القوائـم في بايثـون
Lists in Python

الكاتب : سامر جبل

تاريخ الإصدار : ٢٠١٤\١١\١٤

للتواصل : samer.m.jabal@gmail.com

Thank you for trying Suite

فهرس المحتويات

Thank you for trying PDF Suite

بسم الله الرحمن الرحيم

الحمد لله الذي وفقني لهذا الكتاب، اللهم إجعله في ميزان حسناتي.

عن الكتاب:

((كُتِبَ إعتماداً على إصدار بايثون 2.7.x))

هذا الكتاب موجه أساساً للمبتدئين في بايثون، كُتب بأسلوب بسيط وباللغة العربية الفصحى (وقصدت بعدم استخدامي للهجة العامية (الدارجة) الوصول لعدد أكبر من المهتمين وإيصال المعلومة الكاملة) من ملاحظات كنت قد كتبتها أو جمعتها أثناء تعلمي للغة؛ حيث أن كل شرح أو ملاحظة حتى إذا كانت بسيطة أتبعتها بمثال واحد على الأقل.

وقد وجدت هذا الكتاب بأسلوبه المميز أفكاراً جيدة :

http://www.tutorialspoint.com/python/

فاقتبست بعضها .. فقد تجد تشابهاً في بعض الأجزاء وابتعاداً كبيراً عنه في بعضها..

ووضعت في نهاية الكتاب عدة تمارين تزيد من مهارة المتعلم متبوعة طبعاً بالحل (من أجل أن يتعرف على أخطاءه).

عن الكاتب:

كاتب الكتاب: سامر جبل من سوريا ، طالب جامعي ، ليس دارساً للبرمجة ولكن يأخذ البرمجة كـ شغف وليست هواية...
للتواصل الإيميل الإلكتروني: samer.m.jabal@gmail.com

Thank you for trying PDF Suite

مـقـدِّمـــة

نوع هياكل البيانات الأكثر شيوعاً في بايثون هو السلاسل(sequence)!.

بحيث أن كل عنصر من عناصر السلسلة يسند إلى رقم، الرقم يشير إلى موضعه (position) أو مؤشره (index).

المؤشـر:

في الحوسبة أو علوم الحاسوب، **المؤشر** هو نوع من أنواع البيانات في بعض لغات البرمجة تمثل قيمته (أو تشير إلى) قيمة أخرى مخزنة في مكان آخر في ذاكرة الحاسوب، وذلك باستخدام عنوان الذاكرة لها.

عملية الحصول على القيمة التي يشير إليها المؤشر تسمى تتبع المؤشر(dereferencing).

يعتبر المؤشر تطبيق بسيط لنوع البيانات العام المرجع برغم أنه مختلف عن الوسيلة المسماة مرجع(reference) في لغة سي++ .

تفيد المؤشرات إلى البيانات في تحسين الأداء للعمليات المتكررة مثل تمرير السلاسل الحرفية وتراكيب(بنيات) الأشجار.

وتستخدم المؤشرات للدوال في ربط الوسائل(methods) في البرمجة كائنية التوجه والربط عند وقت التشغيل بمكتبات الربط الديناميكية التي يرمز لها بـ (DLL).

>>> لمزيد من المعلومات عن مصطلح "مؤشر" إذهب إلى هنا

أول مؤشر هو الصفر 0 ، الثاني هو الواحد 1 وهكذا...

في بايثون هناك 6 أنواع من السلاسل المبنية داخلياً ، لكن الأكثر شيوعاً هي: السلاسل النصية(String)، الصفوف(tuple)، القواميس(dictionary)، والقوائم(List).. وهو ما سنتعلمه في هذا الملف التعليمي.

هناك أشياء محددة يمكنك القيام بها مع السلاسل من نوع القوائم!، سنمر عليها تباعاً.

Thank you

تعريف بالقوائم :

القوائم هي أكثر أنواع البيانات المتوفرة في بايثون استخداماً، ويمكن أن تكتب كـ قائمة من القيم(value) المفصولة عن بعضها بـ فاصلة < ، > ضمن قوسين مربعي الشكل < [] > والقيم تسمى item .

الشيء الجيد في القوائم أن القيم لا تحتاج لأن تكون من نفس النوع!

