Statistics with Python by Example

الإحصاء في بايسون بالأمثلة

يعتبر بايسون من أقوى البرامج الحديثة حيث يتميز بمكتبة شاملة من البرامج التي تشمل التعامل مع الشبكات الحاسوبية و قواعد البيانات و معالجة XML و معالجة البيانات الضخمة.

سوف نتطرق لأساسيات البرمجة بلغة بايسون مستخدمين الأمثلة في ذلك وسوف نتدرج من السهل حتى نصل لمستوى من التعقيد المناسب.

يوجد إصدارين من بيسون الإصدار 2.7 (والأخير في السلسلة) و الإصدار 3.x الأحدث.
المشكلة ان الإصدار 3.x غير متوافق خلفيا مع الإصدار 2.7 والذي يتميز بوجود مكتبات كثيرة متوافقة معه. لذلك معظم التطبيقات لاتزال تستخدم الإصدار 2.7

تحميل و تثبيت Python

الموقع الرسمي لـ Python هو

https://www.python.org/

وتوجد واجهات إستخدام عدة لـ Python منها Anaconda وهو برنامج مجاني مفتوح لمعالجة البيانات الكبيرة Larg-scale Data processing و التحليل التوقعي Predictive Analysis و الحسابات العلمية Scientific Computing .

سوف نستخدم Anaconda في محاضراتنا ويمكن تحميل البرنامج من

https://repo.continuum.io/archive/

وآخر إصدار هو:

لنظام win32

https://repo.continuum.io/archive/Anaconda3-2.2.0-Windows-x86.exe

لنظام win64

https://repo.continuum.io/archive/Anaconda3-2.2.0-Windows-x86_64.exe

يمكنك أيضا تحميل البرنامج الأساسي من:

https://www.python.org/downloads/

ومن ثم إضافة المكتبات Modules الاخرى حسب الطلب.

الموقع الرسمي لبايسون هو www.python.org و يمكن تحميل بايسون لأنظمة التشغيل windows و Mac S و linux الخ من ذلك الموقع.

يمكن إدخال الترميز من خلال محرر نصي مثل Notepad في النوافذ و إجرائه من خلال خط الأوامر CLI للنوافذ.

تراكيب البيانات الأساسية في Python

أي عملية على البيانات في Python تعتمد على تركيب تلك البيانات فمثلا لو حاولنا تطبيق دالة على بيانات غير معروف نوعها لتلك الدالة سوف تصدر رسالة خطأ (سوف تصادف بعضها أثناء الإستخدام)

تعاريف و أساسيات:

أنواع البيانات:

1) أعداد صحيحة **Integers**

مثل:

```
1, -3, 44, 355, 98546789, -654983
```

2) الفاصلة العشرية **Floats**

مثل:

```
3.0, 33e12, -7e-4
```

3) الأعداد المركبة **Complex numbers**

مثل:

```
3 + 2j, -4 - 3j,3.2 + 4.2j
```

3) منطقي **Boolean**

مثل:

```
True, False
```

(قيمتين فقط)

يمكن معالجة هذه الأنواع بالعمليات الحسابية (+) الجمع و (-) الطرح و (*) الضرب و (/) القسمة و (**) الرفع لقوة و (%) القياس (أو باقي القسمة)

مثل:

الأعداد الصحيحة

```
>>> x = 5 + 2 - 3 * 2
>>> x
1
>>> 5 / 2
2.5                          <= التاتج عشري
>>> 5 // 2
2                            <= الناتج قطع
>>> 5 % 2
1
>>> 2 ** 8
256
>>> 1000000001 ** 3
1000000003000000003000000001
```

الأعداد العشرية

```
>>> x = 4.3 ** 2.4
>>> x
33.137847377716483
>>> 3.5e30 * 2.77e45
9.6950000000000002e+75
>>> 1000000001.0 ** 3
1.000000003e+27
```

الأعداد المركبة

```
>>> (3+2j) ** (2+3j)
(0.68176651908903363-2.1207457766159625j)
>>> x = (3+2j) * (4+9j)
>>> x
(-6+35j)
>>> x.real
-6.0
>>> x.imag
35.0
```

مثال على للعمليات على النوع المنطقي

```
>>> x = False
>>> x
False
>>> not x
True
>>> y = True * 2
>>> y
2
```

لاحظ ان رغم تمثيل النوع المنطقي بـ True و False الا انها تتصرف كالأرقام 1 (True) و 0 (False) .

أنواع تراكيب البيانات:

1) **Lists** (قوائم) متتبعات مرتبة يمكن تغيير عناصرها mutable وتعرف بـ [ ]

مثل:

```
[]
[1]
[1, 2, 3, 4, 5, 6, 7, 8, 12]
[1, "two", 3L, 4.0, ["a", "b"], (5,6)]
```

عمليات على القوائم:

التقسيم او التشريح Slicing

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[0]
'first'
>>> x[2]
'third'
>>> x[-1]
'fourth'
>>> x[-2]
'third'
>>> x[1:-1]
['second', 'third']
>>> x[0:3]
['first', 'second', 'third']
>>> x[-2:-1]
['third']
>>> x[:3]
['first', 'second', 'third']
>>> x[-2:]
```

```
['third', 'fourth']
```

الإضافة و التغيير

```
>>> x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x[1] = "two"
>>> x[8:9] = []
>>> x
[1, 'two', 3, 4, 5, 6, 7, 8]
>>> x[5:7] = [6.0, 6.5, 7.0]
>>> x
[1, 'two', 3, 4, 5, 6.0, 6.5, 7.0, 8]
>>> x[5:]
[6.0, 6.5, 7.0, 8]
```

دوال على القوائم

```
>>> x = [1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> len(x)
9
>>> [-1, 0] + x
[-1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> x.reverse()
>>> x
[9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> max(x)
9
>>> min(x)
1
>>> x.append(10)
>>> x
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

2) **Tuples** (مرتبات) متتابعات مرتبة لايمكن تغيير عناصرها immutable وتعرف بـ ( ) أو بدون أقواس

```
()
(1,)
(1, 2, 3, 4, 5, 6, 7, 8, 12)
(1, "two", 3L, 4.0, ["a", "b"], (5, 6))
```

تحويل قائمة لمرتبة والعكس

```
>>> x = [1, 2, 3, 4]
>>> tuple(x)
(1, 2, 3, 4)
```

```
>>> x = (1, 2, 3, 4)
>>> list(x)
[1, 2, 3, 4]
```

3) **Dictionaries** (قاموس) وتعرف تطبيق mapping من مفاتيح keys إلى قيم values وتعرف بـ { }

مثل

```
>>> x = {1: "one", 2: "two"}
>>> x["first"] = "one"
>>> x[("Delorme", "Ryan", 1995)] = (1, 2, 3)
>>> list(x.keys())
['first', 2, 1, ('Delorme', 'Ryan', 1995)]
>>> x[1]
'one'
>>> x.get(1, "not available")
```

```
'one'
>>> x.get(4, "not available")
'not available'
```

3) **Strings** (نصوص) وهي متتابعة من الرموز وتعرف بـ " " أو ' '

مثال

```
>>> e = 2.718
>>> x = [1, "two", 3, 4.0, ["a", "b"], (5, 6)]
>>> print("The constant e is:", e, "and the list x is:", x)

The constant e is: 2.718 and the list x is: [1, 'two', 3, 4.0,
['a', 'b'], (5, 6)]
>>> print("the value of %s is: %.2f" % ("e", e))
the value of e is: 2.72
```

(3) Sets (مجموعات) وهو تجميعة من الأشياء غير المرتبة ويستخدم هذا التركيب عندما يكون أهم شيئ هو الإنتماء و الوحدانية للمجموعة فقط

مثال

```
>>> x = set([1, 2, 3, 1, 3, 5])
>>> x
{1, 2, 3, 5}
>>> 1 in x
True
>>> 4 in x
False
>>>
```

(4) **File object** أشياء الملفات: يمكن الوصول لملف عن طريق شيئ ملف

مثال

```
>>> f = open("myfile", "w")
>>> f.write("First line with necessary newline character\n")
44
>>> f.write("Second line to write to the file\n")
33
>>> f.close()
>>> f = open("myfile", "r")
>>> line1 = f.readline()
>>> line2 = f.readline()
>>> f.close()
>>> print(line1, line2)
First line with necessary newline character
Second line to write to the file
>>> import os
>>> print(os.getcwd())
c:\My Documents\test
>>> os.chdir(os.path.join("c:", "My Documents", "images"))
>>> filename = os.path.join("c:", "My Documents",
"test", "myfile")
>>> print(filename)
c:\My Documents\test\myfile
>>> f = open(filename, "r")
>>> print(f.readline())
First line with necessary newline character
>>> f.close()
```

تركيبات التحكم في الإنسياب Control flow structures

و تشمل الدورات Looping و التشعب Branching و التي تتحكم بها تعابير تقيم إلى نتيجة منطقية يكون نتيجتها تحقيق إختبار ما بإستخدام عمال المقارنة (>, >=, !=, is, is not, in, not in) و عمال المنطق (and, not, or) والتي ينتج منها جميعا إما True أو False.

