع علم البيانات والكميات الكبيرة من البيانات يحتاج بشكل عام معرفة في الجبر الخطي (linear algebra) وهو بالضبط ما سنركز عليه في شرح هذه المكتبة.

مقارنة بين (NumPy) و (SciPy)

مما سبق فالمفروض اننا عرفنا ان مكتبة (NumPy) هي مكتبة رائعة في مجال الحسابات العلمية بلغة بايثون حيث انها تحتوي على مجموعة من الأدوات والتقنيات التي يمكن أن تستخدم لحل النماذج الرياضية في الحاسوب. ولكن الاداة الاكثر استخداماً في علم البيانات هي المصفوفات متعددة الأبعاد عالية الاداء والتي توفرها كل من المكتبتين.

بشكل مختصر فإن مكتبة (SciPy) هي نفس مكتبة (NumPy)

وهي ايضاً من بين المكتبات الاساسية للحسابات العلمية وخوارزميات الرياضيات والدوال المعقدة. ولكنها مبنية كأمتداد لمكتبة (NumPy) مما يعني انهما يستخدمان سوية في اغلب الاحيان.

و سنتعرف على المزيد حول ذلك في هذا الدرس ان شاء الله.

مبدئياً وقبل الخوض في التفاصيل، فمن الضروري معرفة ما الذي نحتاجه من هذه المكتبة لعلم البيانات (حيث انها مكتبة ضخمة وشرح كل تفاصيلها قد يأخذ وقتاً طويلاً جداً ولذلك سنركز على ما نحتاجه فقط) وهذه الامور تشمل هياكل المصفوفات وكيفية التعامل مع الانواع المختلفة من البيانات وكيفية التلاعب بأشكال المصفوفات واخيراً دوال الجبر الخطي.

بخصوص انشاء المصفوفات والتعامل معها والتلاعب بشكلها وحجمها فقد تم شرح كل ذلك في الدروس السابقة والرابط التالي يحتوي على ملخص لكل الدوال التي توفرها مكتبة (NumPy):

[انقر هنا لزيارة NumPy Cheat sheet](#)

ولذلك سنقفز مباشرة إلى الجبر الخطي والتعامل مع دواله في مكتبة (SciPy) فتابعوا معنا:

قبل البدء يجدر الاشارة الى ان مكتبة (SciPy) تأتي منسبة بالفعل مع الاناكوندا ويمكن التأكد من ذلك من خلال استيراد المكتبة ثم فحص الاصدار الخاص بها وكما في المثال التالي:

```
In [1]: from numpy import *
import scipy
scipy.__version__

Out[1]: '0.19.1'
```

السطر الاول لأستيراد مكتبة NumPy

السطر الثاني لاستيراد مكتبة SciPy

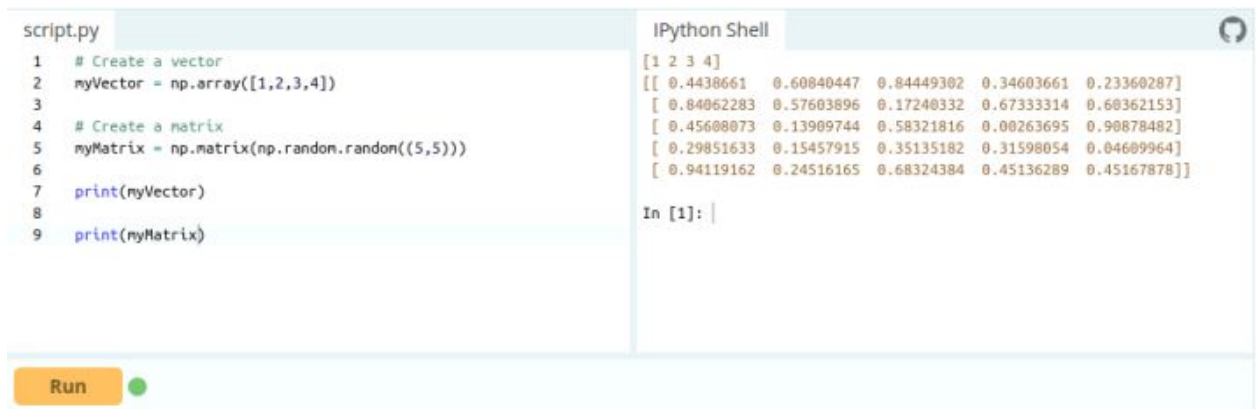
السطر الثالث لعرض الاصدار الخاص بمكتبة SciPy

والان نبدأ التعامل مع ادوات هذه المكتبة:

اساسيات المتجهات والمصفوفات:

فضاء المتجه (Vector Space): وهو مفهوم أساسي في الجبر الخطي وهو عبارة عن مجموعة من المتجهات بحيث يمكنك جمع متجهين بدون ان يخرج الناتج من فضاء المتجه. علماً ان المتجه قد يكون صف او عمود في المصفوفة.

إنشاء المتجهات والمصفوفات باستخدام مكتبة (NumPy): ويمكن عمل ذلك كما تعلمنا في الدروس السابقة او باستخدام الطريقة التالية:



```
script.py
1 # Create a vector
2 myVector = np.array([1,2,3,4])
3
4 # Create a matrix
5 myMatrix = np.matrix(np.random.random((5,5)))
6
7 print(myVector)
8
9 print(myMatrix)

IPython Shell
[1 2 3 4]
[[ 0.4438661  0.60840447  0.84449302  0.34603661  0.23360287]
 [ 0.84062283  0.57603896  0.17240332  0.67333314  0.60362153]
 [ 0.45608073  0.13909744  0.58321816  0.00263695  0.90878482]
 [ 0.29851633  0.15457915  0.35135182  0.31598054  0.04609964]
 [ 0.94119162  0.24516165  0.68324384  0.45136289  0.45167878]]

In [1]: |
```

عند التعامل مع مصفوفات تتكون اغلب عناصرها من اصفار فتلك المصفوفات نسميها مصفوفات متفرقة (Sparse matrix) وأما المصفوفات التي تحتوي على عدد قليل من الأصفار فنسميها المصفوفات الكثيفة (Dense Matrices)

للهولة الاولى قد يبدو هذا الشيء غير مفيد وهي مجرد اختلاف في الأسماء ولكن هنا تبرز اهمية المكتبات العلمية التي تراعي هذه الامور بدقة وتستخدمها لتسريع التنفيذ وهنا تظهر لنا عدة خيارات:

scipy.linalg ونستخدمها مع المصفوفات الكثيفة

scipy.sparse ونستخدمها مع المصفوفات المتفرقة

وهناك عدة عمليات يمكن انجازها على كل نوع من المصفوفات:

```
In [6]: import numpy as np
        from scipy.sparse import csr_matrix
        csr_matrix((3, 4), dtype=np.int8).toarray()
```

```
Out[6]: array([[0, 0, 0, 0],
               [0, 0, 0, 0],
               [0, 0, 0, 0]], dtype=int8)
```

```
In [7]: row = np.array([0, 0, 1, 2, 2, 2])
        col = np.array([0, 2, 2, 0, 1, 2])
        data = np.array([1, 2, 3, 4, 5, 6])
        csr_matrix((data, (row, col)), shape=(3, 3)).toarray()
```

```
Out[7]: array([[1, 0, 2],
               [0, 0, 3],
               [4, 5, 6]], dtype=int64)
```

العمليات على المتجهات:

كما تم رؤيته سابقاً في مكتبة (NumPy) فإن المتجهات وهي مصفوفات احادية البعد يمكن جمعها وطرحها وإجراء العمليات الرياضية البقية عليها بشرط ان تكون بنفس الحجم للجمع والطرح وهكذا.

```
In [8]: print(row+col)
```

```
[0 2 3 2 3 4]
```

```
In [9]: print(row-col)
```

```
[ 0 -2 -1  2  1  0]
```

```
In [10]: print(row*col)  # ضرب كل عنصر بالعنصر المقابل له
```

```
[0 0 2 0 2 4]
```

```
In [11]: print(row**col)
```

```
[1 0 1 1 2 4]
```

```
In [13]: print(np.dot(row,col))  #dot product
```

```
8
```

بايثون-28: تحليل البيانات بلغة بايثون المشروع الاول-1

السلام عليكم ورحمة الله وبركاته

اخوتي الكرام اخواتي الفاضلات

بعد ان شرحنا اساسيات لغة بايثون والكثير من الامثلة عن البرامج البسيطة باستخدامها وبدأنا بشرح مكتبات بايثون لتحليل البيانات في الدروس الماضية وردتنا عدة طلبات للتركيز أكثر على الامثلة والمشاريع الحقيقية لتحليل البيانات بلغة بايثون (وهو جزء من علم البيانات وتطبيقاته) وقد ارتأينا أن نستغل ما تراكم لدينا من معرفة في الدروس السابقة لنبدأ على بركة الله المشروع الاول لتحليل البيانات فلنبدأ بلا اي تأخير:

كالعادة وفي بداية كل درس من هذا الكورس احب التذكير بأهمية الاطلاع المتسلسل على محتويات الكورس لضمان فهم المادة المشروحة اي انه يفضل ان يتم الاطلاع على كل الدروس السابقة قبل الاطلاع على هذا الدرس.

بعد ان قمنا بشرح عدة مكتبات (ولو بشكل مبسط) لتحليل البيانات بلغة بايثون يجدر بالذكر ان هناك مكتبات اخرى مهمة وضرورية لأستيراد وعرض ومعالجة وتحليل البيانات والخروج بأستنتاجات حولها (وهذا هو جوهر علم البيانات اي تحويل البيانات الى استنتاجات وقيمة للشركة او المؤسسة) وفي ادناه قائمة بأهم المكتبات التي تستخدم لهذا الغرض:

1- مكتبة (NumPy) وقد تحدثنا عنها في الدروس السابقة.