كيف تنشئ قائمة ؟

إنشاء قائمة أمر سهل فقط عليك أن تضع عدد من القيم ضمن قوسين مربعي الشكل وأن تفصل بين القيم بـ فاصلة..

ويمكن للقائمة أن تكون متعددة الأسطر .

أمثلـــة :

```
#!/usr/bin/python
List1 = [1, 2, 3, 4, 5]
List2 = ["a", "b", "c", "d"]
List3 = ["python", "is", "fun", 2014, 12, 11]
```

مثال لقائمة متعددة الأسطر :

```
#!/usr/bin/python
List4 = [
"this is",
"a multi-line",
"list!"
]
```

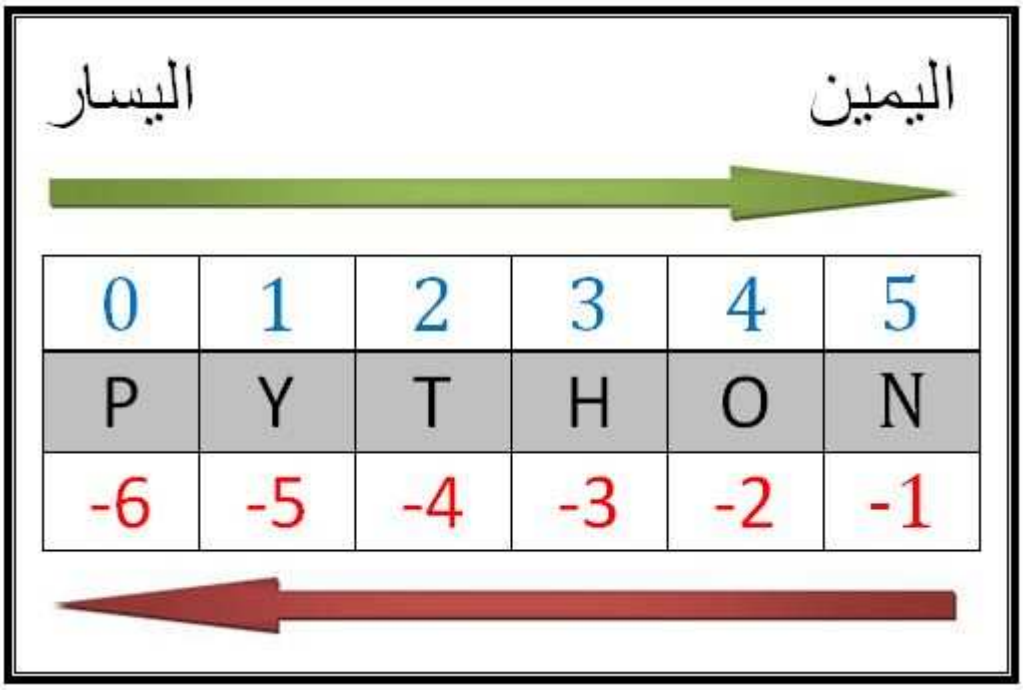
Thank you for using PDFSuite

ترقيم المؤشـر :

يبدأ ترقيم المؤشر من اليسار إلى اليمين بالصفر 0 ثم الواحد 1 وهكذا...

يبدأ ترقيم المؤشر من اليمين إلى اليسار بالواحد سالب 1- ثم بالإثنان سالب 2- وهكذا...

>>> في الصورة التالية ستتضح الفكرة ، الشرح يتم بها على كلمة "PYTHON"



Thank you for using PDFSuite

الولوج إلى قيم القائمة

للولوج إلى القيم في القوائم، استخدم الأقواس مربعة الشكل من أجل التمرير (slicing) على طول السلسلة بإستخدام المؤشر أو المؤشرات للحصول على القيم المتوفرة عند ذلك المؤشر .