العبارة if-elif-else

مثال

```
>>> x = 5
>>> if x < 5:
...     y = -1
...     z = 5
... elif x > 5:
...     y = 1
...     z = 11
... else:
...     y = 0
...     z = 10
... print(x, y, z)
5 0 10
```

العبارة while loop

مثال

```
>>> u, v, x, y = 0, 0, 100, 30
>>> while x > y:
...     u = u + y
...     x = x - y
...     if x < y + 2:
...         v = v + x
...         x = 0
...     else:
...         v = v + y + 2
...         x = x - y - 2
...     print(u,v)
...
30 32
60 40
```

الدورة for

مثال

```
item_list = [3, "string1", 23, 14.0, "string2", 49, 64, 70]
>>> for x in item_list:
...     if not isinstance(x, int):
...         continue
...     if not x % 7:
```

```
...         print("found an integer divisible by 7: %d" % x)
...         break
...
found an int div by 7: 49
>>>
```

تعريف الدوال Function definition

أمثلة

```
>>> def funct1(x, y, z):
... value = x + 2*y + z**2
... if value > 0:
... return x + 2*y + z**2
... else:
... return 0
...
>>> u, v = 3, 4
>>> funct1(u, v, 2)
15
>>> funct1(u, z=v, y=2)
23
>>> def funct2(x, y=1, z=1):
... return x + 2 * y + z ** 2
...
>>> funct2(3, z=4)
21
>>> def funct3(x, y=1, z=1, *tup):
... print((x, y, z) + tup)
...
>>> funct3(2)
(2, 1, 1)
>>> funct3(1, 2, 3, 4, 5, 6, 7, 8, 9)
(1, 2, 3, 4, 5, 6, 7, 8, 9)
```

```
>>> def funct4(x, y=1, z=1, **dictionary):
... print(x, y, z, dict)
>>> funct4(1, 2, m=5, n=9, z=3)
1 2 3 {'n': 9, 'm': 5}
```

تصيد الأخطاء Exceptions

توجد بيد المستخدم بعض الأوامر التي تمكنه لإصتياد الأخطاء وتوضع في أماكن من الترميز للتمكن من ذلك وسوف نشير اليها لاحقا عند إستخدامها.

توليد برمجيات Module creation

سوف لانتطرق لهذا الموضوع المتقدم في هذا الكتاب الأولي.

البرمجة المنحى شيئي Object-oriented programming

سوف لانتطرق لهذا الموضوع المتقدم في هذا الكتاب الأولي رغم اننا سوف نستخدمها بطريقة ضمنية في امثلتنا.

ملاحظات أساسية ومهمة:

1) فراغات أول السطر (إزاحات) **Indentation** و التركيب القالبي **Block structure**

يختلف بايسون عن بقية البرامج في كونه يستخدم الفراغات و الإزاحات الداخلة لتحديد تركيب قالبي (مجموعة من الترميز المتعلقة ببعضها) مثل قلب دورة فمعظم اللغات تستخدم أقواس من انواع خاصة لتحديد القوالب. بعض البرامج تستخدم الإزاحة لغرض تحسين قرائة البرنامج

ولكن في بايسون الإزاحة جزء أساسي من التركيب اللغوي Syntax

ملاحظة: الإزاحة من مستوى لآخر تتكون من 4 فراغات

مثال

```
item_list = [3, "string1", 23, 14.0, "string2", 49, 64, 70]
>>> for x in item_list:
...     if not isinstance(x, int):
...         continue
...     if not x % 7:
...         print("found an integer divisible by 7: %d" % x)
...         break
...
```

رمزنا للفراغ بـ .

ملاحظة: التعليقات comments تبدأ بالرمز # .

المتغيرات Variables و الإسناد Assignments

مثال

```
x = 5
```

المتغير x اسند له العدد الصحيح 5.

في بايسون لا نعلن نوع المتغير أو فاصل لتبيين نهاية السطر كما في معظم البرامج الاخرى.

متغيرات بايسون يمكن ان تكون اي شيئ بعكس اللغات الاخرى التي فيها لايمكن لمتغير اخذ قيمة غير التي اعلن انه منها. الترميز التالي يوضح ذلك

مثال

```
>>> x = "Hello"   <=  متغير نصي x
>>> print(x)
Hello
>>> x = 5          <=     متغير عددي x
>>> print(x)
5
```

أي إسناد جديد يلغي الإسناد السابق.

الأمر del يحذف المتغير

مثال

```
>>> x = 5
>>> print(x)
5
>>> del x
>>> print(x)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
>>>
```

لاحظ العبارة Traceback والتي تطبع عند وجود او إكتشاف خطأ او exception و إعلان أن الخطأ هو NameError حيث الإسم x غير معرف.

ملاحظة مهمة: أسماء المتغيرات حساسة الحالة Case sensitive فالمتغير x يختلف عن المتغير X.

التعابير Expressions

التعابير هي صيغ وعلاقات بين المتغيرات او هي صيغ حسابية او جبرية.

مثال

```
x = 3
y = 5
z = (x + y) / 2
```

واضح ان هذا التعبير يوجد متوسط عددين مخزنين في المتغيرين x و y ثم يقوم بإسناد المتوسط للمتغير z.

الأعداد Numbers

سبق ان ذكرنا انواع الأعداد الأربع (الأعداد الصحيحة و العشرية والمركبة و المنطقية)

مثال

```
>>> 5 + 2 - 3 * 2
1
>>> 5 / 2 # floating point result with normal division
2.5
>>> 5 / 2.0 # also a floating point result
2.5
>>> 5 // 2 # integer result with truncation when divided using
'//'
2
>>> 30000000000 # This would be too large to be an int in many
languages
30000000000
```

```
>>> 30000000000 * 3
90000000000
>>> 30000000000 * 3.0
90000000000.0
>>> 2.0e-8 # Scientific notation gives back a float
2e-08
>>> 3000000 * 3000000
9000000000000
>>> int(200.2)
200
>>> int(2e2)
200
>>> float(200)
200.0
```

الدوال العددية الداخلية Built-in numeric functions

يوجد عدد من الدوال العددية الموجودة في اساس بايسون مثل

abs, divmod, cmp, coerce, float, hex, int, long, max, min, oct, pow, round

دوال عددية متقدمة Advanced numeric functions

و هي موجودة في المكتبة math والتي سوف نشرحها لاحقا. هذه الدوال تشمل

acos, asin, atan, atan2, ceil, cos, cosh, e, exp, fabs, floor, fmod, frexp, hypot, ldexp, log, log10, mod, pi, pow, sin, sqrt, tan, tanh

وغيرها.

الحصول على مدخل من المستخدم Getting input from the user

تستخدم الدالة ( )input للإدخال التفاعلي المباشر.

مثال

```
>>> name = input("Name? ")
Name? Ahmad
>>> print(name)
Ahmad
>>> age = int(input("Age? "))
Age? 28
>>> print(age)
28
>>>
```

كما نرى ان هذه طريقة بسيطة للإدخال ولكن توجد مشكلة بسيطة وهي ان المدخل يجب ان يكون نص ولكي ندخل عدد نحول المدخل النصي إلى عددي كالتالي (("Age?")int وكذلك ( )float

القوائم Lists

سوف نعود لتفصيل أكثر عن القوائم حيث أن القوائم مثل الصفوف Arrays في اللغات الاخرى مثل C و Java وهي عبارة عن تجميعات مرتبة من الأشياء. ويمكن تكوين قائمة من عناصر مفصولين بفاصلة بين أقواس مربعة

مثل

```
x = [1, 2, 3]
```

لاحظ عدم الحاجة لإعلان النوع أو تحديد الحجم للقائمة حيث يمكنها النمو او الضمور في الحجم حسب الحاجة كما ان القوائم في بايسون يمكن ان تحوي عناصر مختلفة النوع لأنها تعامل على انها أشياء objects.

مثال

```
x = [2, "two", [1, 2, 3]]
```

لاحظ ان العنصر الأول عدد صحيح و الثاني نص و الثالث قائمة.

من الدوال الأساسية الداخلية للعمل على القوائم هي ( )len والتي تعطي عدد العناصر في القائمة

مثال

```
>>> x = [2, "two", [1, 2, 3]]
>>> len(x)
3
```

2 عنصر

"two" عنصر

[1, 2, 3] عنصر

المجموع 3 عناصر

مؤشرات القوائم List indices

يبدأ تأشير أي قائمة من 0 فمثلا

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[0]
'first'
>>> x[2]
'third'
```

المؤشرات سالبة الإشارة تبدأ العد من الخلف مثل

```
>>> x[-1]
'fourth'
>>> x[-2]
'third'
```

التشريح Slicing

وهو إستخراج جزء أو أجزاء من القائمة ووضعها في قائمة جديدة. لاحظ التشريح يعمل كالتالي

list[index1: index2]

يقوم بإستقطاع جميع العناصر شاملة index1 وحتى (ولكن لاتشمل) inex2 في قائمة جديدة

مثال

```
>>> x = ["first", "second", "third", "fourth"]
>>> x[1:-1]
['second', 'third']
>>> x[0:3]
['first', 'second', 'third']
>>> x[-2:-1]
['third']
>>> x[-1:2]
[]
>>> x[:3]
['first', 'second', 'third']
>>> x[2:]
['third', 'fourth']
>>> y = x[:]
>>> y[0] = '1 st'
>>> y
['1 st', 'second', 'third', 'fourth']
>>> x
['first', 'second', 'third', 'fourth']
```

تعديل القوائم Modifying lists

بإستخدام المؤشرات نستطيع تعديل قائمة مثل

```
>>> x = [1, 2, 3, 4]
>>> x[1] = "two"
>>> x
[1, 'two', 3, 4]
>>> x = [1, 2, 3, 4]
>>> x[len(x):] = [5, 6, 7]
>>> x
[1, 2, 3, 4, 5, 6, 7]
>>> x[:0] = [-1, 0]
```

```
>>> x
[-1, 0, 1, 2, 3, 4, 5, 6, 7]
>>> x[1:-1] = []
>>> x
[-1, 7]
```

هناك طريقة method لإضافة قيم لنهاية القائمة الطريقة append