2- مكتبة (SciPy) وقد ذكرنا اساسياتها في الدروس السابقة ايضاً.

3- مكتبة (Matplotlib) وتستخدم لأغراض رسم البيانات بطرق متعددة بدءاً من الهستوغرام (histogram) الى الرسم البياني بخط واحد الى الرسومات الحرارية وهكذا. في حالة اردنا تضمين

الرسومات في نفس بيئة الاناكوندا (التي تحدثنا عنها سابقاً وقلنا انها البيئة التي سنعتمدها في الشرح) فيمكن عمل ذلك بكتابة الايغاز التالي قبل البدء بالرسم:

```
ipython notebook –pylab = inline
```

واما ان لم نستخدم هذا الايغاز فبمجرد تنفيذ اي ايغاز رسم من هذه المكتبة فستتحول بيئة الاناكوندا الى بيئة مشابهة للماتلاب وعندها قد تختلف بعض الواجهات التي تعودنا عليها ولكنها قد تصبح اكثر ملائمة للمحترفين بأستخدام الماتلاب وهنا يبقى الخيار لكم في عمل ما يناسبكم. يجدر بالاشارة اننا نستطيع استخدام المحرر العلمي (Latex) لأضافة المعادلات المعقدة الى بيئة اناكوندا لرسمها بأستخدام هذه المكتبة.

4- مكتبة (Pandas): وتستخدم بشكل رئيسي لمعالجة والتلاعب بالبيانات المهيكلة (المرتبة في جداول) وسنقوم بأستخدامها بكثرة لتهيئة وتحضير البيانات للخطوات اللاحقة.

5- مكتبة (Scikit Learn): والتي تختص بأدوات تعليم الماكنة (machine learning) بلغة بايثون وقد تم بنائها بأستخدام المكتبات الثلاثة الاولى في هذه القائمة. وتحتوي الكثير من الادوات للتعامل مع البيانات مقل التصنيف والتجميع وتقليل الابعاد وغيرها من خطوات معالجة البيانات.

6- مكتبة (Statmodels) للنمذجة الاحصائية.

7- مكتبة (Seaborn): لعرض البيانات الاحصائية. وقد تم بنائها بأستخدام المكتبة الرابعة أعلاه.

8- مكتبة (Bokeh): لإنشاء الرسومات التفاعلية ولوحات التحكم وتطبيقات البيانات للمتصفحات الحديثة حيث أنها تمكن المستخدم من انشاء رسومات دقيقة بصيغة (D3.js) ويمكنها أيضاً التعامل بسرعة كبيرة مع كميات هائلة من البيانات حيث أنها توفر أداء تفاعلي عالي (high-performance interactivity).

9- مكتبة (Blaze): لزيادة فعالية مكتبة (numpy) ومكتبة (pandas) للتعامل مع البيانات الموزعة وبيانات الزمن الحقيقي حيث انها تسمح بالوصول الى البيانات من عدة مصادر ومنها قواعد البيانات مثل (Bcloz, MongoDB, SQLAlchemy, Apache, Spark, PyTables, ...etc). وتعمل بالتعاون مع بقية المكتبات لإنشاء لوحات تحكم وعروض دقيقة للبيانات.

10- مكتبة (Scrapy): التحليل الشبكي (web crawling) حيث انها تعتبر إطار عمل فعال للحصول على انماط معينة من البيانات ويمكنها ان تبدأ من الصفحة الرئيسية لأي موقع انترنت للبحث عن بيانات معينة بداخل ذلك الموقع ولكل صفحاته.

11- مكتبة (SymPy): للحسابات الرمزية وتحتوي ادوات كثيرة تبدأ من الرياضيات البسيطة الى الجبر الى الرياضيات المتقطعة والفيزياء الكمية وايضاً يمكن استخدامها مع محرر النصوص العلمي (Latex).

12- مكتبة (Requests): للوصول الى الويب وتعمل بشكل مشابه لمكتبة بايثون القياسية (urllib2) ولكنها اسهل كثيراً من ناحية البرمجة.

من المكتبات الاخرى التي تتوفر للمبرمجين بلغة بايثون اليوم (والتي قد نحتاجها في بعض الامثلة والمشاريع) هي مكتبة (os) لنظم التشغيل والتعامل مع الملفات ومكتبة (networkx) و (igraph) للتعامل مع البيانات الرسومية ومكتبة (regular expressions) لإيجاد صيغ معينة في البيانات باستخدام التعابير القياسية التي تحدثنا عنها في الدروس السابقة واخيراً مكتبة (BeautifulSoup) (المشابهة لمكتبة (Scrapy) ولكنها تعمل للحصول على البيانات بصيغة معينة من صفحة ويب واحدة بدلاً من موقع كامل).

والان بعد ان عرفنا اساسيات لغة بايثون ومكتباتها القياسية لتحليل البيانات نصل الى المرحلة الالهة وهي تحليل البيانات باستخدام هذه الادوات ويتم ذلك عموماً بثلاث مراحل:

1- استكشاف البيانات (Data Exploration): ويتم ذلك عادة بعرض البيانات بصيغ وطرق مختلفة لمحاولة فهمها قبل البدء بالتلاعب بها وتحليلها.

2- تنظيف البيانات (Data Cleaning and preparation): ويتم خلال هذه المرحلة تنظيف البيانات لتهيئتها للتحليل الاحصائي.

3- النمذجة التنبؤية (Predictive Modeling): وهنا نقوم باستخدام الخوارزميات الحقيقية لتحليل البيانات والحصول على استنتاجات وتنبؤات مستقبلية منها. (هنا يبدأ المرح الحقيقي)

استكشاف البيانات باستخدام مكتبة (Pandas):

قبل البدء باستكشاف البيانات لا بد من معرفة بعض الاساسيات عن مكتبة (Pandas) وكما يلي:

هناك مفهومين اساسيين في مكتبة (Pandas) للتعامل مع البيانات وهما السلسلة (series) وهيكل البيانات (dataframe) حيث تعرف السلسلة على انها صف او عمود من البيانات احادي البعد (1D) ومفهرس بحيث ان كل صف او كل عمود يحتوي عنصر واحد فقط. اما هيكل البيانات فهو يشبه ملف الاكسل (Excel) وهو بصورة عامة بيانات متعددة الابعاد تتكون من صفوف (rows) واعمدة (columns) ويمكن الوصول الى اي عنصر من خلال فهرس الصف وفهرس العمود (row index and column index).

ولمزيد من المعلومات عن هذه المكتبة يمكن زيارة الرابط التالي الذي يشرح أهم أدواتها للمهتمين ([انقر هنا](#)).

والان لتنزيل البيانات التي سنعمل عليها يمكن ببساطة النقر على الرابط التالي ([انقر هنا لتنزيل البيانات](#))

حيث من المفترض ان تجدوا الملفات التالية في المجلد المضغوط في الرابط اعلاه:

train.csv

وهو ملف البيانات المستخدمة للتدريب وامتداده (CSV.) مختصر لعبارة

comma separated values أو القيم المفصولة بفارزة وهو امتداد شائع جداً في عالم البيانات

test.csv

وهو الملف المستخدم لاختبار النموذج الذي سنقوم بتطويره في مشروعنا

sample submission.csv

وهو مثال لشكل الملف الذي يفترض أن نقوم بتسليمه في نهاية المشروع

والسبب وراء وجود ملف مثال للتسليم هو ان مشروعنا هذا هو جزء من مسابقة عالمية لتحليل البيانات ويمكن الوصول إليه من الرابط التالي:

[Analytics Vidhya Loan Prediction Practice Problem](#)

حيث يمكن الدخول للموقع اعلاه والتسجيل فيه للاشتراك في المسابقة وربما نيل الجوائز في حالة الفوز

وبعد التسجيل في الموقع يمكن مشاهدة النافذة التالية والخاص بمشروعنا:

والان وبعد ان قمنا بتنزيل ملفات المشروع والتعرف على الموقع الذي نستطيع من خلاله الحصول على البيانات لهذا المشروع (والمشاريع المستقبلية ان شاء الله) نترككم برعاية الله وحفظه ونكمل معكم ان شاء الله في الدرس القادم حيث سنبدأ الخطوة الاولى في مشروعنا وهي **استكشاف البيانات**

هذا الدرس والدروس القادمة هي ترجمة (بتصرف وتبسيط) للكورس المشروح في الرابط التالي

[انقر هنا](#)

بايثون-29: تحليل البيانات بلغة بايثون المشروع الاول-2

السلام عليكم ورحمة الله وبركاته

اخوتي الكرام اخواتي الفاضلات

وصلنا معكم الى المرحلة الاولى من مراحل تحليل البيانات الا وهي استكشاف البيانات (data exploration) ولكن قبل ذلك احب ان ابين عدة أمور:

بعد ان قمنا بتنزيل الملفات التي سنستخدمها في مشروعنا في الدرس السابق وقلنا انها ستكون ملف بيانات التدريب (training) وملف بيانات الاختبار (testing) وملف لنموذج التسليم ووضحنا معاني هذه الملفات باختصار احببت ان ابين الموضوع بشكل أكثر تفصيلاً هنا:

الهدف من تحليل البيانات بشكل عام هو إنشاء نموذج (model) قادر على تحليل البيانات المستقبلية بناءً على معطيات حالية او ماضية بمعنى اننا نريد ان نبني نموذج يتدرب على بيانات سابقة لدينا نعرف نتائجها ثم نقوم بتطبيق هذا النموذج على بيانات جديدة لم يسبق له ان تعامل معها بهدف معرفة ما هي مخرجات كل منها وسنحاول توضيح هذين المعنيين (التدريب والاختبار) اكثر اثناء شرح المشروع وخطوات تحليله.