في ما يلي عدة أمثلـــة ستوصل الفكرة إليك :

```
#!/usr/bin/python
a_List = [1, 2, 3, 4, 5, 6, 7, 8, 9]
#الحصول على أول قيمة من القائمة.
>>> a_List[0]
1
#الحصول على ثاني قيمة من القائمة.
>>> a_List[1]
2
#الحصول على آخر قيمة من القائمة.
>>> a_List[8]
9
>>> a_List[-1]
9
#الحصول على قيمة المؤشر رقم 7 من القائمة.
>>> a_List[7]
8
```

للتمرير على طول السلسلة; القوسين مربعي الشكل يحويان القيم التالية :

[الخطوة : النهاية : البداية]

Thank you for trying PDF Suite

[start : end : step]

يتم التعامل معها بهذا الشكل :

[start, start+step, start+(step*2), start+(step*3)……etc]

القيمة الإفتراضية لنقطة البداية هي : 0

القيمة الإفتراضية للخطوة هي : 1

#الحصول على القيم من المؤشر 2 إلى المؤشر 7.

```
>>> a_List[2:8]
[3, 4, 5, 6, 7, 8]
```

#الحصول على القيم من بعد أول قيمة!

```
>>> a_list[1:]
[2, 3, 4, 5, 6, 7, 8, 9]
```

#الحصول على نسخة من القائمة!

```
>>> copy_of_list = a_List[:]
>>> copy_of_list
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

#الحصول على القيم من المؤشر 0 إلى المؤشر 5. أي: أول 6 قيم..

```
>>> a_List[0:6]
[1, 2, 3, 4, 5, 6]
>>> a_List[:6]
[1, 2, 3, 4, 5, 6]
```

#الحصول على قيم القائمة كاملة عدا القيمة الأخيرة.

```
>>> a_List[:-1]
[1, 2, 3, 4, 5, 6, 7, 8]
>>> a_List [:][:-1]
```

Thank you for trying PDF Suite

[1, 2, 3, 4, 5, 6, 7, 8]

#الحصول على القيمة الأخيرة على شكل قائمة!

>>> a_List[-1:]

[9]

#(حرر تفكيرك...)في هذه الحالات سنحصل على قائمة فارغة! .. سنعطي الحل

>>> a_List [8:3]

[]

>>> a_List [-1:-6]

[]

>>> a_List [10:0:-2]

[]

الـحـــــل :

>>> a_List [3:8]

[4, 5, 6, 7, 8, 9]

>>> a_List [8:3:-1]

[9, 8, 7, 6, 5]

>>> a_List [-6:-1]

[4, 5, 6, 7, 8]

#فلننشئ قائمة تحوي ضمنها قائمة أخرى(فرعية) ، لنحاول الولوج إلى قيم القائمة الفرعية

another_list = [1, 2, 3, 4, [5, 6, 7, 8], 9]

الطريقة تتم هكذا...

إسم القائمة[مؤشر القائمة الأساس][مؤشر القائمة الفرعية]

list_name[index][sub-index]

>>> another_list[4][2]

7

Thank you for trying PDF Suite

```
>>> another_list[4][:2]
[5, 6]
```

وبنفس الطريقة للسلاسل النصية :

```
Str_within_list = ["python"]
>>> Str_within_list[0][2]
't'
```

#مثال آخر لتراكب(overlying) القوائم ضمن بعضها البعض..

((لقد تماديت في الشرح هههههه، لكن كلما تعرفت على أمثلة أكثر زادت مهارتك))

```
>>> overlying_lists = [1, 2, 3, [4, 5 , [6, 7, 8]], 9]
>>> overlying_lists[3][2][1]
7
```

#الآن فلنحاول الولوج لقيمة(value) لقاموس(dictionary) موجود ضمن قائمة... في هذا الملف لن يتم شرح القواميس !! ،ما يجب أن تعرفة هو التالي:

القاموس في بايثون عبارة عن: أزواج(مفتاحkey وقيمةvalue) وكل زوج يسمىitem ،يفصل بين المفتاح والقيمة نقطتين عاموديتين < : > ، والموضوعة ضمن قوسين مجعدين! < {} > .