```
>>> x = [1, 2, 3]
>>> x.append("four")
>>> x
[1, 2, 3, 'four']
```

نعني بطريقة أو method من المعنى المنحى شيئي object-oriented

<u>ملاحظة هامة:</u> إضافة قائمة لقائمة

```
>>> x = [1, 2, 3, 4]
>>> y = [5, 6, 7]
>>> x.append(y)
>>> x
[1, 2, 3, 4, [5, 6, 7]]
```

لاحظ ان القائمة اضيفت كعنصر واحد ولكن لنضيف عنصر بعنصر نستخدمالطريقة extend

<u>مثل</u>

```
>>> x = [1, 2, 3, 4]
>>> y = [5, 6, 7]
>>> x.extend(y)
>>> x
[1, 2, 3, 4, 5, 6, 7]
```

توجد الطريقة insert والتي تضيف عناصر لقائمة بين عنصرين سابقين مثل

```
>>> x = [1, 2, 3]
>>> x.insert(2, "hello")
>>> print(x)
[1, 2, 'hello', 3]
>>> x.insert(0, "start")
>>> print(x)
['start', 1, 2, 'hello', 3]
```

لاحظ إستخدام الطرق السابقة

```
list.append(element)
list.extend(element)
list.insert(n, element)
```

لاحظ في الطريقة insert يتم إدخال العنصر element قبل العنصر رقم n

لاحظ المثال التالي

```
>>> x = [1, 2, 3]
>>> x.insert(-1, "hello")
>>> print(x)
[1, 2, 'hello', 3]
```

الدالة del

مثال

```
>>> x = ['a', 2, 'c', 7, 9, 11]
>>> del x[1]
>>> x
['a', 'c', 7, 9, 11]
>>> del x[:2]
>>> x
[7, 9, 11]
```

العبارات

```
del list[n]
list[n:n+1] = [ ]
```

متكافئة وكذلك العبارات

```
del list[m:n[
list[m:n] = [ ]
```

الطريقة remove لاتعمل بعكس الطريقة insert حيث ان insert تقوم بوضع عنصر عند مكان معين إلا ان الطريقة remove تنظر إلى اول حدوث للعنصر و تزيله مثل

```
>>> x = [1, 2, 3, 4, 3, 5]
>>> x.remove(3)
>>> x
```

```
[1, 2, 4, 3, 5]
>>> x.remove(3)
>>> x
[1, 2, 4, 5]
>>> x.remove(3)
Traceback (innermost last):
File "<stdin>", line 1, in ?
ValueError: list.remove(x): x not in list
```

الطريقة reverse

مثال

```
>>> x = [1, 3, 5, 6, 7]
>>> x.reverse()
>>> x
[7, 6, 5, 3, 1]
```

الطريقة sort

مثال

```
>>> x = [3, 8, 4, 0, 2, 1]
>>> x.sort()
>>> x
[0, 1, 2, 3, 4, 8]

>>> x = [2, 4, 1, 3]
>>> y = x[:]
>>> y.sort()
>>> y
[1, 2, 3, 4]
>>> x
[2, 4, 1, 3]
```

الدالة sorted( )

مثال

```
>>> x = (4, 3, 1, 2)
>>> y = sorted(x)
>>> y
```

```
[1, 2, 3, 4]
```

عمليات إضافية على القوائم

العامل in (العضوية في قائمة)

مثال

```
>>> 3 in [1, 3, 4, 5]
True
>>> 3 not in [1, 3, 4, 5]
False
>>> 3 in ["one", "two", "three"]
False
>>> 3 not in ["one", "two", "three"]
True
```

إضافة قائمتين (ضم او توحيد Concatenation)

العامل +

مثال

```
>>> z = [1, 2, 3] + [4, 5]
>>> z
[1, 2, 3, 4, 5]
```

المرتبات Tuples

سبق ان قدمنا تعريف اولي للمرتبات. هنا نقدم عرض اكثر تفصيلا.

المرتبات هي كما سبق ان عرفناها هي تراكيب بيانات Data structures تشبه كثيرا القوائم و لكن لايمكن أن تعدل و تكمن أهميتها في انها تستخدم كمفاتيح للقواميس كما سنرى لاحقا حيث القوائم لايمكنها فعل ذلك.

أساسيات المرتبات

تكون او تنشأ المرتبات بإحواء عناصرها بين قوسين (و)

مثال

```
>>> x = ('a', 'b', 'c')
```

هنا كونا مرتبة تحوي 3 عناصر. بعد تكوين مرتبة تستخدم كإستخدام قائمة ويمكن أن ننسى انهم تراكيب بيانات مختلفة

امثلة

```
>>> x = ('a', 'b', 'c')
>>> x[2]
'c'
>>> x[1:]
('b', 'c')
>>> len(x)
3
>>> max(x)
'c'
>>> min(x)
'a'
>>> 5 in x
False
>>> 5 not in x
True
```

الفرق الرئيسي بين القوائم و المرتبات ان هذه الأخيرة غير قابلة للتغيرImmutable

مثال

```
>>> x[2] = 'd'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

لاحظ عند محاولة تبديل عنصر بآخر في المرتبة نتج خطأ.

يمكن تكوين مرتبة من مرتبة موجودة بإستخدام + و *

مثال

```
>>> x + x
('a', 'b', 'c', 'a', 'b', 'c')
>>> 2 * x
('a', 'b', 'c', 'a', 'b', 'c')
```

كما يمكن نسخ مرتبة كالتالي

```
>>> x[:]
('a', 'b', 'c')
>>> x * 1
('a', 'b', 'c')
>>> x + ()
('a', 'b', 'c')
```

ملاحظة هامة: المرتبة ذات العنصر الواحد يجب وضع فاصلة بعد العنصر الوحيد.

مثال

```
>>> x = 3
>>> y = 4
>>> (x + y)
7
```

تم جمع x و y

```
>>> (x + y,)
(7,)
```

وضع الفاصلة كون مرتبة بعنصر واحد وهو مجموع العددين.

لتكوين مرتبة فارغة أستخدم ().

رزم Packing و تفكيك Unpacking المرتبات

مثال

```
>>> (one, two, three, four) = (1, 2, 3, 4)
>>> one
1
>>> two
2
```

لاحظ ان بايسون يسمح بإستخدام المرتبات في الجهة اليسرى من عملية إسناد و التي تسمح لقيم في مرتبة على اليسار ان يسند إليها قيم من مرتبة على اليمين. ويمكن ان يعبر عن السابق بشكل اكثر إختصار حيث بايسون يتعرف على المرتبة حتى لوكانت بدون أقواس

```
one, two, three, four = 1, 2, 3, 4
```

هنا يقال ان القيم على اليسار قد رزمت (او حزمت) في مرتبة ثم ققكت في متغيرات في الطرف الأيمن لاحظ ان السطر السابق من الترميز يكافئ

```
one = 1
two = 2
three = 3
four = 4
```

لاحظ انه لتبديل قيم لمتغيرين كنا نستخدم الترميز

```
temp = var1
var1 = var2
var2 = temp
```

يمكن عمل ذلك بالترميز

```
var1, var2 = var2, var1
```

خاصية * في التفكيك

مثال

```
>>> x = (1, 2, 3, 4)
>>> a, b, *c = x
>>> a, b, c
(1, 2, [3, 4])
>>> a, *b, c = x
```

```
>>> a, b, c
(1, [2, 3], 4)
>>> *a, b, c = x
>>> a, b, c
([1, 2], 3, 4)
>>> a, b, c, d, *e = x
>>> a, b, c, d, e
(1, 2, 3, 4, [])
```

لاحظ ان العنصر ذا العلامة * يقوم بإحتواء أي عدد من العناصر و التي لا تتطابق مع العناصر الاخرى كما تلاحظ ايضا ان العنصر ذا العلامة * يحوي العناصر كقائمة.

الرزم و التفكيك يمكن ان يستخدم ايضا كالتالي

```
>>> [a, b] = [1, 2]
>>> [c, d] = 3, 4
>>> [e, f] = (5, 6)
>>> (g, h) = 7, 8
>>> i, j = [9, 10]
>>> k, l = (11, 12)
>>> a
1
>>> [b, c, d]
[2, 3, 4]
>>> (e, f, g)
(5, 6, 7)
>>> h, i, j, k, l
(8, 9, 10, 11, 12)
```

تذكير: [ ] قائمة و ( ) مرتبة.

التحويل بين القوائم و المرتبات :

الدوال ( )list و ( )tuple

مثال

```
>>> list((1, 2, 3, 4))
[1, 2, 3, 4]
>>> tuple([1, 2, 3, 4])
```

```
(1, 2, 3, 4)
```

لاحظ كيف تفكك الدالة list النص التالي

```
>>> list("Hello")
['H', 'e', 'l', 'l', 'o']
```

المجموعات Sets

سبق تعريفها بشكل مختصر وهنا نضيف بعض التفاصيل. المجموعة في بايسون هي تجميعة من الأشياء غير المرتبة والتي تكون أهميتها في عضوية شيئ اليها و تفرده مثل المفاتيح في القواميس والتي سنذكرها لاحقا. كما يجب لعناصر المجموعة أن تكون غير قابلة للتغير Immutable وغيرقابلة للتجزيئ Hashable وهذا يعنى أن int و float و strings و tuples يمكن ان يكونوا أعضاء في مجموعة ولكن lists و dictionaries و sets لايمكن ان يكونوا أعضاء في مجموعة.