ايضاً يجدر الاشارة الى **المطلوب في مشروعنا** هذا وهو باختصار (ويمكن إيجاد معلومات اكثر تفصيلاً عنه في صفحة المشروع المذكورة في الدرس السابق):

هنا تم اعطائنا بيانات مواطنين قاموا بالاقتراض من مصرف معين مع حالة ارجاع القرض (هل تم تسديد القرض في وقتها أم لا) والمطلوب منا ايجاد علاقة بين ارجاع القرض وبقية المعلومات الاخرى (كمثال: هل ان الحالة الزوجية او معدل الدخل للمقترض يحدد باحتمالية كبيرة هل سيقوم المقترض بأرجاع القرض او لا؟) ونفس الشيء ينطبق على بقية المتغيرات الاخرى والفائدة من معرفة هذه العلاقة لكي نقوم ببناء نموذج التخمين (predictive model) ليقوم بأخبارنا في المستقبل هل ان المقترضين الجدد (الذي سيقومون بتقديم طلب للأقتراض) سيقومون بأرجاع القرض او لا؟ ويتم هذا بأدخال بياناتهم (التي سيقومون بتزويدنا بها) الى

النموذج الذي سنبنيه في هذا الكورس وطبعاً يتم كل ذلك بمبدأ التخمين الاحصائي المعتمد على الاحتمالية.

الان وكما قلنا في الدرس السابق نبدأ برنامج اناكوندا من سطر الأوامر بطباعة العبارة

```
ipython notebook
```

بشكل طبيعي كما تعلمنا سابقاً وفي داخل متصفح الانترنت حيث تظهر لنا الواجهة الرئيسية للبرنامج نقوم بإنشاء ملف جديد من نوع (python 3) ونقوم بتسميته بأي اسم ثم نقوم بكتابة هذا الايعاز

```
%pylab inline
```

لنقوم بجعل كل النتائج تعرض في نفس النافذة وليس في نوافذ اخرى (كما اشرنا الى ذلك في الدرس الماضي) ويقوم الايعاز اعلاه باستيراد الدوال الرسومية والرياضية لمكتبة (numpy) ومكتبة (matplotlib) مباشرة ولأختبار ذلك نجرب رسم دالة بسيطة ولتكن رسم دالة القيم من الصفر الى الخمسة وكما في الصورة التالية:

Jupyter data_science_1 (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

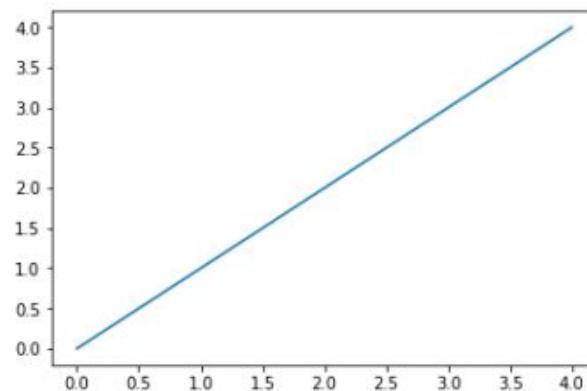
Code

```
In [1]: %pylab inline
```

Populating the interactive namespace from numpy and matplotlib

```
In [2]: plot(arange(5))
```

```
Out[2]: [<matplotlib.lines.Line2D at 0x7f503c0a9ef0>]
```



الان لأننا رأينا نتيجة الرسم فهذا معناه نجاح استيراد دوال المكتبات سابقة الذكر.

استيراد المكتبات المهمة للعمل عليها:

المكتبات الرئيسية التي نحتاج استيرادها الى بيئة العمل خاصتنا هي (numpy, matplotlib, pandas) ورغم اننا قمنا باستيراد اول مكتبتين منهما بالايجاز الاول اعلاه الا اننا سنقوم بأدراج كودات الاستيراد لهما ايضاً في حالة كنتم تعملون في بيئة اخرى غير الاناكوندا:

```
In [3]: import pandas as pd      # استيراد مكتبة Pandas
import numpy as np             # استيراد مكتبة numpy
import matplotlib as plt      # استيراد مكتبة matplotlib

df = pd.read_csv("/home/mustafa/Downloads/train.csv") # قراءة بيانات التدريب
```

الآن نشرح الكود اعلاه قليلاً:

السطر الأول كما هو مبين في التعليق لأستيراد مكتبة (pandas) وإسناد اسم (pd) لها.

ونفس الشيء بالنسبة للسطر الثاني والثالث.

وأما بخصوص السطر الرابع فهو قراءة لملف البيانات الخاصة بالتدريب (train.csv) الذي قمنا بتنزيله سابقاً بأستخدام دالة القراءة الموجودة ضمن مكتبة (pandas) وهي دالة (read_csv) ولاحظ أننا قمنا بوضع اسم المكتبة الذي إسندناه لها (pd) ثم نقطة قبل اسم الدالة لأستدعائها وهذه هي الطريقة العامة لأستدعاء دوال هذه المكتبات من الآن وصاعداً.

وأخيراً قمنا بوضع امتداد خزن الملف (train.csv) بداخل علامات اقتباس مزدوجة بداخل قوسي الدالة وإسناد الناتج (خزن الملف) بداخل متغير اسمه (df).

الآن لإلقاء نظرة سريعة على البيانات في هذا الملف يمكننا عرض الحقول الأولى من الملف بأستخدام دالة (head) وتحديد عدد الأسطر التي نريد استعراضها وكما في ادناه:

```
In [4]: df.head(7)
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	L
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	

هنا قمنا بطباعة أول 7 اسطر من البيانات وكما نرى هناك مجموعة من الحقول بحيث أن كل حقل له نوعية بيانات (data type) مختلفة وهذه هي الفائدة من استعراض البيانات.

لأخذ نظرة عامة عن الحقول الرقمية يمكن استخدام دالة الوصف (describe) وكما في ادناه:

```
In [5]: df.describe()
```

```
Out[5]:
```

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.000000	564.000000
mean	5403.459283	1621.245798	146.412162	342.000000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.000000	0.000000
25%	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	3812.500000	1188.500000	128.000000	360.000000	1.000000
75%	5795.000000	2297.250000	168.000000	360.000000	1.000000
max	81000.000000	41667.000000	700.000000	480.000000	1.000000

ونلاحظ أنه قام بعرض الحقول الرقمية ومجموعها والمعدل والانحراف المعياري وبقية الحسابات الرياضية الأخرى للحقول الرقمية فقط.

والان ومن خلال هذه المعطيات نستطيع استنتاج ان بعض الحقول تحتوي بيانات ناقصة (مثلاً loanAmount عددها اقل من ال ApplicantIncome) وهكذا.

كذلك نستطيع تحديد ان أغلب المتقدمين للحصول على قروض (في مثالنا هذا) لديهم سجل وتاريخ رصيد لأن معدل تاريخ الرصيد (Credit_History) هو 84% وهو معدل مرتفع.

وكذلك من خلال المقاييس الاحصائية لدخل المتقدمين (ApplicantIncome) والتوقع للسداد (Expectation) نرى أن كلاهما لهما نفس التوجه (orientation).

وبصورة عامة نستطيع من خلال استعراض البيانات اخذ نظرة اولية عن معناها والعلاقة بينها وهل هناك توجه معين (trend) يعطينا اشارة معينة لوجود علاقة بين زيادة مقدار ما ونقصان مقدار اخر او اي نوع من انواع العلاقات الاخرى وفي هذه الخطوة من الضروري معرفة التطبيق او المشكلة التي نتناولها والنظر لها من وجهة نظر مختص (ولا بأس بالاستعانة بمختص في هذا المجال للمساعدة في فهم الجداول) لأن ذلك سيحدد الخطوات اللاحقة في التحليل.

اما بالنسبة للقيم الغير رقمية فيمكننا ايضاً عرض نسبة تكرار كل منها لنعرف ايضاً هل هي منطقية ومعقولة وهل هناك علاقة بينها او لا ومن خلال الابعاز التالي:

```
In [6]: df['Property_Area'].value_counts()
```

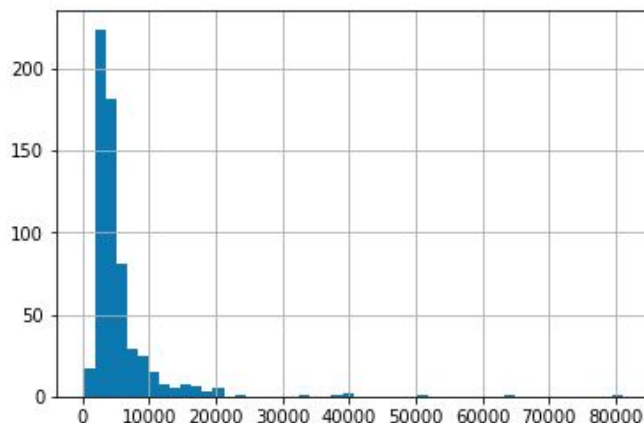
```
Out[6]: Semiurban    233
        Urban        202
        Rural        179
        Name: Property_Area, dtype: int64
```

تحليل التوزيع:

الآن وبعد معرفة الخصائص الأساسية للبيانات نستطيع دراسة توزيعها و للمتغيرات (الاعمدة) المختلفة ولنبدأ بالمتغيرات الرقمية ولنأخذ مثلاً عليها وهو معدل دخل المتقدمين للحصول على قرض (ApplicantIncome) ونقوم برسم الهستوغرام او الرسم البياني (Histogram) وبأستخدام الابعاز التالي:

```
In [7]: df['ApplicantIncome'].hist(bins=50)
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5029ab7400>
```



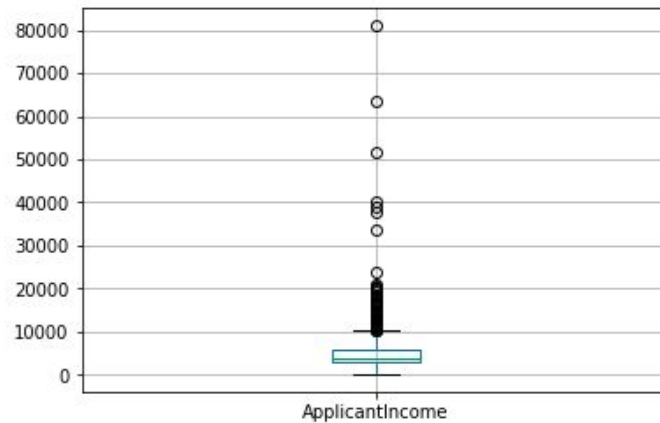
هنا ايضاً رأينا ان دالة (hist) التي تستخدم لرسم المخطط البياني للمتغير ApplicantIncome استخدمت مع معامل اخر وهو عدد ال (bins) او عدد النقاط في الاحداثي السيني لتمثيل هذا المتغير بشكل دقيق ونلاحظ ان اغلب المتقدمين يقعون في المساحة بين الصفر والعشرين ألف

دولار بالسنة (والذي يعتبر دخل منخفض او دون مستوى الفقر في الولايات المتحدة الامريكية كمثال) وهو شيء منطقي ولكن مع ذلك نجد هناك عدة قفزات عند الاربعين الف والخمسين ألف وخمسة وستون الف وهكذا.

اما اذا اردنا استخدام الرسم بطريقة (box plot) فيمكننا عمل ذلك كما يلي:

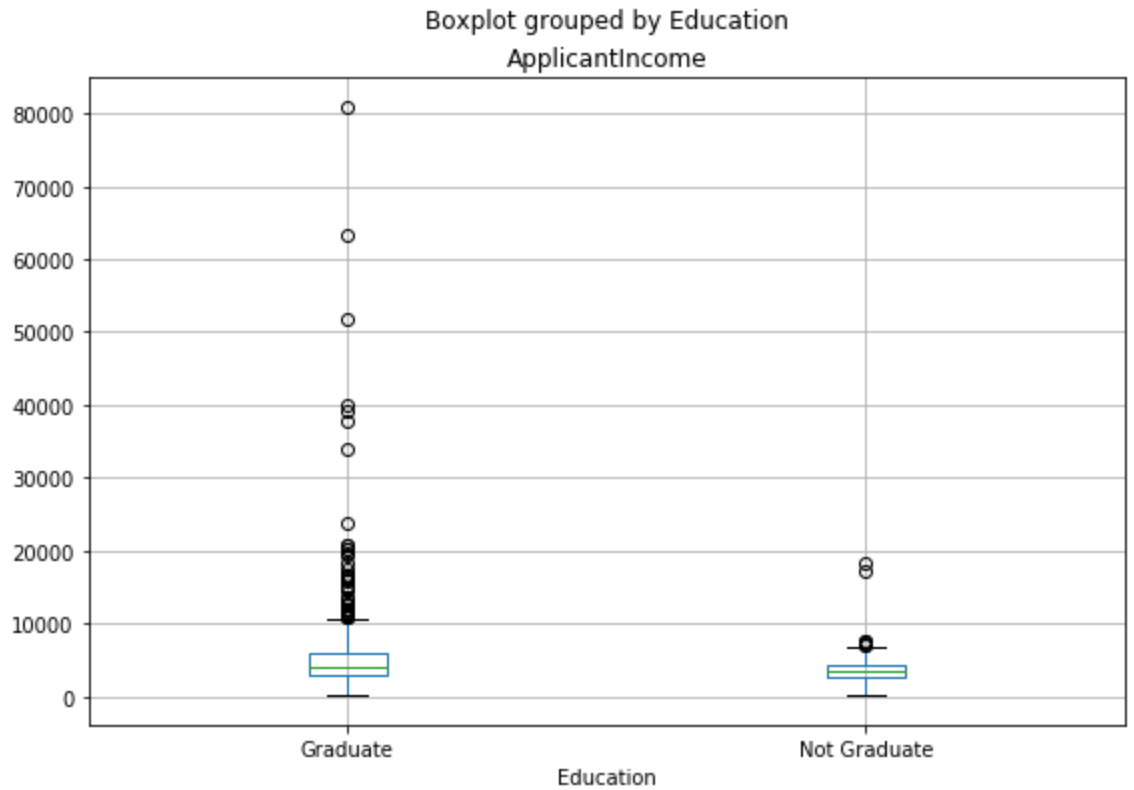
```
In [8]: df.boxplot(column='ApplicantIncome')
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f5029a7f080>
```



وهنا نلاحظ أننا استطعنا رؤية التوزيع للبيانات ضمن هذا المتغير (ApplicantIncome) بطريقة أخرى والتي تبين وجود حالات شاذة (outliers) بعيدة عن أغلبية البيانات وهي الدوائر الصغيرة المنفردة في الأعلى.

والآن لنحاول الربط بين هذا التوزيع وبين المستوى العلمي للمتقدمين ونقوم برسم دخل المتقدمين مع الحالة (متخرج graduate او غير متخرج Not graduated) وكما في الايعاز التالي:

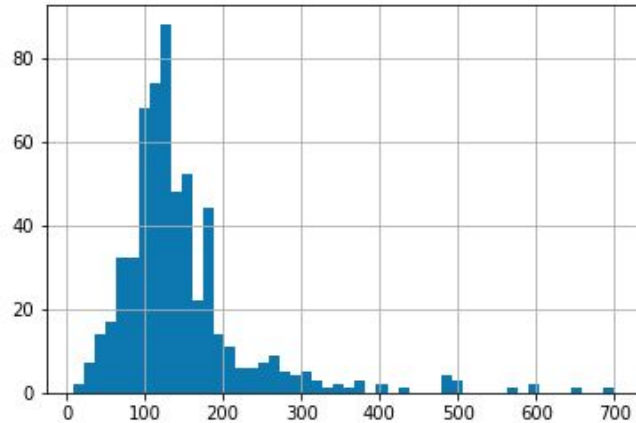


من الرسم اعلاه نستطيع استنتاج أن معدل دخل الخريجين وغير الخريجين متقارب ولكن هناك الكثير من الخريجين بمعدلات دخل اعلى بكثير من أقرانهم الغير خريجين وهذه نقطة يجب ان نبقها في بالنا لأستخدامها في خطوات التحليل المقبلة.

والان لنرسم مقدار القرض المطلوب بالطريقتين أعلاه وكما يلي:

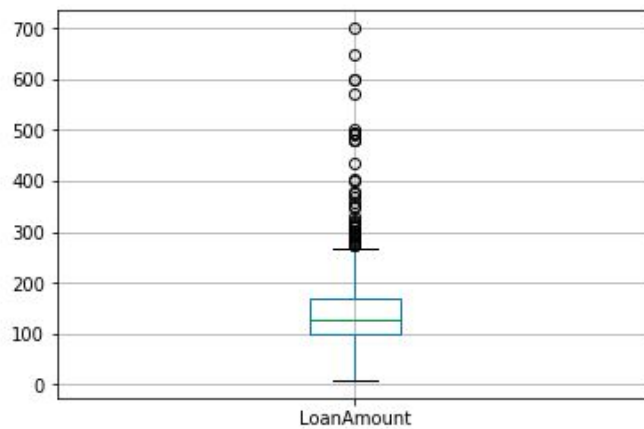
```
In [10]: df['LoanAmount'].hist(bins=50)
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50299077b8>
```



```
In [11]: df.boxplot(column='LoanAmount')
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50298207f0>
```



وكما نرى ايضاً هناك قيم شاذة في كلا الرسمين السابقين مما يعني اننا بحاجة إلى تحليل أعمق للبيانات وهو ما سنقوم به في الخطوة الثانية من خطوات التحليل (في الدروس القادمة ان شاء الله).

تحليل المتغيرات الغير رقمية Categorical Variable Analysis

ويمكن ذلك باستخدام عدة ادوات وكما مبين في الصورة التالية:

```
In [14]: temp1 = df['Credit_History'].value_counts(ascending=True)
temp2 = df.pivot_table(values='Loan_Status', index=['Credit_History'], aggfunc=lambda x: x.map({'Y':1, 'N':0}).mean())
print ('Frequency Table for Credit History:')
print (temp1)

print ('\nProbability of getting loan for each Credit History class:')
print (temp2)
```

Frequency Table for Credit History:

0.0	89
1.0	475

Name: Credit_History, dtype: int64

Probability of getting loan for each Credit History class:

Credit_History	Loan_Status
0.0	0.078652
1.0	0.795789

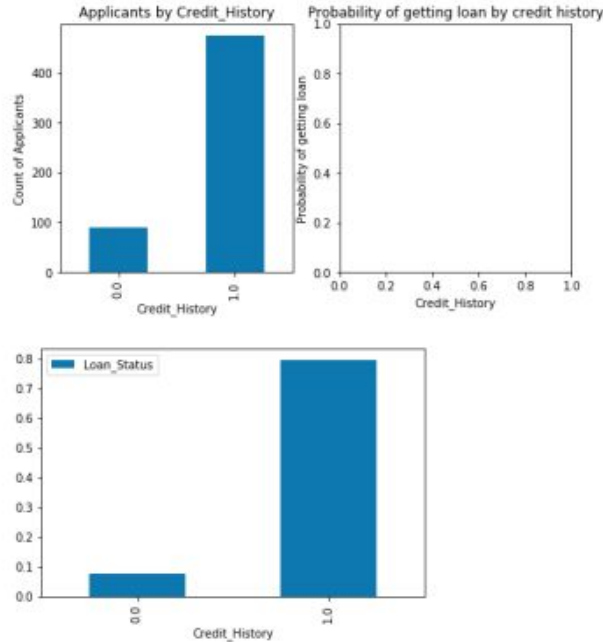
حيث قمنا بخزن قيمة تاريخ الرصيد للمتقدمين بطلب اقتراض في متغير اسمه (temp1) وخزن حالة القرض وهل تم ارجاعه او لا في جدول في المتغير (temp2) ثم قمنا بطباعتها وكما في الصورة اعلاه.

ولعمل نفس الشيء ولكن بطريقة رسومية باستخدام المكتبة (matplotlib) يمكننا استخدام الابعازات التالية:


```
In [15]: import matplotlib.pyplot as plt
fig = plt.figure(figsize=(8,4))
ax1 = fig.add_subplot(121)
ax1.set_xlabel('Credit_History')
ax1.set_ylabel('Count of Applicants')
ax1.set_title("Applicants by Credit_History")
temp1.plot(kind='bar')

ax2 = fig.add_subplot(122)
temp2.plot(kind='bar')
ax2.set_xlabel('Credit_History')
ax2.set_ylabel('Probability of getting loan')
ax2.set_title("Probability of getting loan by credit history")

Out[15]: Text(0.5,1,'Probability of getting loan by credit history')
```



وكما هو واضح فأغلب الايعازات مشابهة الى حد كبير لتلك المستخدمة في برنامج الماتلاب لرسم البيانات واطافة اسماء للمحور السيني (x-axis) والصادي (y-axis) وكذلك اسم لمحتويات الرسم وعنوان للرسم وتقسيم الرسومات الى عدة رسومات فرعية (sub plots).

ومن الرسم اعلاه يتضح ان فرص استرجاع القرض أكثر بثمان مرات للمتقدمين ممن لديهم تاريخ رصيد (Credit_History) اكثر من البقية ويمكن رسم رسومات مشابهة للحالة الزوجية والحالة الوظيفية وغيرها ومقارنتها بحالة استرجاع القرض او لا للبيانات المتوفرة لمحاولة رسم استنتاجات عن العلاقة بين هذه الحالات.

وهناك طريقة أخرى لجمع الرسمين في شكل واحد وكما في ادناه:

```
In [16]: temp3 = pd.crosstab(df['Credit_History'], df['Loan_Status'])
temp3.plot(kind='bar', stacked=True, color=['red', 'blue'], grid=False)
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f50296ea550>
```



حيث نرى توافق حالة القرض مع تاريخ الرصيد الى حد كبير.

الى هنا رأينا كيف يمكننا الاستفادة من طرق عرض ورسم البيانات المختلفة للخروج بأستنتاجات مبكرة عن العلاقة بين المتغيرات المختلفة والنتيجة التي نبحث عنها للأجابة عن السؤال الجوهري: هل تقوم الشركة بمنح المتقدم للحصول على قرض القرض المطلوب او لا اذا اخذنا بنظر الاعتبار تاريخه السابقة وحالته الوظيفية والزوجية والعلمية وغيرها من المعلومات الاخرى التي يتم جمعها منه عند التقديم لطلب القرض.

الى هذه النقطة اعتقد ان الامور اصبحت واضحة بخصوص المشروع الذي نعمل عليه وهو بأختصار كالاتي:

انت مختص لتحليل البيانات في شركتك (او المصرف التجاري الذي تعمل فيه) ووظيفتك ان تقوم بجمع البيانات السابقة للعملاء الحاليين وتقارن بينها وبين النتيجة النهائية للقروض وهل قاموا بتسديدها او لا وتحاول إيجاد علاقة بين عدم التسديد واي من المتغيرات الاخرى فمثلاً إذا استنتجت (في نهاية المشروع) أن كل من يقل معدل دخلهم عن 10000 دولار سنوياً لن يقوموا بتسديد القرض في وقته (فرضاً) فهذا معناه ان النموذج الذي يجب ان تبنيه يجب ان يعطي نتيجة (لا) لكل متقدم مستقبلي ممن يقل دخلهم عن 10000 دولار سنوياً وهذا هو المفهوم الاساسية

لعلم البيانات او تعليم الماكنة او تمييز الانماط او المسمى الاكبر والذي يحتويها كلها وهو (الذكاء الاصطناعي).

وكما ذكرنا في الدرس السابق فمصدر هذه الدروس (مع الترجمة والتبسيط والشرح) هو الرابط التالي ([انقر هنا لزيارة المصدر الاصلي](#))

انتظرونا في الدروس القادمة حيث سنبدأ بتحليل البيانات وتنظيفها اكثر لتهيئتها للخطوة الاخيرة وهي تطبيق خوارزميات التصنيف والعزل.

بايثون-30: تحليل البيانات بلغة بايثون المشروع الاول-3

السلام عليكم ورحمة الله وبركاته

اخوتي الكرام اخواتي الفاضلات

وصلنا الى المرحلة الثانية من مراحل تحليل البيانات وهي مرحلة معالجة البيانات (تنظيف وترتيب وتقليل للحجم وغيرها) لتكون جاهزة لتطبيق احد خوارزميات التحليل والتصنيف عليها في الخطوة القادمة فتابعوا معنا:

أثناء استكشاف البيانات وجدنا فيها مجموعة من المشاكل التي يجب حلها قبل ان تكون البيانات جاهزة للتحليل النهائي ويتم حل هذه المشاكل ضمن هذه المرحلة الثانية المسماة (data munging) وفي ادناه بعض المشاكل التي اكتشفناها حتى الآن:

1- هنا بعض القيم المفقودة لبعض المتغيرات (والتي اشرنا لها في الدرس السابق) وهنا يجب علينا ملء كل الجداول بقيم مناسبة تعتمد على كمية البيانات المفقودة وأهميتها للنتيجة النهائية.

2- لاحظنا أن قيم (دخل المتقدمين للحصول على القروض) و(مقدار القرض المطلوب) فيها قيم شاذة عن المعدل وهنا يستوجب مراعاة ذلك لأنه يعني وجود حالات تخالف القاعدة (القيم القريبة من المعدل) فيجب معالجتها قبل إكمال التحليل.

وبالإضافة الى هذه المشاكل في الحقول الرقمية فنحن يجب ان نهتم بالقيم في المتغيرات الغير رقمية مثل الحالة الزوجية والجنس ومساحة البناء ومستوى التعليم وغيرها لمحاولة الحصول منها على معلومات مفيدة.

الخطوة الاول: تدقيق القيم المفقودة في المتغيرات المختلفة:

اغلب خوارزميات التحليل لا تتعامل مع جداول ناقصة ومعلومات غير كاملة وحتى لو كانت كذلك فإن النتائج ستكون غير منطقية ولذلك يجب اولاً ايجاد القيم المفقودة ومحاولة ملئها بطريقة ما. اولاً سنبحث عن الحقول الفارغة التي تحتوي (null) او (NaN) باستخدام مكتبة (pandas) وكما في المثال أدناه:

```
In [18]: df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Out[18]: Loan_ID          0
Gender             13
Married            3
Dependents         15
Education          0
Self_Employed     32
ApplicantIncome   0
CoapplicantIncome 0
LoanAmount        22
Loan_Amount_Term   14
Credit_History    50
Property_Area     0
Loan_Status       0
dtype: int64
```

حيث يقوم هذا الايجاز بأرجاع عدد القيم الفارغة (isnull) في كل عمود من الجدول المخزون اصلاً في المتغير المسمى (df) وكما نرى بعض الاعمدة فيها عدد القيم المفقودة صفر أي أنها مكتملة وبعضها فيها قيم مفقودة تتراوح بين 3 و 50.

هناك الكثير من الاسباب لفقدان البيانات ومنها ان أغلب هذه الجداول يتم جمعها من قبل حواسيب تقوم باستخراج البيانات من مواقع انترنت او معلومات مستخدمين ونحن نعرف ان اغلب المستخدمين يميلون الى ملئ الحقول الالزامية فقط حين يقومون بملئ حقول اي جدول او التسجيل في اي موقع واما عن كيفية التعامل مع هذه البيانات المفقودة فهناك عدة طرق مشروحة بالتفصيل في الرابط التالي:

[انقر هنا لزيارة درس استكشاف البيانات والتعامل مع المفقودات بالتفصيل](#)

ويمكن تلخيص هذه الطرق بما يلي:

1- حذف السطر كاملاً ان كان احد متغيراته مفقوداً؛ وهذه الطريقة مفيدة لبساطتها ولكنها قد تجعلنا نفقد الكثير من المعلومات الضرورية التي تؤثر على الناتج النهائي.

2- ملء المكانات الفارغة بقيمة تمثل معدل القيم الاخرى في العمود وبالتالي نحرص على أن هذه القيمة المضافة لن تكون شاذة عن المعدل العام للقيم في ذلك المتغير.

3- ملء القيم المفقودة بقيم اخرى غير المعدل مثل الوسيط او المنوال وبحسب توزيع البيانات (هنا نحتاج بعض الاساسيات في العمليات الاحصائية على البيانات) فعندما نقوم برسم البيانات المتوفرة في عمود ما نستطيع معرفة هل ان اغلب البيانات متوزعة حول المعدل (mean) او الوسيط (median) او المنوال او غيرها فنختار القيمة الأكثر احتمالاً ونملأ بها الحقول الفارغة.

4- يمكن ايضاً الاستعانة بالاعمدة البقية لملء المعلومات المفقودة من الأعمدة الاخرى فمثلاً اذا كان الحقل الفارغ خاص بشخص ونحن نعرف طوله ولكننا لا نعرف جنسه فمن خلال أخذ معدل الطول للذكور والإناث ثم مقارنة الطول لدينا مع المعدلات المتوفرة نستطيع ان "نخمن" هل هذا الحقل المفقود هو لذكر (male) أو لأنثى (female) وهكذا. كما قلنا التفاصيل كاملة موجودة في الرابط اعلاه.

والان نرجع الى مثالنا ومشروعنا واملء الحقول الفارغة من مقدار القرض المطلوب يمكننا ببساطة ملئها بمعدل القروض البقية ويتم ذلك بلغة بايثون وباستخدام مكتبة (pandas) بأيعاز واحد وكما يلي:

```
In [19]: df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
```

```
In [20]: df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Out[20]: Loan_ID          0
Gender          13
Married         3
Dependents      15
Education       0
Self_Employed  32
ApplicantIncome 0
CoapplicantIncome 0
LoanAmount      0
Loan_Amount_Term 14
Credit_History 50
Property_Area   0
Loan_Status     0
dtype: int64
```

وهنا نلاحظ ان عدد الحقول المفقودة في ال (loanAmount) كان 22 قبل تطبيق اليعاز السابق وبعد تطبيقه اصبح صفر أي أن كل الحقول في هذا العمود تم ملئها بنجاح بقيمة المعدل للحقول البقية.

طبعاً هناك طرق اخرى اكثر صعوبة وهي ببناء نموذج تحليل متكامل (مثل الذي نريد عمله في النهاية) فقط لتخمين قيم كل حقل مفقود ولكن هذه العملية تأخذ وقت طويل (رغم انها تعطي نتائج افضل في اغلب الاحيان) ولكن يبقى الخيار هنا لمحلل البيانات وهل يحتاج كل هذه الدقة في التحليل وملء الفراغات او لا وطبعاً يعتمد القرار على اهمية البيانات ونتائجها.

والان لملء الفراغات في عمود (self employed) أي هل يعمل المتقدم لطلب القرض عمل خاص (لحسابه الخاص) أو لا فهنا نقوم بعرض كم واحد نعم وكم واحد لا ثم نقرر وكما في اليعاز التالي:

```
In [22]: df['Self_Employed'].value_counts()
```

```
Out[22]: No      500
Yes       82
Name: Self_Employed, dtype: int64
```

وبما ان اغلب المتقدمين لطلب القروض هم ليسوا موظفين ذاتياً (No=500) فمن الطبيعي ان نقوم بملء الحقول الفارغة ب(No) لأن تكرارها اكثر (طبعاً العملية كلها احتمالية ولا شيء

مضمون وهنا يبرز ذكاء محلل البيانات لاختيار الأكثر احتمالية ومنطقية ولذلك يفترض ان يكون محلل البيانات ملماً بالكثير من العلوم الاخرى مثل الاحصاء والاحتمالية والتحليل المنطقي وغيرها).

نقوم بملء الحقل بقيمة (No) بأستخدام الايغاز التالي:

```
df['Self_Employed'].fillna('No',inplace=True)
```

و لتجنب تكرار هذه العملية لكل عمود يدوياً نقوم بأنشاء دالة تقوم بحساب الوسيط (median) للحقول الأخرى وملء الفراغات بها وكما في الكود التالي:

```
In [29]: table = df.pivot_table(values='LoanAmount', index='Self_Employed', columns='Education', aggfunc=np.median)
# Define function to return value of this pivot_table
def fage(x):
    return table.loc[x['Self_Employed'],x['Education']]

In [35]: # Replace missing values
df['LoanAmount'].fillna(df[df['LoanAmount'].isnull()].apply(fage, axis=1), inplace=True)
```

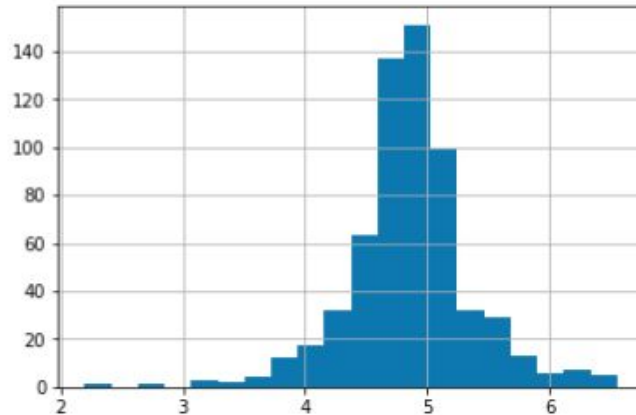
وبهذا يفترض ان تكون كل الحقول قد تم ملئها بالوسيط (median) لكل منها.

كيف نقوم بمعالجة القيم المتطرفة (extreme values)؟

من المهم كما قلنا سابقاً فهم القيم المحتملة والمنطقية لكل متغير وبالتالي سنعرف هل ان بعض القيم شاذة وغير منطقية أو انها منطقية ولكنها بعيدة عن المعدل (طفرات غير محسوبة) ولناخذ مثالاً على ذلك وهو مقدار القرض المطلوب والذي يمكننا رسمه بطريقة الأعمدة (bar chart) بأستخدام الايغاز التالي:


```
In [36]: df['LoanAmount_log'] = np.log(df['LoanAmount'])
df['LoanAmount_log'].hist(bins=20)
```

```
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x7f912c263f98>
```

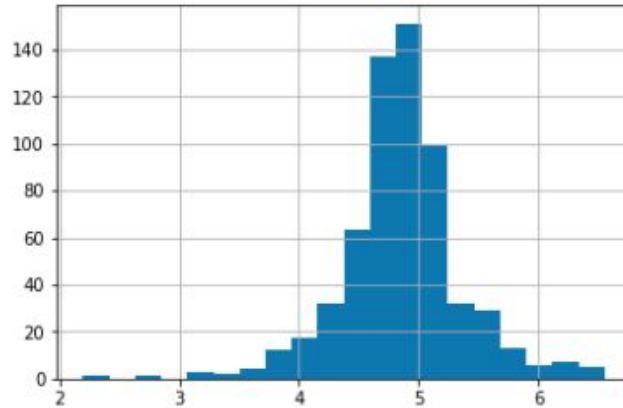


هنا نرى أن كل القيم (او اغلبها) تدور حول المعدل وذلك لأننا أخذنا التحويل اللوغاريتمي (log transformation) لها وهي أحد الطرق لمعالجة القيم المتطرفة.

والان للتخلص من القيم المتطرفة في عمود معدل دخل المتقدمين للحصول على القرض (ApplicantIncome) نقوم ايضاً بحساب التحويل اللوغاريتمي لها وكما في الكود التالي:

```
In [37]: df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df['TotalIncome_log'] = np.log(df['TotalIncome'])
df['LoanAmount_log'].hist(bins=20)
```

```
Out[37]: <matplotlib.axes._subplots.AxesSubplot at 0x7f912c14b208>
```



وفي ادناه شرح الكود السابق:

السطر الاول هو انشاء عمود جديد في الجدول الذي قمنا بخزنه في البداية في متغير اسمه (df) واسم هذا العمود هو الدخل الكلي (TotalIncome) والذي قمنا فيه بجمع دخل المتقدم (ApplicantIncome) ودخل المتقدم الزميل او شريك السكن (CoapplicantIncome).

السطر الثاني: قمنا فقط بإنشاء عمود جديد ووضعنا فيه ال (log) الخاص بالعمود السابق. وكما هو واضح استخدمنا دالة ال (log) من مكتبة (NumPy) بكتابة (np.log()).

السطر الثالث: رسم المخطط البياني للعمود الجديد ويتضح فيه ايضاً التخلص من كل (او اغلب) القيم المتطرفة.

هذه كانت بعض الخطوات لتنظيف البيانات ومعالجتها وملء الفراغات والنقص في الجداول وهناك الكثير من الاجراءات الاخرى والتي يستطيع محلل البيانات القيام بها بناءً على نوعية المشروع والبيانات التي يعمل عليها وكذلك كما ذكرنا الدقة المتوخاة واهمية النتائج.

بايثون-31: تحليل البيانات بلغة بايثون المشروع الاول-4

السلام عليكم ورحمة الله وبركاته

وصلنا معكم الى المرحلة الاخيرة من مراحل تحليل البيانات والتي تتمثل في بناء نموذج التنبؤ (building a predictive model) بلغة بايثون بعد ان قمنا بتهيئة البيانات في الدروس السابقة.

المكتبة الأكثر استخداماً في بناء نماذج التخمين والتنبؤ هي مكتبة (skicit-learn) او ما تسمى (sklearn) والتي يمكن الاطلاع على تفاصيلها من الرابط التالي:

[انقر هنا لزيارة رابط شرح مكتبة \(sklearn\)](#)

قبل البدء بتحليل البيانات يجب ان نتأكد ان الجدول المستخدم للتحليل لا يحتوي حقول فارغة ويتم ذلك باستخدام الايعاز التالي:

```
In [43]: df.apply(lambda x: sum(x.isnull()),axis=0)
```

```
Out[43]: Loan_ID          0
         Gender          0
         Married         0
         Dependents      0
         Education       0
         Self_Employed   0
         ApplicantIncome  0
         CoapplicantIncome 0
         LoanAmount      0
         Loan_Amount_Term 0
         Credit_History  0
         Property_Area   0
         Loan_Status     0
         LoanAmount_log  0
         TotalIncome     0
         TotalIncome_log 0
         dtype: int64
```

وكما نرى هنا فإن كل الحقول مليئة لأن الايعاز اعلاه يعرض عدد الحقول الفارغة لكل متغير وهي في هذه الحالة كلها اصفار (حيث قمنا بملئها في الدرس السابق بقيم تتناسب مع نوعية متغيرات وميول (trends) كل عمود).

من الأمور المهمة التي يجب ملاحظتها ان هذه المكتبة تتعامل مع الجداول الرقمية فقط وبالتالي يجب تحويل أي بيانات غير رقمية (categorical data) الى بيانات رقمية وذلك باستخدام الكود التالي:

```
In [68]: from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['Gender']=le.fit_transform(df['Gender'])
df['Married']=le.fit_transform(df['Married'])
df['Education']=le.fit_transform(df['Education'])
df['Self_Employed']=le.fit_transform(df['Self_Employed'])
df['Property_Area']=le.fit_transform(df['Property_Area'])
df['Loan_Status']=le.fit_transform(df['Loan_Status'])

df.dtypes
```

```
Out[68]: Loan_ID          object
Gender              int64
Married             int64
Dependents          object
Education           int64
Self_Employed      int64
ApplicantIncome    int64
CoapplicantIncome  float64
LoanAmount          float64
Loan_Amount_Term   float64
Credit_History     float64
Property_Area      int64
Loan_Status         int64
LoanAmount_log     float64
TotalIncome        float64
TotalIncome_log    float64
dtype: object
```

نلاحظ هنا ان كل الحقول التي قمنا بمعالجتها قد تحول نوعها البياني الى قيم رقمية (int64, float64) ما عدا المتغير (Dependants) حيث كانت فيه مشكله وهي انه يحتوي على بعض القيم التي ليست رموز ولا ارقام بل خليط بينهما وهي القيم (3+) حيث قمنا بتركه كما هو ولم نقم بأدخاله في خطوات التحليل اللاحقة علماً انه يمكن معالجة هذه المشكلة بطرق كثيرة ولكننا ارتأينا ان نبقي الامور بأبسط صورة ونترك معالجة هذه الحالات للمشاريع القادمة ان شاء الله.

الآن نقوم باستيراد المكتبات الخاصة بالتحليل والتصنيف والتي تقوم بقبول الجدول الخاص ببيانات التدريب (training data) وتستخرج لنا نموذج التخمين مع حساب الدقة والتدقيق المتقاطع (cross validation).

ملاحظة: في هذا المشروع سنقوم باستيراد وشرح امثلة بسيطة عن خوارزميات التصنيف وتعليم الماكينة (Classification and machine learning algorithms) ولمعرفة تفاصيل اكثر عن بقية الاصناف يمكن زيارة الرابط التالي: ([انقر هنا](#)).

الآن نقوم بإنشاء دالة التصنيف (وهي ثابتة لأغلب المشاريع) والذي يتغير باستمرار هو فقط خوارزمية التصنيف ومعاملاتها (parameters) وكما سنرى بعد قليل.

```
In [74]: #Import models from scikit learn module: استيراد مكتبات التصنيف
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold #For K-fold cross validation للتدقيق المتقاطع
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics

#Generic function for making a classification model and accessing performance: دالة التصنيف وتقييم الاداء
def classification_model(model, data, predictors, outcome):
    #Fit the model: انتاج نموذج التخمين
    model.fit(data[predictors],data[outcome])

    #Make predictions on training set: تطبيق نموذج التخمين على بيانات التدريب
    predictions = model.predict(data[predictors])

    #Print accuracy طباعة الدقة الخاصة بنموذج التخمين
    accuracy = metrics.accuracy_score(predictions,data[outcome])
    print ('Accuracy : %s' % "{0:.3%}".format(accuracy))

    #Perform k-fold cross-validation with 5 folds تطبيق التدقيق المتقاطع بعد تقسيم البيانات الى 5 اجزاء
    kf = KFold(data.shape[0], n_folds=5)
    error = []
    for train, test in kf:
        # Filter training data ترميز بيانات التدريب
        train_predictors = (data[predictors].iloc[train,:])

        # The target we're using to train the algorithm. نموذج التدريب الهدي
        train_target = data[outcome].iloc[train]

        # Training the algorithm using the predictors and target. تدريب الخوارزمية باستخدام نموذج التخمين
        model.fit(train_predictors, train_target)

        #Record error from each cross-validation run تسجيل التخمينات الخاطئة
        error.append(model.score(data[predictors].iloc[test,:], data[outcome].iloc[test]))

    print ("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean(error)))

    #Fit the model again so that it can be refered outside the function: يسط النموذج مرة اخرى لنتم الوصول له من اي مكان
    model.fit(data[predictors],data[outcome])
```

الكود أعلاه يمثل (function) عامة مهمتها القيام بتطبيق خوارزميات التصنيف على بيانات التدريب (training data) ثم انتاج نموذج التخمين لتطبيقه في المستقبل على بيانات الاختبار (testing data).

والآن لنبدأ بأول خوارزمية تصنيف:

تحليل الانحدار اللوجستي (Logistic Regression):

الآن سنبدأ بإنشاء أول نموذج تحليل انحدار لوجستي:

أحد الطرق لعمل ذلك هو ادخال كل المتغيرات الى النموذج ولكن هذا قد ينتج عنه ما يسمى (over fitting) أي التطابق المفرط وهو حالة تحصل حين نقوم باستخدام بيانات تدريب أكثر من اللازم وأكثر من المطلوب بحيث تعطينا نتائج تخمين عالية جداً لبيانات التدريب ولكنها واطئة جداً لبيانات الاختبار والسبب ان النموذج الذي قمنا بإنشائه كان مطابقاً بشكل مبالغ فيه لبيانات التدريب والتي قد تتغير بمرور الوقت حين نريد تطبيق النموذج على بيانات اخرى (عند الاختبار او العمل الحقيقي).

وبدلاً من إدخال كل البيانات الى النموذج نستطيع القيام بعدة تخمينات مدروسة لمعرفة اي بيانات أكثر تأثيراً في تطبيقنا:

نحن لدينا جدول يبين بيانات العملاء الذين قاموا بأخذ قروض من مصرف او شركة ما ونعرف (في بيانات التدريب) كم منهم قام بإرجاع القرض بدون مشاكل وكم منهم لم يقم بذلك ولأننا نريد أن نجد علاقة (نموذج) يربط بين معلومات المقترضين وبين احتمالية إرجاع القروض في المستقبل فمن الطبيعي أن نربط بين:

- فرصة إرجاع القرض تكون أكبر بالنسبة للمتقدمين الذين لديهم تاريخ رصيد (credit history) وهذا ما لاحظناه في تحليلنا السابق.
- احتمالية إرجاع القرض تكون أكبر أيضاً للمقترضين الذين يمتلكون شخص اخر معهم في القرض (co-applicant) ممن يعيشون معهم ولديهم دخل ثابت أيضاً.
- احتمالية إرجاع القرض تزيد أيضاً كلما كانت نسبة التعليم اعلى (تم استنتاج ذلك في الدروس السابقة).

- طلب الاقتراض من قبل سكان المدن التي تشهد نمو اقتصادي اضمن من المقترضين في الأماكن الأقل نمواً اقتصادياً (الريف مثلاً).

والان نقوم باستدعاء الدالة أعلاه لخوارزمية التحليل اللوجستي كما في الكود أدناه:

```
In [75]: outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History']
classification_model(model, df, predictor_var, outcome_var)

Accuracy : 80.945%
Cross-Validation Score : 80.946%
```

حيث قمنا بأدخال متغير واحد فقط لبناء النموذج وهو حالة تاريخ الرصيد (credit history) و أما النتيجة التي نريد الوصول لها (Loan_Status) فهي المتغير الذي يمثل حالة القرض (تم إرجاعها أو لا) والتي تسمى في حالتنا هذه (Loan_Status).

ونرى أن الدقة (accuracy) اكثر من 80% بقليل والتي تمت بتقنية التدقيق المتقاطع (corss validation) فما هي هذه التقنية؟

Cross Validation التدقيق المتقاطع

النموذج بشكل عام يحتاج أن يرسم استنتاجات من بيانات معينة (training) ويطبقها على بيانات أخرى (testing) ثم يختبر مقدار صحة النتائج وهنا لأننا قمنا بأدخال بيانات التدريب فقط (training) فتقنية التدقيق المتقاطع تقوم بتجزئة البيانات الخاصة بالتدريب الى عدة اجزاء متساوية (تجزئة عشوائية) وهنا في حالتنا اخبرنا النموذج بتجزئة البيانات الى 5 مجاميع متساوية (من الكود الخاص بالدالة العامة اعلاه). والذي حصل هنا ان النموذج قام بتجزئة البيانات الى 5 اجزاء متساوية ثم اخذ واحدة منها وبنى عليه استنتاجه معتبراً اياها هي بيانات التدريب (training data) ثم قام بتطبيق استنتاجه على المجاميع الاربعة المتبقية بصفتها بيانات اختبار (testing)

(data) ثم قام بأختبار دقة الاستنتاج وهكذا اعاد العملية 5 مرات وفي كل مرة يأخذ احد الاخماس المختلفة ويعيد العملية.

يمكننا اعادة العملية لبيانات اخرى (من ضمن جدول بيانات التدريب) وكما في المثال التالي:

```
In [76]: # يمكننا تجربة تراكيب مختلفة من البيانات ورؤية الفرق في النتائج كل مرة :
predictor_var = ['Credit_History', 'Education', 'Married', 'Self_Employed', 'Property_Area']
classification_model(model, df, predictor_var, outcome_var)

Accuracy : 80.945%
Cross-Validation Score : 80.946%
```

ونرى ان الدقة نفسها هنا (وهي صدفة فقد لا تكون الدقة نفسها لأمثلة اخرى).

نموذج شجرة القرار (Decision Tree Model)

وهي طريقة تحليل وتصنيف اخرى توفر عادة دقة أكبر من الطريقة السابقة ويمكن استدعائها وتطبيقها كما في الامثلة التالية:

```
In [77]: model = DecisionTreeClassifier()
predictor_var = ['Credit_History', 'Gender', 'Married', 'Education']
classification_model(model, df, predictor_var, outcome_var)

Accuracy : 80.945%
Cross-Validation Score : 80.946%
```

```
In [78]: #We can try different combination of variables:
predictor_var = ['Credit_History', 'Loan_Amount_Term', 'LoanAmount_log']
classification_model(model, df, predictor_var, outcome_var)