```
dict_within_list=[{"name":"Samer"}, {"name":"Mohammad"}]
```

الطريقة تتم هكذا...

إسم القائمة[مؤشر القائمة الأساس][مفتاح القاموس]

list_name[index][key]

```
>>> dict_within_list[0]['name']
' Samer '
>>> dict_within_list[1]['name']
' Mohammad'
```

ـــ تم بعون الله قسم : الولوج إلى قيم القائمة ـــ

Thank you for trying PDF Suite

تحديث القوائم

((إضافة عنصر أو عدة عناصر إلى القائمة))

يمكنك تحديث عنصر(element) أو عدة عناصر من القائمة.. من خلال إحدى الطرق التالية:

١. **إعطاء رقم المؤشر(slice) في الطرف اليساري من عملية الإسناد(assignment) '='، والقيمة في الطرف اليميني اليميني ، يتم بها إستبدال قيمة موجودة في القائمة بأخرى! .**
٢. **جمع القوائم بإستخدام '+' أو ما يسمى بالبرمجة : التسلسل(Concatenation).**
٣. **بإستخدام الوسيلة(method) : ()append بالعربية: الإلحاق، تمكنك من إضافة عنصر واحد فقط!!..**
٤. **بإستخدام الوسيلة(method) : ()extend بالعربية: الإمتداد، تمكنك من إضافة عناصر كائن(object) قابل للعد(iterable) إلى القائمة!!.. سنتطرق إليها في هذا القسم لنرى الفرق بينها وبين ()append .**
٥. **بإستخدام الوسيلة(method) : ()insert بالعربية: الإدخال، تمكنك من إضافة عنصر إلى عناصر القائمة في موضع محدد(specific position).**

---

1. إعطاء رقم المؤشر(slice) في الطرف اليساري من عملية الإسناد(assignment) '='، والقيمة في الطرف اليميني ، يتم بها إستبدال قيمة موجودة في القائمة بأخرى! .

الطريقة تتم هكذا...

إسم القائمة[المؤشر] = القيمة

List_name[index] = 'item'

```
#!/usr/bin/python
lst = ["a", "b", 4, 8, 3, "s"]
>>> lst[5] = "py"
>>> lst
["a", "b", 4, 8, 3, "py"]
```

Thank you for trying PDFSuite

استبدال قيمة بأخرى...

```
>>> lst[3] = 7
>>> lst
["a", "b", 4, 7, 3, "py"]
```

2. جمع القوائم بإستخدام '+' أو ما يسمى بالبرمجة : التسلسل(Concatenation).

إضافة محتويات قائمة إلى قائمة أخرى من خلال التسلسل..

```
#!/usr/bin/python
>>> [1, 2, 3] + [4, 5, 6]
[1, 2, 3, 4, 5, 6]
>>> w = ["a", "b", "c"]
>>> m = [5, 7, 4]
>>> w + m
["a", "b", "c", 5, 7, 4]
```

3. بإستخدام الوسيلة(method) : ()append بالعربية: الإلحاق، تمكنك من إضافة عنصر واحد فقط!!..

تمكنك من إضافة عنصر إلى نهاية القائمة..

الطريقة تتم هكذا...

list_name.append(an object)

```
#!/usr/bin/python
>>> lst2 = ["a", "b"]
>>> lst2.append("c")
>>> lst2
["a", "b", "c"]
```

Thank you for trying PDFSuite

```
>>> lst2.append(2)
>>> lst2
["a", "b", "c", 2]
```

4. بإستخدام الوسيلة(method) : ()extend بالعربية: الإمتداد، تمكنك من إضافة عناصر كائن(object) "قابل للعد" أو "قابل للترتيب في صفوف مرقمة" إذا صح التعبير(iterable) إلى القائمة!!..

الوسيلة(method) : ()extend هي وسيلة للكائن الذي هو القائمة حيث أنها تقبل فقط الكائن القابل للعد كـ مُدخل وتضيف قيمَهُ إلى القائمة، عادة يتم تمرير(pass) قائمة إليها..

تقريباً هي تقوم بنفس مهام الإلحاق ()append والتسلسل Concatenation.

الطريقة تتم هكذا...

List_name.extend(an iterable object)

أمثلـــة :

```
#!/usr/bin/python
>>> a = [1, 2]
>>> b = [3, 4]
>>> a.extend(b)
>>> a
[1, 2, 3, 4]
```

```
>>> c = [1, 2]
```

Thank you for trying PDFSuite

```
>>> d = "python"
>>> c.extend(d)
>>> c
[1, 2, 'p', 'y', 't', 'h', 'o', 'n']
```

```
# إنشاء قائمة فارغة! ...
>>> e =[]
>>> f = {"py":35, "tw":67}
>>> e.extend(f)
>>> e
['tw', 'py']
```

الإختلافات بين append() وَ extend()