عمليات على المجموعات

بألإضافة للعمليات in و len و إسيخدام دورة for على جميع عناصر المجموعة فللمجموعات عمليات خاصة بها

مثل

```
>>> x = set([1, 2, 3, 1, 3, 5])
>>> x
{1, 2, 3, 5}
>>> x.add(6)
>>> x
{1, 2, 3, 5, 6}
>>> x.remove(5)
>>> x
{1, 2, 3, 6}
>>> 1 in x
True
>>> 4 in x
False
>>> y = set([1, 7, 8, 9])
>>> x | y
{1, 2, 3, 6, 7, 8, 9}
>>> x & y
{1}
>>> x ^ y
```

```
{2, 3, 6, 7, 8, 9}
>>>
```

لاحظ علامة | تعني إتحاد مجموعتين و & تقاطع و ^ الفرق.

المجموعات الجامدة Frozensets

بما ان المجموعات غير immutable وغير hashable فإنه لايمكنها ان تكون أعضاء في مجموعات اخرى. لتخطي هذه العقبة يوجد نوع آخر من المجموعات وهو المجموعات المجمدة والتي تماما كالمجموعة ولكنها immutable و hashable ويمكنها ان تكون عناصر في مجموعات اخرى

مثال

```
>>> x = set([1, 2, 3, 1, 3, 5])
>>> z = frozenset(x)
>>> z
frozenset({1, 2, 3, 5})
>>> z.add(6)
Traceback (most recent call last):
File "<pyshell#79>", line 1, in <module>
z.add(6)
AttributeError: 'frozenset' object has no attribute 'add'
>>> x.add(z)
>>> x
{1, 2, 3, 5, frozenset({1, 2, 3, 5})}
```

القواميس Dictionaries

ويطلق على صفوف متقارنة Associative Arrays والتي يمكن تكوينها بإستخدام جداول التجزيئ Hash tables. القواميس مفيدة جدا كما سنرى.

يكون او يعرف القاموس بأقواس معكوفة { و }

مثال

```
>>> y = { }
```

كونا قاموس جديد فارغ و اسندناه لـ y

بعد تكوين القاموس يمكن خزن قيم فيه كما لو كان قائمة

مثال

```
>>> y[0] = 'Hello'
>>> y[1] = 'Goodbye'
```

ملاحظة: لكي نوضح الإختلاف بين القائمه و القاموس نجرب التالي

```
>>> x = [ ]
>>> x[0] = 'Hello'
Traceback (innermost last):
File "<stdin>", line 1, in ?
IndexError: list assignment index out of range
```

العمليات التالية على القواميس لايمكن إجرائها على القوائم

```
>>> y["two"] = 2
>>> y["pi"] = 3.14
>>> y["two"] * y["pi"]
6.2800000000000002
```

مؤشرات القواميس هي مفاتيح ويمكن الن تكون اعداد او نصوص او نوع من البيانات بينما مؤشرات القوائم هي اعداد صحيحة فقط.

لماذا سميت القواميس قواميس؟

القاموس هو طريقة لتطبيق mapping مجموعة من الأشياء الإختيارية لمجموعة متصلة ولكنها إختيارية من الأشياء.

مثال

```
>>> import os
>>> import utf-8
>>> english_to_arabic = {}
>>> english_to_arabic['red'] = 'احمر'
>>> english_to_arabic['blue'] = 'ازرق'
>>> english_to_arabic['green'] = 'اخضر'
>>> print("red is", english_to_arabic['red'])
red is احمر
```

المصفوفات Matrices

تمثل المصفوفة بقائمة قوائم

```
>>> matrix = [[3, 0, -2, 11], [0, 9, 0, 0], [0, 7, 0, 0], [0, 0,
0, -5]]
>>> matrix[2][3]
0
>>> matrix[0][0]
3
>>> matrix
[[3, 0, -2, 11], [0, 9, 0, 0], [0, 7, 0, 0], [0, 0, 0, -5]]
>>>
```

لاحظ:

```
element = matrix[rownum][colnum]
```

المصفوفات غير الكثيفة Sparse Matrices

وهي المصفوفات التي تحوي عناصر قليلة جدا غير صفرية وبقية العناصر كلها اصفار. تنتج مثل هذه المصفوفات كثيرا في التطبيقات العملية. مثلا في التنبؤ عن الطقس تنتج مصفوفات كبيرة جدا تحوي آلاف العناصر لموقع معين وهذا يعني ملايين عنصر ككل في المصفوفة وتحوي مثل هذه المصفوفات عدد ضخم جدا من الأصفار بل في بعض التطبيقات قد تكون الصفوفة مكونة كلها من اصفار ماعدى نسبة صغيرة من العناصر غير الصفرية. فلكي نوفر الذاكرة نقوم بتخزين مثل هذه المصفوفات بحيث نحدد العناصر الغير صفرية فقط.

من السهل التعبير عن المصفوفات غير الكثيفة بإستخدام القواميس وتستخدم المراتب كمؤشرت.

مثال

المصفوفة غير الكثيفة

```
matrix = [[3, 0, -2, 11], [0, 9, 0, 0], [0, 7, 0, 0], [0, 0, 0,
-5]]
```

لاحظ 6 عناصر من 16 عنصر غير صفرية. نعبر عن المصفوفة السابقة كالتالي

```
matrix = {(0, 0): 3, (0, 2): -2, (0, 3): 11,
(1, 1): 9, (2, 1): 7, (3, 3): -5}
```

الآن نستطيع الوصول لعنصر محدد من المصفوفة عند سطر و عمود معطاه كالتالي

```
if (rownum, colnum) in matrix:
element = matrix[(rownum, colnum)]
else:
element = 0
```

هناك طريق أكثر فعالية وهي بإستخدام طريقة get الخاصة بالقواميس كالتالي

```
element = matrix.get((rownum, colnum), 0)
```

وتقوم بإسترجاع 0 إذا لم يكن هناك مفتاح في القاموس و إلا تسترجع القيمة التابعة للمفتاح.

ملاحظة: المكتبة Numpy تحوي دوال وطرق كثيرة للتعامل مع المصفوفات.

قبل ان نبدأ الجزء المتعلق بالإستخدامات الإحصائية في بايسن نود تقديم بعض التعليمات والدوال التي تستخدم داخل الدورات.

الجمل break و continue

الجمل الخاصة break و continue يمكن ان تستخدم في قلب الدورات while و for إذا تم تنفيذ الجملة break فإنها تنهي حالا الدورة while و إذا تم تنفيذ continue فإنها تسبب في القفز عن بقية البرنامج الذي توقف لديه و يعاد تقييم الشرط مرة اخرى و تسنمر الدورة إعتياديا.

الدالة مجال The range function

ويستخدم كالتالي

مثال

```
x = [1, 3, -7, 4, 9, -5, 4]
for i in range(len(x)):
if x[i] < 0:
print("Found a negative number at index ", i)
```

فمعطى عدد n الدالة range(n) ترجع المتتابعة من الأعداد 0, 1, 2, ..., n-2, n-1 لاحظ ان هذه الدالة لاتكون قائمة من الأعداد بل ان المتتابعة الناتجة هي شيئ مجال range object.

امثلة

```
>>> list(range(3, 7))
[3, 4, 5, 6]
>>> list(range(2, 10))
[2, 3, 4, 5, 6, 7, 8, 9]
>>> list(range(5, 3))
[]

>>> list(range(0, 10, 2))
[0, 2, 4, 6, 8]
>>> list(range(5, 0, -1))
[5, 4, 3, 2, 1]
```

الإستيعاب الفهمي comprehensions للقوائم و القواميس

وهو عبارة عن دورة for في سطر واحد و الذي يكون قائمة او قاموس جديد من قائمة او قاموس آخر

مثال:

للقوائم

```
>>> x = [1, 2, 3, 4]
>>> x_squared = [item * item for item in x]
>>> x_squared
[1, 4, 9, 16]

>>> x = [1, 2, 3, 4]
>>> x_squared = [item * item for item in x if item > 2]
>>> x_squared
[9, 16]
```

للقواميس

```
>>> x = [1, 2, 3, 4]
>>> x_squared_dict = {item: item * item for item in x}
>>> x_squared_dict
{1: 1, 2: 4, 3: 9, 4: 16}
```

و الفرق بين الإثنين انه في حتالة القواميس نعين المفتاح ايضا.

العبارات Statements و القوالب Blocks و فراغات اول السطر Indentation

يستخدم في بايسون فراغات اول السطر في العبارات لتحديد حدود القوالب المختلفة (القالب هو مجموعة من اسطر الترميز المتعلقة ببعضها) وكذلك عبارات التحكم المختلفة

مثال

إزاحات صحيحة

```
>>> x = 1
>>> if x == 1:
...     y = 2
...     if v > 0:
...         z = 2
...         v = 0
...
>>> x = 2
```

إزاحة سطر z = 2 غير صحيحة

```
>>> x = 1
>>> if x == 1:
          y = 2
       z = 2
File "<stdin>", line 3
z = 2
^
IndentationError: unindent does not match any outer indentation
level
```

لاحظ رسالة الخطأ.

عودة للدوال:

سوف نعود لتعريف الدوال و لإستخداماتها بالتفصيل. التركيب اللغوي الأساسي للدالة في بايسون هو

```
def name(parameter1, parameter2, . . .):
    body
```

مثال

دالة لحساب المضروب للعدد n أي !n

```
>>> def fact(n):
...     """Return the factorial of the given number."""
...     r = 1
...     while n > 0:
...         r = r * n
...         n = n - 1
...     return r
...
```

ملاحظات:

1) العلامات الثلاثة " " " توضع امام التعليق الذي قد يمتد على اكثر من سطر

2) العلامات الثلاثة . . . طريقة بايسون على انه ينتظر المزيد من المدخلات لإتمام تعريف المطلوب

3) إدخال Enter في سطر مباشرة بعد . . . يخبر بايسون بأن التعريف أكتمل.