Accuracy : 89.414%
Cross-Validation Score : 67.909%
```

نلاحظ ان الدقة في الحالة الاولى كانت مساوية للطريقة الاولى رغم كوننا استخدمنا متغيرات مختلفة ولكن في الحالة الثانية زادت الدقة الى ما يقارب 90% وهو شيء جيد طبعاً.

نموذج الغابة العشوائية (Random Forest Model):

وهو خوارزمية تصنيف وتحليل اخرى اكثر تعقيداً من النماذج السابقة ومن فوائدها أننا نستطيع تطبيقها على كل المتغيرات و ستقوم بأرجاع مصفوفة تبين لنا اهمية كل من المتغيرات في تحديد التصنيف النهائي وكما في الامثلة التالية:

```
In [80]: model = RandomForestClassifier(n_estimators=100)
predictor_var = ['Gender', 'Married', 'Education',
                'Self_Employed', 'Loan_Amount_Term', 'Credit_History', 'Property_Area',
                'LoanAmount_log', 'TotalIncome_log']
classification_model(model, df, predictor_var, outcome_var)

Accuracy : 100.000%
Cross-Validation Score : 76.220%
```

ونرى هنا اننا وصلنا الى دقة 100% وهو أفضل ما يمكن ولكنه مخيف نوعاً ما فهو دلالة على وجود حالة (over fitting) والتخلص من ذلك يمكننا اتباع الطرق التالية:

1- تقليل عدد المتغيرات الداخلة الى هذا النموذج.

2- التلاعب بمعاملات النموذج.

والان دعونا نقوم بكل منهما:

```
In [81]: #Create a series with feature importances:
featimp = pd.Series(model.feature_importances_, index=predictor_var).sort_values(ascending=False)
print (featimp)

TotalIncome_log    0.282891
Credit_History     0.272377
LoanAmount_log     0.251326
Property_Area      0.049638
Loan_Amount_Term   0.044713
Married            0.028626
Education          0.023913
Self_Employed     0.023524
Gender             0.022994
dtype: float64
```

هنا قمنا بطباعة مصفوفة تبين وزن وقيمة كل من المتغيرات الداخلة في عملية بناء النموذج أعلاه (ومجموعها دائماً يجب أن يساوي 1).

والان نقوم بأخذ اهم 4 متغيرات (ذات اعلى اوزان) ونتلاعب بمعاملات النموذج وكما في ادناه:

```
In [83]: model = RandomForestClassifier(n_estimators=25, min_samples_split=25, max_depth=7, max_features=1)
predictor_var = ['TotalIncome_log', 'LoanAmount_log', 'Credit_History', 'Property_Area']
classification_model(model, df, predictor_var, outcome_var)

Accuracy : 82.248%
Cross-Validation Score : 80.135%
```

نلاحظ هنا أن الدقة انخفضت الى 82% فقط ولكن الدقة الناتجة من التدقيق المتقاطع (cross validation) اصبحت افضل من السابق وهذه علامة جيدة على التخلص من ال (over fitting) أو تقليل تأثيره على الأقل.

بعد معرفة كل هذه الامور تبقى الخطوة الاخيرة وهي تطبيق هذا النموذج على بيانات الاختبار (testing data) وهو شيء سهل جداً فقط نقوم باستبدال بيانات التدريب ببيانات الاختبار في الامثلة السابقة مع ملاحظة انها تحتاج الى التهيئة والتنظيف ايضاً قبل إدخالها الى نماذج التصنيف والتحليل. وكما في التوضيح التالي:

```
In [61]: ytest=np.ones((368,1))
```

```
In [56]: # save the model to disk   خزن نموذج التخمين في ملف في حاسوبنا الشخصي
import pickle
filename = 'finalized_model.sav'
pickle.dump(model, open(filename, 'wb'))
```

```
In [62]: # load the model from disk   (تطبيق نموذج التخمين على بيانات جديدة (بيانات الاختبار))
loaded_model = pickle.load(open(filename, 'rb'))
test=dfest['Credit_History']
result = loaded_model.score(test,ytest)
print(result)
```

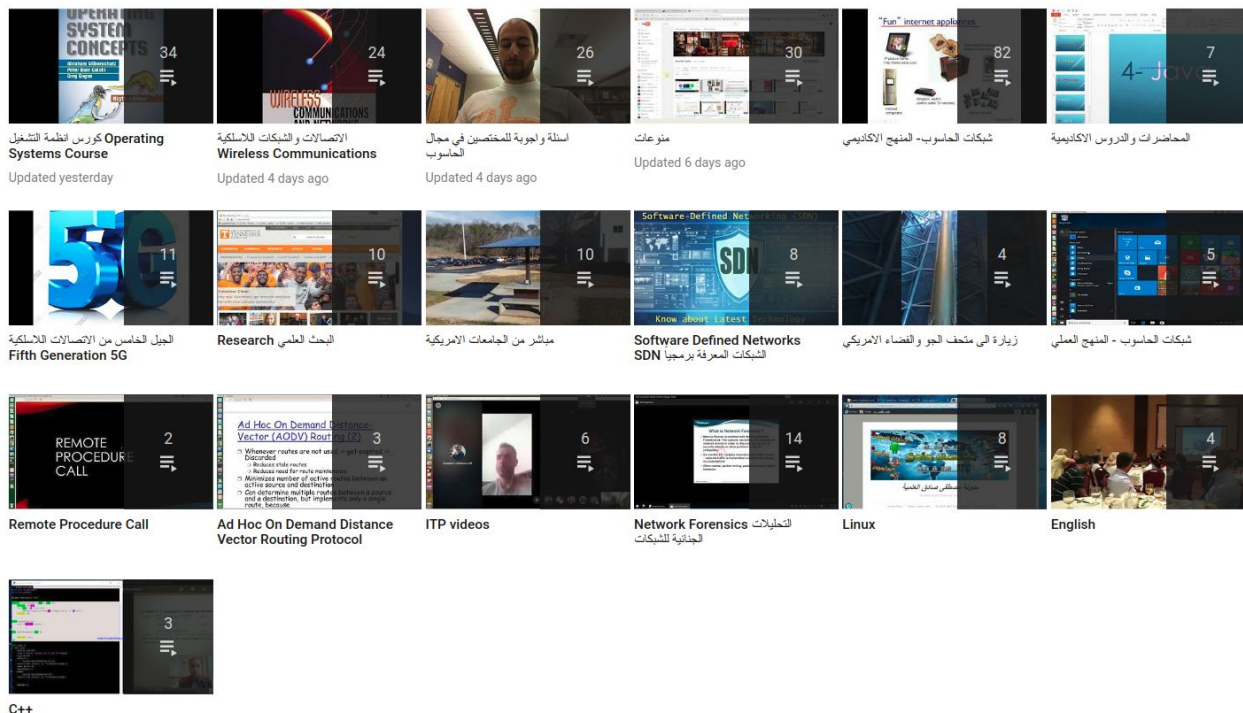
الى هنا ينتهي مشروعنا الاول على امل ان يكون قد افاد الجميع وقام بتوضيح الكثير من الامور حول علم البيانات واستخدامات لغة بايثون فيه حيث كما رأينا هناك الكثير من المكتبات الجاهزة التي نستطيع استدعائها بسهولة وهي توفر علينا الكثير من الوقت والجهد.

في حالة وجود اسئلة او استفسارات حول الدروس المشروحة فيمكنكم دائماً ترك الاسئلة في التعليقات في المدونة [هنا](#) ولا ننسى التذكير بالمصدر الأصلي لهذه الدروس الخاصة بعلم البيانات وهو موقع رائع انصح كل المهتمين بهذا المجال بمتابعته والاستفادة منه وهو في الرابط التالي (المصدر الاصلي باللغة الانكليزية).

في الختام لا يسعدنا الا ان نتمنى ان يكون الكتاب قد كان عند حسن ظن الجميع وكما هو واضح فالالمام بكل تفاصيل علم البيانات في لغة بايثون في كتاب واحد (مهما كان حجمه) شيء صعب جداً ولذا يمكن اعتبار هذا الكتاب فقط الخطوة الاولى في هذا الطريق وقد رأينا كيف ان التخصص في علم البيانات يحتاج معرفة جيدة على الاقل في المجالات التالية (البرمجة بلغة بايثون او R، مفاهيم الرياضيات والجبر الخطي العامة، فهم المشروع الذي نعمل عليه من وجهة نظر المختصين في ذلك المجال، فهم اساسيات تعليم الماكينة وبناء النماذج وتطبيقها على البيانات وكيفية فهم النتائج التي تعطينا اياها تلك النماذج والتذكر دائماً انها تعتمد التخمين والاحتمالية وليس فيها شيء مضمون 100%.

اخيراً يسرنا دوماً ان نراكم في المواقع التعليمية التالية:

القناة على اليوتيوب: www.youtube.com/mustafasadiq



مدونة مصطفى صادق العلمية



موسوعة تعليم اللغة الانكليزية تطبيق المدونة لأجهزة الاندرويد موسوعة المكتبات العربية للتحميل المجاني مكتبة الشروحات العربية
تحميل برامج الكمبيوتر مجاناً مكتبة المواقع العلمية والاكاديمية الهدف من الموقع أفضل المواقع العالمية في مجال الحاسوب مجلات حاسوبية
قناتي على اليوتيوب سؤال وجواب مجلدي في موقع الميديا فاير رواق الصراحة موسوعة مواقع التحديات البرمجية نبذة عن الموقع والقناة
قنوات يوتيوب مفيدة



المدونة العلمية: www.mustafasadiq0.com

تحياتي للجميع وانتظرونا في الاصدارات القادمة ان شاء الله
اخوكم مصطفى صادق لطيف