```
#!/usr/bin/python
>>> a_list = ['a', 'b', 'c']
>>> a_list.extend(['d', 'e', 'f'])
>>> a_list
['a', 'b', 'c', 'd', 'e', 'f']
>>> len(a_list)
6
>>> a_list[-1]
'f'
>>> a_list.append(['g', 'h', 'i'])
```

Thank you for trying PDF Suite

```
>>> a_list
['a', 'b', 'c', 'd', 'e', 'f', ['g', 'h', 'i']]
>>> len(a_list)
7
>>> a_list[-1]
['g', 'h', 'i']
```

5. بإستخدام الوسيلة(method) : ()insert بالعربية: الإدخال، تمكنك من إضافة عنصر إلى عناصر القائمة في موضع محدد(specific position).

يمكن إضافة العنصر إلى موضع لم يوجد بعد في القائمة!(أول مثال يوضح الطريقة تتم هكذا...

list_name.insert(index, item)

```
#!/usr/bin/python
>>> t = [1, "a", 6, 4, "z"]
>>> t.insert(5, 8)
>>> t
[1, "a", 6, 4, "z", 8]
>>> t. insert(5, "r")
 >>> t
[1, "a", 6, 4, "z", "r", 8]
```

Thank you for trying PDFSuite

ـــــ تم بعون الله قسم : تحديث القوائم ـــــ

حذف العناصر من القائمة

لحذف العناصر من القائمة ، هناك ثلاث طرق :

١. بإستخدام الوسيلة(method) : remove() ، بالعربية : إزالة .

٢. بإستخدام التعبير(ststement) : del ،مأخوذة من delete وتعني: حذف

٣. بإستخدام الوسيلة(method) : pop() ، بالعربية : " الفُجائية " .

---

1. بإستخدام الوسيلة : remove()

نستخدم هذه الطريقة إذا كنا لا نعرف تحديداً موضع العنصر في القائمة!!

list_name.remove('item')

```
#!/usr/bin/python
>>> lst33 = ["a", "b", "c", "d"]
>>> lst33.remove("c")
>>> lst33
["a", "b", "d"]
```

2. بإستخدام التعبير : del

نستخدم هذه الطريقة إذا كنا نعرف تحديداً موضع العنصر في القائمة!!

del list_name[index]

```
#!/usr/bin/python
>>> lst = ["a", "b", "c", "d"]
>>> del lst[2]
>>> lst
```

Thank you for trying PDFSuite

["a", "b", "d"]

3. بإستخدام الوسيلة : pop()

تقوم بحذف العنصر عند المؤشر الذي تحدده أنت، وتطبعه على الشاشة!

إذا لم تمرر لها أي مؤشر ستحذف آخر عنصر من القائمة، وتطبعه على الشاشة!

list.pop(obj=list[-1])

أمثلـــة :

```
#!/usr/bin/python
>>> lst7 = ["a", "r", 8, 3, "d"]
>>> lst7.pop()
'd'
>>> lst7
["a", "r", 8, 3]
>>> lst7.pop(2)
8
>>> lst7
["a", "r", 3]
```

ـــ تم بعون الله قسم : حذف العناصر من القائمة ـــ

Thank you for trying PDF Suite

وسائل القوائم – List methods

شرحنا مما سبق هذه الوسائل :

append() , extend() , insert() , remove() , pop()

list.append(obj)

list.extend(seq)

list.insert(index, obj)

list.remove(obj)

list.pop(obj=list[-1])

بايثون تتضمن إضافة لما سبق العديد من الوسائل للتعامل مع القوائم، سنشرحها تباعاً..

1. الوسيلة : count() ، بالعربية تعني : الـعَــدْ .

تعود برقم! ، الرقم هو: عدد مرّات ظهور العنصر < obj > في القائمة.

list.count(obj)

```
>>> [1, 2, 5, 1, 7, 9, 1, 0].count(1)
3
```

2. الوسيلة : index() ، بالعربية : المؤشِّـر .

تعود برقم، الرقم هو: أقل مؤشر وجد فيه العنصر < obj > في القائمة.

list.index(obj)