خيارات معالم الدالة Function parameter options

المعالم الموضعية Positional parameters

اسهل طريقة لإمرار معالم لدالة هي بالموضع ففي السطر الأول نعين تعريف أسماء المتغيرات لكل معلم وعندما ننادي الدالة فإن المعالم المستخدمة في النداء تطابق مع معالم الدالة حسب موضعها و ترتيبها.

مثال

الدالة التالية تحسب القوة y للمتغير x

```
>>> def power(x, y):
...     r = 1
...     while y > 0:
...         r = r * x
...         y = y - 1
...     return r
...
>>> power(3, 3)
27
```

هذه الطريق تتطلب أن يكون عدد المعالم المستخدمة في نداء الدالة تتطابق تماما مع عدد المعالم في تعريف الدالة وإلا ينتج خطأ

مثال

```
>>> power(3)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: power() takes exactly 2 positional arguments (1 given)
>>>
```

المعالم الإفتراضية Default values

معالم الدالة يمكن ان تعطى قيم إفتراضية و التي تحدد مسبقا في تعريف الدالة والتي يمكن تبديلها بقيم اخرى عند نداء الدالة.

مثال

```
>>> def power(x, y=2):
...     r = 1
...     while y > 0:
...         r = r * x
...         y = y - 1
...     return r
...

>>> power(3, 3)
27
>>> power(3)
9
```

تمرير حجج arguments بواسطة اسماء المعالم

نستطيع تمرير حجج للدالة بإستخدام إسم المعلم المناسب في الدالة بدلا من موقع المعلم متابعين المثال السابق

مثال

```
>>> power(2, 3)
8
>>> power(3, 2)
9
>>> power(y=2, x=3)
9
```

المتغيرات المحلية local و العامة global

نعود لمثال حساب المضروب !n

```
def fact(n):
    """Return the factorial of the given number."""
    r = 1
```

```
    while n > 0:
        r = r * n
        n = n - 1
    return r
```

المتغيرات r و n تعتبر محلية local لأي نداء ما للدالة fact و تتغير قيمهم عند إجراء نداء للدالة وليس لهم اي تأثير على اي متغير خارج الدالة فأي متغير في قائمة معالم الدالة و اي متغير تكون داخل الدالة مثل r = 1 يعتبر محلي في الدالة.

نستطيع ان نجعل متغير عام global بإعلان ذلك قبل إستخدامه وذلك بإستخدام العبارة global.

المتغيرات العامة والتي تكون خارج الدالة يمكن للدالة الوصول اليها وتغييرها وكذلك الدوال الاخرى يمكنها ذلك.

<u>مثال</u>

```
>>> def fun():
... global a
... a = 1
... b = 2
...
```

هنا نعرف دالة والتي تعامل a على انه متغير عام و b متغير محلي وتحاول تعديل كليهما.

الآن لنجرب هذه الدالة

```
>>> a = "one"
>>> b = "two"
>>> fun()
>>> a
1
>>> b
'two'
```

تعابير لمدا lambda exressions

و تعرف دوال قصيرة لها التركيب

```
lambda parameter1, parameter2, . . .: expression
```

مثال

```
lambda x: x==" ", A, B
lambda x:2*x, xrange(5)
lambda x,y:(x, y, x*y)
```

البرامج الجزئية او الموديول Modules

و تستخدم لتنظيم برامج بايسون و إضافة مقدرات للبرامج عند الحاجة إليهافمثلا مكتبة بايسون القياسية Python standard library مقسمة إلى موديولات لتسهيل إدارتها و إستخدامها.

والتعريف الرسمي للموديول هو عبارة عن ملف يحوي ترميز. وهو يعرف مجموعة من الدوال او اي اشياء.

فضائآت الأسماء namespaces

وهي قاموس من المعرفات لمجموعة من الترميز او دالة او فئة او موديول وهكذا.

لكل موديول فضاء أسماء والذي يمنع تضارب الأسماء بين المديولات.

مثال

سوف نكتب موديول بسيط

```
"""mymath - our example math module"""
pi = 3.14159
def area(r):
    """area(r):return the area of a circle with radius
       r."""
    global pi
    return(pi * r * r)
```

خزن هذا الترميز في ملف mymath.py في مجلد python . قم بإجراء التالي

```
>>> pi
Traceback (innermost last):
File "<stdin>", line 1, in ?
```

```
NameError: name 'pi' is not defined
>>> area(2)
Traceback (innermost last):
File "<stdin>", line 1, in ?
NameError: name 'area' is not defined
```

لاحظ رسائل الأخطاء pi غير معرفة و الدالة area غير معرفة

الآن قم بالتالي

```
>>> import mymath
>>> pi
Traceback (innermost last):
File "<stdin>", line 1, in ?
NameError: name 'pi' is not defined
>>> mymath.pi
3.1415899999999999
>>> mymath.area(2)
12.56636
>>> mymath.__doc__
'mymath - our example math module'
>>> mymath.area.__doc__
'area(r): return the area of a circle with radius r.'
```

ملاحظات:

1) أحضرنا تعريف pi و area من mymath.py مستخدمين العبارة import

2) للوصول لتعريف وإستخدام pi نضيف في البداية prepending إسم الموديول الذي يحوي التعريف أي mymath.pi (قد نسمي pi صفة attribute للموديول mymath)

3) التعاريف داخل موديول تستخدم تعاريف اخرى داخل نفس الموديول بدون إضافة في البداية لإسم الموديول مثل الدالة mymath.area استخدمت mymath.pi فقط كـ pi .

ايضا يمكننا السؤال عن شيئ معين من موديول و إجتلابه import بطريقة يمكن إستخدامه بدون إضافة بداية إسم المديول له

مثال

```
>>> from mymath import pi
>>> pi
3.1415899999999999
>>> area(2)
Traceback (innermost last):
File "<stdin>", line 1, in ?
NameError: name 'area' is not defined
```

لاحظ أن pi اصبح الوصول إليه مباشرة لأننا طلبناه للإستخدام مخصوصا بالعبارة

```
from module import name
```

لاحظ ان الدالة area لم تجتلب صراحة ولهذا عند محاولة إستخدامها نتج خطأ. لكي نستخدمها يجب إستخدامها كالتالي mymath.area()

العبارة import

ولها ثلاثة أشكال

(1

```
import modulename
```

يستخدم هذا الشكل لجلب جميع التعاريف و الدوال في الموديول المسمى ولكي نستخدم أي تعريف او دالة داخل الموديول يجب إضافة إسم المديول في اول إسم التعريف أو الدالة حتى نستطيع إستخدامها.

(2

```
from modulename import name1, name2, name3, . . .
```

هذا الشكل يسمح بجلب اسماء معينة من الموديول ويمكن إستخدام هذه الأسماء مباشرة بدون إضافة إسم الموديول إليها.

(3

```
from modulename import *
```

الشكل هذا يعني جلب جميع الدوال و التعاريف من الموديول وإستخدامها مباشرة بدون إضافة إسم الموديول قبلها.

ملاحظة هامة: الشكل الأخير قد يسبب في توقف الحاسب وذلك حسب الذاكرة المتاحة لأنه قد يقوم بجلب مكتبة ضخمة جدا بكل دوالها والتي قد لانحتاج لمعظمها.

ممر بحث المديول module search path

لكي نحدد ممر بحث الموديولات نعمل التالي

```
>>> import sys
>>> sys.path
['', 'C:\\windows\\SYSTEM32\\python34.zip',
'C:\\Python34\\DLLs', 'C:\\Python34\\lib', 'C:\\Python34',
'C:\\Python34\\lib\\site-packages']
>>>
```

وهذا سرد لجميع المجلدات في ممر البحث.

ملاحظة: اهمية معرفة ممر البحث هو عند كتابتنا لبرنامج بلغة بايسون نخزن هذا البرنامج في مجلد يكون في ممر البحث حتى يستطيع بايسون إيجادة و إجرائه.

مجلد آخر مهم هو المجلد الذي يحوي برامج نظام بايسون ويمكن معرفته كالتالي

```
>>> sys.prefix
'C:\\Python34'
>>> sys.argv
['']
>>>
```

نظام الملفات filesystem

مجلد العمل الحالي

```
>>> import os
>>> os.getcwd()
'C:\\Windows\\System32'
```

لمعرفة محتوى مجلد العمل الحالي

```
>>> os.listdir(os.curdir)
```

يمكن تغيير مجلد العمل كالتالي

```
>>> os.chdir(folder)
>>> os.getcwd()
```

البحث عن مجلدات و ملفات

نستخدم الدوال

```
os.path.exists( )
os.path.isdir( )
os.path.isfile( )
```

```
>>> import os
>>> os.path.exists('C:\\Doc..\\amb\\My Documents')
True
>>> os.path.exists('C:\\Doc..\\amb\\My..\\Letter.doc')
False
>>> os.path.exists('C:\\Doc..\\amb\\\My..\\ljsljkflkjs')
False
>>> os.path.isdir('C:\\Doc..\\amb\\My Documents')
True
>>> os.path.isfile('C:\\Doc..\\amb\\My Documents')
False
>>> os.path.isdir('C:\\Doc..\\amb\\My..\\Letter.doc')
False
>>> os.path.isfile('C:\\Doc..\\amb\\My..\\Letter.doc')
```

```
False
>>>
```

تغيير مجلد العمل وإضافة مجلدات لمسار لبحث

```
>>> os.chdir(os.path.join('C:'))
>>> os.listdir(os.curdir)
```

تغيير إسم ملف أو مجلد

```
>>> os.rename('registry.bkp', 'registry.bkp.old')
```

إزالة ملف

```
os.remove('book1.doc.tmp')
```

تكوين مجلد

```
os.makedirs('mydir')
```

القرائة و الكتابة من ملف

فتح ملفات و أشياء الملف file objects

تستخدم الدالة open لفتح الملفات

```
mypyfile = open('myfile', 'r')
line = mypyfile.readline()
```

السطر الأول فتحنا الملف myfile للقرائة r واسندناه للإسم mypyfile

السطر الثاني يتم قرائة سطر من mypyfile ووضعه في الإسم line

لاحظ ان الدالة open لاتقوم بقرائة اي شيئ بل تقوم بإرجاع شيئ يسمى file object والذي يستخدم للوصول إلى الملف للقرائة منه. الشيئ ملف file object يتابع ملف وكم من الملف تمت قرائتة او الكتابة إليه. كل مدخلات ومخرجات الملفات files I/O في بايسون تتم بإستخدام شيئ ملف بدلا من أسماء ملفات.