```
>>> [1, 2, 5, 1, 7, 9, 1, 0].index(1)
0
```

Thank you for trying PDF Suite

3. الوسيلة : sort() ، بالعربية : الترتيب .

تقوم بترتيب عناصر القائمة ، وتَستخدِم دالة(وظيفة\function) للترتيب إذا تم إعطائها!.

((الشبكة مليئة بالشروحات عن هذه الوسيلة؛ بحث google بسيط يكفي!!))

list.sort([func])

```
>>> a = [7, 4, 0, 3, 1, 1, 6]
>>> a.sort()
>>> a
[0, 1, 1, 3, 4, 6, 7]
>>> b = ["a", "b", 3, 1, "n", 5]
>>> b.sort()
>>> b
[1, 3 , 5, 'a', 'b', 'n']
```

خطأ شائع

إسناد list.sort() إلى متغير يعود بالقيمة:

None

القيام بالعملية التالية أيضاً يعود بالقيمة:

None

```
>>> [5, 7, 2, 1, 0].sort()
>>> print  [5, 7, 2, 1, 0].sort()
None
```

4. الوسيلة : reverse() ، بالعربية : العكس .

تقوم بعكس مواضع عناصر القائمة!! ضمن نفس القائمة.

list.reverse()

```
>>> v = [2, 6, "d", "n", 1]
>>> v.reverse()
>>> v
[1, 'n', 'd', 6, 2]
```

ـــ تم بعون الله قسم : وسائل القوائم ـــ

Thank you for the suite

عمليات القوائم الأساسية :

القوائم تستجيب لعمليات الـ '+' التسلسل concentenation وَ '*' التكرار repetition تماماً مثل السلاسل النصية، والنتيجة هي قائمة جديدة .

في الواقع، القوائم تستجيب لجميع عمليات السلاسل المستخدمة في السلاسل النصية.

هذا الجدول يظهر كيف تعمل العمليات الأساسية في القوائم..

العملية :	النتيجة :	الوصف :
len([1, 2, 3])	3	طول القائمة
[1, 2, 3]+[4, 5, 6]	[1, 2, 3, 4, 5, 6]	التسلسل
['Hi!']*4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	التكرار
3 in [1, 2, 3]	True	العضوية membership
for x in [1, 2, 3]: print x	1 2 3	العـد iteration

استخدام بعض الدوال المبنية داخلياً مع القوائم :

1. دالة الطول len() مأخوذة من : length .

len(list)

تعود بالطول الإجمالي للكائن القابل للعد..

```
>>> len([1, 3, 5, "e", "p"])
5
```

2. دالة القيمة العليا max() مأخوذة من : maximum .

max(list)

Thank you for trying PDFSuite

تعود بأعلى قيمة موجودة ضمن القائمة..

```
>>> max([2, 4, 6, 8, 9, 1])
9
```

3. دالة القيمة السفلى min() مأخوذة من : minimum .

min(list)

تعود بأقل قيمة موجودة ضمن القائمة..

```
>>> max([2, 4, 6, 8, 9, 1])
1
```

4. دالة المقارنة cmp() مأخوذة من : compare .

cmp(list1, list2)

إذا تحققت list1 == list2 النتيجة صفر 0 .

إذا تحققت list1 > list2 النتيجة موجبة 1 .

إذا تحققت list1 < list2 النتيجة سالبة -1 .

```
>>> cmp([1, 2, 3], [1, 2, 3])
0
>>> cmp([1, 2, 3], [1, 2])
1
>>> cmp([1, 2, 3], [4, 5])
-1
>>> cmp([1, 2, 3], [3, 2, 1])
-1
```

Thank you for trying PDF Suite