لاحظ ان النداء الأول للطريقة readline تعيد السطر الأول في الشيئ ملف (من بداية السطر الأول وحتى رمز سطر جديد newline و النداء الثاني تعيد السطر الثاني وهكذا.

يجب ان يتم قفل الملف بعد الإنتهاء من إستخدامه العبارة

```
mypyfile.close()
```

فتح ملف للكتابة

```
file_object = open("myfile", 'w')
file_object.write(‘‘Hello, World\n’’)
file_object.close()
```

الإدخال و الإخراج من النهاية الطرفية بواسطة الدالة ( )input

```
>>> x = input("enter file name to use: ")
enter file name to use: myfile
>>> x
'myfile'
>>> x = int(input("enter your number: "))
enter your number: 39
>>> x
39
```

المكتبة العددية Numpy

تستخدم المكتبة numpy للمعالجات العددية مثل الصفوف arry و التأشير indexing والعمليات الرياضية و في تحميل و حفظ البيانات.

ndarry

وهي تشبه القوائم ماعدى انها تستخدم نوع واحد من العناصر في اي عامود.

مثال على إستخدام numpy

سوف نقوم بحل نظام من المعادلات

$$3x + 6y - 5z = 12$$
$$x - 3y + 2z = -2$$
$$5x - y + 4z = 10$$

والذي يمثل بالشكل المصفوفي

$$\begin{bmatrix} 3 & 6 & -5 \\ 1 & -3 & 2 \\ 5 & -1 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 12 \\ -2 \\ 10 \end{bmatrix}$$

والذي يكتب على شكل $\mathbf{AX = B}$ و الحل يكون $\mathbf{X = A^{-1}B}$ ويعبر عنة بلغة بايسون

```
import numpy as np
# Defining the matrices
A = np.matrix([[3, 6, -5],
               [1, -3, 2],
               [5, -1, 4]])
B = np.matrix([[12],
               [-2],
               [10]])
# Solving for the variables, where we invert A
X = A ** (-1) * B
print(X)
matrix([[ 1.75],
        [ 1.75],
```

```
       [ 0.75]])
```

ملاحظات:

في العبارة import numpy as np نقوم بعد تحميل numpy بإعطائة إسم بديل مختصر لسهولة النداء فبدل ان نستخدم الإسم الطويل numpy.matrix نستخدم np.matrix .

طريقة أخرى لحل المثال السابق و أكثر فعالية

```
import numpy as np
a = np.array([[3, 6, -5],
              [1, -3, 2],
              [5, -1, 4]])
# Defining the array
b = np.array([12, -2, 10])
# Solving for the variables, where we invert A
x = np.linalg.inv(a).dot(b)
print(x)
# array([ 1.75, 1.75, 0.75])
```

هنا أستخدمنا طريقة عكس المصفوفة inv والتي هي طريقة من linalg والتابعة للمكتبة numpy وذلك بالنداء np.linalg.inv( ). الدالة dot( ) تعني الضرب المصفوفي.

مثال

بعض الدوال الإحصائية في numpy

```
>>> import numpy as np
>>> x = np.random.randn(1000)
>>> mean = x.mean()
>>> std = x.std()
>>> var = x.var()
>>> mean
0.00083263604454047525
>>> std
0.97761359387048452
>>> var
```

```
0.9557283389203646 8
>>>
```

المكتبة العلمية Scipy

تسنخدم المكتبة scipy للدوال الرياضية المتقدمة مثل الأمثلية و الإمتداد و التكامل و التجميع و الإحصاء الخ

المكتبة الجزئية Scipy stats

أمثلة

نستورد numpy و نستخدم الدوال المبنة داخله

```
>>> import numpy as np
>>> x = np.random.randn(1000)
>>> mean = x.mean()
>>> std = x.std()
>>> var = x.var()
>>> mean
0.00083263604454047525
>>> std
0.97761359387048452
>>> var
0.95572833892036468
```

الآن نستورد scipy ومنه نستورد stats

```
>>> import scipy as sp
>>> from scipy import stats
```

لتوليد بيانات طبيعية و إجراء إختبارات نحتاج المكتبة norm

```
>>> from scipy.stats import norm
>>> print(norm.a,norm.b)
(-inf, inf)
```

نهايات التوزيع الطبيعي

```
>>> y = norm.rvs(size = 1000)
```

تم توليد 1000 قيمة من التوزيع الطبيعي القياسي و اسندت للمتغير y

```
>>> stats.describe(y)
```

```
DescribeResult(nobs=1000L,
minmax=(-3.6219575282524796, 3.2303851667890995), mean=-
0.019475394704845807, variance=1.0721232210417662,
skewness=0.047585375503389454, kurtosis=0.08902490659323314)
>>>
```

الدالة describe( ) تعطي الإحصاء الوصفي

```
>>> np.random.seed(282629734)
>>> z = stats.t.rvs(10, size=1000)
```

حددنا قيمة اولية لمولد الأعداد العشوائية ثم تم توليد 1000 قيمة من توزيع t بدرجات حرية 10

```
>>> z.max()
5.2632773298071651
>>> z.min()
-3.7897557242248197
>>> z.std()
1.1353386552387288
```

إختبار t لعينة واحدة

```
>>> m = z.mean()
>>> stats.ttest_1samp(z, m)
(0.0, 1.0)
```

قيمة الإحصائة 0 والـ p-value = 1.0 وهذا متوقع لأن m هي متوسط البيانات

إختبار K-S

```
>>> stats.kstest(z, 't', (10,))
(0.01597048315646199, 0.96063581462417624)
```

أختبرنا فيما إذا كانت z لها توزيع t بدرجات حرية 10 فأعطى إحصائة ...0.0159 و p-value = 0.960..

```
>>> stats.kstest(z,'norm')
(0.028315793880653972, 0.39485757274432154)
```

إختبار فيما إذا كانت البيانات المخزنة في z طبيعية و النتيجة (test statistics, p-value)

```
>>> stats.skewtest(z)
(2.7846411274334515, 0.0053586995736829915)
```

إختبار التفلطح skewness

```
>>> stats.kurtosistest(z)
(4.7565018293119161, 1.9697645500378176e-06)
```

إختبار التحدب kurtosis

```
>>> stats.normaltest(z)
(30.378535860841446, 2.5315397653922047e-07)
```

إختبار الطبيعية

```
>>> stats.normaltest(stats.t.rvs(10, size=100))
(2.4053653542879436, 0.30038728792621194)
```

في هذا المثال نختبر طبيعية بيانات مولدة لها توزيع t حجمها 1000 و درجات حرية 10

```
>>> stats.normaltest(stats.norm.rvs(size=1000))
(0.42443841163790574, 0.80878738723190435)
```

في هذا المثال نختبر طبيعية بيانات مولدة لها توزيع طبيعي قياسي حجمها 1000

في المثال التالي نولد عينات من توزيع طبيعي بمتوسط 5 و إنحراف معياري 10 حجمها 500

```
>>> rvs1 = stats.norm.rvs(loc=5, scale=10, size=500)
>>> rvs2 = stats.norm.rvs(loc=5, scale=10, size=500)
>>> stats.ttest_ind(rvs1, rvs2)
(1.1770563675812491, 0.23945365099781088)
```

قمنا بإختبار t لعينتين مستقلة. نولد الآن عينة ثالثة بمتوسط 8 ونفس الإنحراف المعياري.

```
>>> rvs3 = stats.norm.rvs(loc=8, scale=10, size=500)
>>> stats.ttest_ind(rvs1, rvs3)
(-4.479765082832647, 8.3358279001851181e-06)
>>>
```

إختبار K-S لعينتين

```
>>> stats.ks_2samp(rvs1, rvs2)
(0.057999999999999996, 0.35990286596479987)
>>> stats.ks_2samp(rvs1, rvs3)
```

```
(0.11199999999999999, 0.0034136307392505553)
>>>
```

مثال على الإنحدار Linear Regression

سوف نولد بيانات صناعية (محاكاة) ونجري تطبيق معادلة إنحدار عليها

```
>>> import numpy as np
>>> from scipy.optimize import curve_fit
```

نعرف دالة func لتحديد شكل المعادلة

```
>>> def func(x, a, b):
...     return a * x + b
...
```

نولد بيانات غير عشوائية (نظيفة)

```
>>> x = np.linspace(0, 10, 100)
>>> y = func(x, 1, 2)
```

نضيف ضجة (عشوائية) للبيانات

```
>>> yn = y + 0.9 * np.random.normal(size=len(x))
```

نستخدم دالة تطبيق المنحنيات curve_fit على البيانات

```
>>> popt, pcov = curve_fit(func, x, yn)
```

القيمة popt تعطي أفضل قيم تطبيق لمعالم النموذج (الدالة) المعطاة

```
>>> print(popt)
[ 1.02381944  1.98481639]
```

لاحظ مدى التطابق (القيم الأصلية هي 1, 2)

```
>>> print(pcov)
[[ 0.00075571 -0.00377854]
 [-0.00377854  0.02531748]]
```

pcov مصفوفة التباين-التغاير للمعالم. ايضا تؤكد جودة التطبيق.

```
>>>
```

العمل مع بيانات حقيقية

إدخال بيانات وإجراء بعض العمليات

```
>>> import numpy as np

>>> import scipy as sc

>>> x = np.array([474.688, 506.445, 524.081, 530.672, 530.869,
566.984, 582.311, 582.940, 603.574, 792.358])

>>> x

array([ 474.688,  506.445,  524.081,  530.672,  530.869,
566.984, 582.311,  582.94 ,  603.574,  792.358])

>>> from scipy import stats

>>> sc.stats.describe(x)

DescribeResult(nobs=10L, minmax=(474.68799999999999,
792.35799999999995), mean=569.49220000000003,
variance=7689.5458092888857, skewness=1.7024129743728933,
kurtosis=2.3738129817185847)

>>>
```

قرائة ملفات القيم المفصولة بفاصلة Reading csv files

```
>>> import pandas as pn

>>> from pandas import *

>>> iris = read_csv("C:/Users/ibin/Desktop/iris.csv")

>>> print iris

     sepal_length  sepal_width  petal_length  petal_width  name

0      5.1          3.5           1.4          0.2     setosa
```

```
1      4.9          3.0           1.4          0.2     setosa
2      4.7          3.2           1.3          0.2     setosa
. . .
>>> print iris["name"]
0     setosa
1     setosa
2     setosa
. . .


>>> print iris["name"][12]
setosa
>>> print iris["sepal_length"][12]
4.8
>>> print iris[iris["sepal_length"]> 5.0]
sepal_length  sepal_width  petal_length petal_width  name
0      5.1          3.5           1.4          0.2      setosa
5      5.4          3.9           1.7          0.4      setosa
10     5.4          3.7           1.5          0.2      setosa
14     5.8          4.0           1.2          0.2      setosa
15     5.7          4.4           1.5          0.4      setosa
. . .
>>> print iris[iris["sepal_length"]< 5.0]
  sepal_length sepal_width petal_length petal_width  name
```

```
1     4.9          3.0          1.4          0.2    setosa
2     4.7          3.2          1.3          0.2    setosa
3     4.6          3.1          1.5          0.2    setosa
6     4.6          3.4          1.4          0.3    setosa
. . .
>>> print iris["sepal_length"].mean()
5.84333333333
>>> import scipy
>>> from scipy import stats
>>> print scipy.stats.sem(iris["sepal_length"])
0.0676113162276

>>> print iris["sepal_length"].std()
0.828066127978
>>> print iris["sepal_length"].var()
0.685693512304
>>> print iris["sepal_length"].median()
5.8
>>> print iris["sepal_length"].mode()
0    5
>>> print iris["sepal_length"].sem()
0.0676113162276
```

```
>>> f_val, p_val = stats.f_oneway(iris.sepal_length,
iris.sepal_width)
>>> f_val
1335.7678308241761
>>> p_val
3.9878381148481482e-112
>>> x = iris.sepal_length
>>> y = iris.sepal_width
>>> z = iris.petal_length
>>> w = iris.petal_width
>>> f_val, p_val = stats.f_oneway(x, y, z, w)
>>> f_val
483.57128302425951
>>> p_val
3.4996987081941695e-159
>>>
```

مثال آخر

```
>>> import pandas
>>> data =
pandas.read_csv('C:/Users/ibin/Desktop/brain_size.csv', sep=';',
na_values=".")
>>> print data
Unnamed: 0  Gender  FSIQ  VIQ  PIQ  Weight  Height  MRI_Count\t
```

```
0        1  Female   133  132  124     118    64.5       816932

1        2    Male   140  150  124     NaN    72.5      1001121

2        3    Male   139  123  150     143    73.3      1038437

...

>>> print data ['Gender']

0     Female

1       Male

...

38      Male

39      Male

>>> gender_data = data.groupby('Gender')

>>> print gender_data.mean()

Unnamed: 0   FSIQ    VIQ    PIQ     Weight    Height MRI_Count\t

Gender

Female  19.65 111.9 109.45 110.45 137.200000 65.765000
862654.6

Male    21.35 115.0 115.25 111.60 166.444444 71.431579
954855.4

>>> for name, value in gender_data['VIQ']:

     print name, np.mean(value)

Female 109.45

Male 115.25

>>> data[data['Gender'] == 'Female']['VIQ'].mean()
```

```
109.45
>>> from pandas.tools import plotting
>>> plotting.scatter_matrix(data[['PIQ', 'VIQ', 'FSIQ']])
>>> from scipy import stats
>>> stats.ttest_1samp(data['VIQ'], 0)
>>> female_viq = data[data['Gender'] == 'Female']['VIQ']
>>> male_viq = data[data['Gender'] == 'Male']['VIQ']
>>> stats.ttest_ind(female_viq, male_viq)
>>> stats.ttest_ind(data['FSIQ'], data['PIQ'])
>>> stats.ttest_rel(data['FSIQ'], data['PIQ'])
>>> stats.ttest_1samp(data['FSIQ'] - data['PIQ'], 0)
>>> from scipy import stats
>>> import numpy as np
>>> x = [10, 11, 13, 9, 7, 12, 12, 9, 10, 12]
>>> y = [13, 21, 5, 10, 8, 14, 10, 12, 7, 15]
slope, intercept, r_value, p_value, std_err =
stats.linregress(x,y)
# To get coefficient of determination (r_squared)
>>> print "r-squared:", r_value**2
r-squared: 0.15286643777
```

المكتبة pandas

```
>>> import pandas
>>> data = pandas.read_csv('examples/brain_size.csv', sep=';',
na_values=".")
>>> print data
 Unnamed: 0  Gender  FSIQ  VIQ  PIQ  Weight  Height  MRI_Count
0          1  Female   133  132  124     118    64.5     816932
1          2    Male   140  150  124     NaN    72.5    1001121
2          3    Male   139  123  150     143    73.3    1038437
. . .
>>> print data['Gender']
0     Female
1       Male
2       Male
3       Male
4     Female
...
>>> gender_data = data.groupby('Gender')
>>> print gender_data.mean()
Unnamed: 0   FSIQ    VIQ    PIQ     Weight    Height  MRI_Count
Gender
Female  19.65 111.9 109.45 110.45 137.200000 65.765000  862654.6
```

```
Male    21.35 115.0 115.25 111.60 166.444444 71.431579  954855.4
>>> for name, value in gender_data['VIQ']:
...     print name, np.mean(value)
Female 109.45
Male 115.25
>>> data[data['Gender'] == 'Female']['VIQ'].mean()
109.45
```

فتح ملف تفاعليا:

```
>>> import numpy as np
>>> import scipy as sc
>>> import pandas as pd
>>> from pandas import *
>>> import Tkinter,tkFileDialog
>>> ## C:\Users\amb\Desktop\R python course\hsb2.csv
>>> hsb = tkFileDialog.askopenfile()
>>> hsb2 = read_csv(hsb)
>>> hsb.close()
>>> hsb2
      id  female  race  ses  schtyp  prog  read  write  math  science  socst
0     70       0     4    1       1     1    57     52    41       47     57
1    121       1     4    2       1     3    68     59    53       63     61

198  118       1     4    2       1     1    55     62    58       58     61
199  137       1     4    3       1     2    63     65    65       53     61
```

```
>>> hsb2["race"]
0     4
1     4


198    4
199    4
Name: race, Length: 200, dtype: int64
>>> x = hsb2["read"]
>>> y = hsb2["write"]
>>> z = hsb2["math"]
>>> w = hsb2['science']
>>> from scipy import stats
>>> stats.ttest_ind(x,y)
(-0.551990645527904, 0.58126457287969857)
>>> stats.ttest_ind(x,z)
(-0.42257874015791508, 0.67283084594388431)
>>> print hsb2.mean()
id         100.500
female       0.545
race         3.430
ses          2.055
```

```
schtyp       1.160
prog         2.025
read        52.230
write       52.775
math        52.645
science     51.850
socst       52.405
dtype: float64
>>> hsb2.describe()
                id     female        race         ses      schtyp        prog  \
count  200.000000  200.00000  200.000000  200.000000  200.000000  200.000000
mean   100.500000    0.54500    3.430000    2.055000    1.160000    2.025000
std     57.879185    0.49922    1.039472    0.724291    0.367526    0.690477
min      1.000000    0.00000    1.000000    1.000000    1.000000    1.000000
25%     50.750000    0.00000    3.000000    2.000000    1.000000    2.000000
50%    100.500000    1.00000    4.000000    2.000000    1.000000    2.000000
75%    150.250000    1.00000    4.000000    3.000000    1.000000    2.250000
max    200.000000    1.00000    4.000000    3.000000    2.000000    3.000000

             read       write        math     science       socst
count  200.000000  200.000000  200.000000  200.000000  200.000000
mean    52.230000   52.775000   52.645000   51.850000   52.405000
std     10.252937    9.478586    9.368448    9.900891   10.735793
min     28.000000   31.000000   33.000000   26.000000   26.000000
25%     44.000000   45.750000   45.000000   44.000000   46.000000
50%     50.000000   54.000000   52.000000   53.000000   52.000000
```

```
75%     60.000000   60.000000   59.000000   58.000000   61.000000
max     76.000000   67.000000   75.000000   74.000000   71.000000
>>> help(ols)
>>> result = ols(y=y, x=x)
>>> result
---------Summary of Regression Analysis--------------------

Formula: Y ~ <x> + <intercept>

Number of Observations:         200
Number of Degrees of Freedom:   2

R-squared:         0.3561
Adj R-squared:     0.3529

Rmse:              7.6249

F-stat (1, 198):   109.5213, p-value:     0.0000

Degrees of Freedom: model 1, resid 198

--------Summary of Estimated Coefficients----------------
Variable   Coef  Std Err  t-stat p-value CI 2.5%   CI 97.5%
```

```
-------------------------------------------------------------

    x       0.5517  0.0527   10.47  0.0000   0.4484  0.6550

intercept 23.9594  2.8057    8.54  0.0000  18.4602 29.4587

------------End of Summary-----------------------------------
```