```
>>> cmp([1, 2, 3], [4, 5, 6])
-1
>>> cmp([1, 2, 3], [2, 3, 1])
-1
```

الـدالـــة : قائمة ـ list()

list([iterable])

تعود إلينا بقائمة من العناصر التي هي نفس العناصر ولها نفس الترتيب للكائن القابل للعد(الترقيم).

الكائن القابل للعد أو القابل للترقيم : يمكن أن يكون إما متسلسل(sequence) ، حاوية(container) تحوي عناصر أو بيانات تدعم العد\الترقيم ، أو أي كائن قادر على إعطاء أرقام لعناصره(iterator object).

إذا كان الكائن القابل للعد(الترقيم) هو بالفعل قائمة!، عندها يتم عمل نسخة منها وتعاد لنا.. هذا مشابه لـ : iterable[:] ، تكلمنا عنه سابقاً .

كـ مثال :

list('abc') تعود بـ: ['a', 'b', 'c'] وَ list((1, 2, 3)) تعود بـ: [1, 2, 3]

إذا لم يتم تمرير أي شيء إلى الدالة، فهي تعود بقائمة جديدة فارغة [] .

القائمة هي نوع من أنواع البيانات قابل للتحول\التغير(mutable) إلى نوع آخر متسلسل كـ : string , tuple .... إلخ. بإستخدام دوالٍ مثـل:

str() , tuple() .....etc

Thank you for trying PDF Suite

تمارين :

(التمرين الأول)

أكتب دالة وليكن اسمها: rev_lst(x) ، تقبل متغيراً واحداً x ..

بحيث أنك تمرر لها قائمة ما وتعود لك بعكسها!! (لا تستعمل الوسيلة reverse())

مـثـــال :

```
>>> rev_lst([1, 2, 3, 4, 5])
[5, 4, 3, 2, 1]
```

(التمرين الثاني)

أكتب دالة وليكن اسمها: rev_lst_into_lsts(y) ، تقبل متغيراً واحداً y

بحيث أنك تمرر لها قائمة ما وتعود لك بعكسها لكن كل عنصر من عناصر القائمة يكون عبارة عن قائمة فرعية!!! (لاااا تستعمل الوسيلة reverse())

مـثـــال :

```
>>> rev_lst_into_lsts([1, 2, 3, 4, 5])
[[5], [4], [3], [2], [1]]
```

بعدها اطبع أول ثلاث قيم من القائمة (كل قيمة لوحدها).

(التمرين الثالث)

أكتب دالة وليكن اسمها: mix_lsts(z, w) ، تقبل متغيرين z وَ w .

بحيث أنك تمرر لها قائمتين فتقوم بخلطهما!!!؟؟...

مـثـــال :

```
>>> mix_lsts(["a", "b"], ["c", "d"]
['ac', 'ad', 'bc', 'bd']
```

حل التمارين في الصفحة التالية

Thank you for trying PDF Suite

حل التمارين

حل التمرين الأول :

```
def rev_lst(x):
    ans  = []
    while len(x)>0:
        ans.append(x[-1])
        x = x[:-1]
    return ans
```

حل التمرين الثاني :

```
def rev_lst_into_lsts(y):
    ans = []
    while len(y)>0:
        ans.append(y[-1:])
        y = y[:-1]
    return ans
طباعة أول ثلاث عناصر كلٌ على حدى :
print rev_lst_into_lsts(y)[0]
print rev_lst_into_lsts(y)[1]
print rev_lst_into_lsts(y)[2]
```

حل التمرين الثالث :

طريقة أولى (الأفضل) :

```
def mix_lsts(z, w):
    ans = []
    for i in z:
        for j in w:
            ans.append(i+j)
    return ans
```

طريقة ثـانيــة :

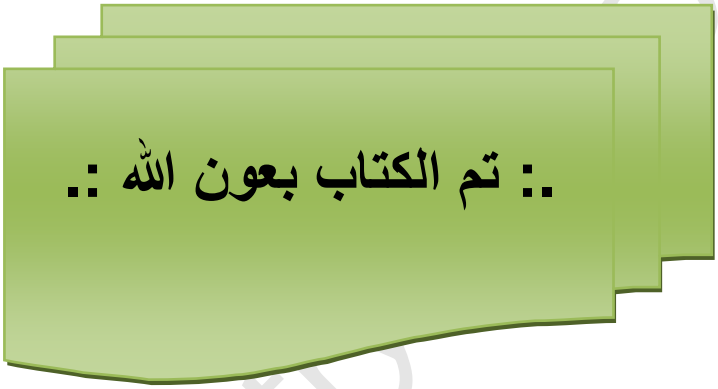
```
def mix_lsts(z, w):
    ans = []
    for i in range(len(z)):
        for j in range(len(w)):
            ans.append(z[i]+w[j])
    return ans
```

روابط مفيدة :

http://www.tutorialspoint.com/python/python_lists.htm

http://www.python.org/

http://tinyurl.com/getting-started-python-ppl



Thank you for trying PDF Suite