التحليل الإحصائي في بايسون Statistical Analysis in Python

سوف نستعرض بعض الطرق لتحليل البيانات مستخدمين مكتبات numpy و scipy و pandas. الميزة المفيدة في pandas هي في توفير إطارات بيانات DataFrame تشبه R وهي قوية وفعالة وتساعد على تحليل كميات كبيرة من البيانات في وقت واحد.

```
>>> import numpy as np
>>> import scipy as sp
>>> import pandas as pd
>>> from scipy import stats
>>> from pandas import *
>>> experimentDF =
read_csv("C:\Users\ibin\Desktop\parasite_data.csv",
na_values=[""])
>>> print experimentDF
     Virulence  Replicate  ShannonDiversity
0          0.5          1          0.059262
1          0.5          2          1.093600
. . .
348        NaN         49          0.383512
349        NaN         50          0.511329

[350 rows x 3 columns]
>>> print experimentDF["Virulence"]
0     0.5
```

```
1     0.5
. . .
348   NaN
349   NaN
Name: Virulence, Length: 350, dtype: float64
>>> print experimentDF["ShannonDiversity"][12]
1.58981
>>> print experimentDF[experimentDF["ShannonDiversity"] > 2.0]
     Virulence  Replicate  ShannonDiversity
8          0.5          9           2.04768
89         0.6         40           2.01066
92         0.6         43           2.90081
. . .
237        0.9         38           2.16535
243        0.9         44           2.17578
251        1.0          2           2.16044
>>> print experimentDF[np.isnan(experimentDF["Virulence"])]
     Virulence  Replicate  ShannonDiversity
300        NaN          1          0.000000
301        NaN          2          0.000000
. . .
348        NaN         49          0.383512
349        NaN         50          0.511329
```

```
>>> print "Mean virulence across all treatments:",
experimentDF["Virulence"].mean()

Mean virulence across all treatments: 0.75

>>> from scipy.stats import sem

>>> print "Mean virulence across all treatments:",
sem(experimentDF["Virulence"])

Mean virulence across all treatments: nan

>>> print experimentDF.dropna()

     Virulence  Replicate  ShannonDiversity

0          0.5          1          0.059262

1          0.5          2          1.093600

. . .

298        1.0         49          0.000000

299        1.0         50          0.000000


[300 rows x 3 columns]

>>> print experimentDF.fillna(0.0)["Virulence"]

0     0.5

1     0.5

. . .

348    0

349    0

Name: Virulence, Length: 350, dtype: float64
```

```
>>> print ("Mean virulence across all treatments w/ dropped
NaN:",
        experimentDF["Virulence"].dropna().mean())


('Mean virulence across all treatments w/ dropped NaN:', 0.75)
>>> print ("Mean virulence across all treatments w/ filled
NaN:",
        experimentDF.fillna(0.0)["Virulence"].mean())
('Mean virulence across all treatments w/ filled NaN:',
0.6428571428571429)
>>> print ("Mean Shannon Diversity w/ 0.8 Parasite Virulence =",
        experimentDF[experimentDF["Virulence"] ==
0.8]["ShannonDiversity"].mean())


('Mean Shannon Diversity w/ 0.8 Parasite Virulence =',
1.2691338188000001)
>>> print ("Variance in Shannon Diversity w/ 0.8 Parasite
Virulence =",
        experimentDF[experimentDF["Virulence"] ==
0.8]["ShannonDiversity"].var())
('Variance in Shannon Diversity w/ 0.8 Parasite Virulence =',
0.61103843331267282)
>>> print ("SEM of Shannon Diversity w/ 0.8 Parasite Virulence
=",
        sem(experimentDF[experimentDF["Virulence"] ==
0.8]["ShannonDiversity"]))
```

```
('SEM of Shannon Diversity w/ 0.8 Parasite Virulence =',
0.11054758552882764)

>>> treatment1 = experimentDF[experimentDF["Virulence"] ==
0.5]["ShannonDiversity"]

>>> treatment2 = experimentDF[experimentDF["Virulence"] ==
0.8]["ShannonDiversity"]

>>> print "Data set 1:\n", treatment1

Data set 1:

0     0.059262

1     1.093600

. . .

49    0.000000

Name: ShannonDiversity, dtype: float64

>>> print "Data set 2:\n", treatment2

Data set 2:

150    1.433800

151    2.079700

. . .

198    0.000000

199    1.990900

Name: ShannonDiversity, dtype: float64

>>> treatment3 = experimentDF[experimentDF["Virulence"] ==
0.8]["ShannonDiversity"]
```

```
>>> treatment4 = experimentDF[experimentDF["Virulence"] ==
0.9]["ShannonDiversity"]

>>> type(experimentDF)

<class 'pandas.core.frame.DataFrame'>
```

حالة دراسة:

```
Python 2.7.9 |Anaconda 2.2.0 (64-bit)| (default, Dec 18 2014,
16:57:52) [MSC v.1500 64 bit (AMD64)] on win32

Type "copyright", "credits" or "license()" for more information.

>>> import scipy as sp

>>> import numpy as np

>>> from scipy import stats

>>> s = sp.randn(100)

>>> s.mean()

0.10683297347180584

>>> s.min()

-2.470091937455591

>>> s.max()

2.8750247220873035

>>> s.var()

1.1162508518405709

>>> s.std()

1.0565277335879881

>>> sp.mean(s)
```

```
0.10683297347180584
>>> sp.var(s)
1.1162508518405709
>>> sp.std(s)
1.0565277335879881
>>> sp.median(s)
0.16752620985057426
>>> n, min_max, mean, var, skew, kurt = stats.describe(s)
>>> n
100L
>>> min_max
(-2.470091937455591, 2.8750247220873035)
>>> mean
0.10683297347180584
>>> var
1.1275261129702736
>>> skew
-0.19503710136976135
>>> kurt
-0.24255807597047108
>>> sampl1 = stats.norm.rvs(loc=3.5, scale=2.0, size=100)
>>> stats.describe(sampl1)
DescribeResult(
```

```
nobs=100L,
minmax=(-1.0780854373372888, 7.5579278512797066),
mean=3.4292092298254819,
variance=3.3775667140475258,
skewness=-0.1179983952230429,
kurtosis=-0.12496640095573852)
>>>
```

ملحق:

مكونات المكتبة الجزئية stats في المكتبة scipy

stats

Statistical functions -- This module contains a large number of probability distributions as well as a growing library of statistical functions.

Access via:

```
>>> from scipy import stats
```

Also see: http://docs.scipy.org/doc/scipy/reference/tutorial/stats.html

For an overview, in IPython, do:

```
>>> from scipy import stats

>>> help(stats)
```

Each included distribution is an instance of the class rv_continous: For each given name the following methods are available:

- rv_continuous([momtype, a, b, xtol, ...]) -- A generic continuous random variable class meant for subclassing.
- rv_continuous.pdf(x, *args, **kwds) -- Probability density function at x of the given RV.
- rv_continuous.logpdf(x, *args, **kwds) -- Log of the probability density function at x of the given RV.
- rv_continuous.cdf(x, *args, **kwds) -- Cumulative distribution function of the given RV.

- rv_continuous.logcdf(x, *args, **kwds) -- Log of the cumulative distribution function at x of the given RV.
- rv_continuous.sf(x, *args, **kwds) -- Survival function (1-cdf) at x of the given RV.
- rv_continuous.logsf(x, *args, **kwds) -- Log of the survival function of the given RV.
- rv_continuous.ppf(q, *args, **kwds) -- Percent point function (inverse of cdf) at q of the given RV.
- rv_continuous.isf(q, *args, **kwds) -- Inverse survival function at q of the given RV.
- rv_continuous.moment(n, *args, **kwds) -- n'th order non-central moment of distribution.
- rv_continuous.stats(*args, **kwds) -- Some statistics of the given RV
- rv_continuous.entropy(*args, **kwds) -- Differential entropy of the RV.
- rv_continuous.fit(data, *args, **kwds) -- Return MLEs for shape, location, and scale parameters from data.
- rv_continuous.expect([func, args, loc, ...]) -- Calculate expected value of a function with respect to the distribution.

Calling the instance as a function returns a frozen pdf whose shape, location, and scale parameters are fixed.

Similarly, each discrete distribution is an instance of the class rv_discrete:

- rv_discrete([a, b, name, badvalue, ...]) -- A generic discrete random variable class meant for subclassing.
- rv_discrete.rvs(*args, **kwargs) -- Random variates of given type.
- rv_discrete.pmf(k, *args, **kwds) -- Probability mass function at k of the given RV.
- rv_discrete.logpmf(k, *args, **kwds) -- Log of the probability mass function at k of the given RV.
- rv_discrete.cdf(k, *args, **kwds) -- Cumulative distribution function of the given RV.

- rv_discrete.logcdf(k, *args, **kwds) -- Log of the cumulative distribution function at k of the given RV
- rv_discrete.sf(k, *args, **kwds) -- Survival function (1-cdf) at k of the given RV.
- rv_discrete.logsf(k, *args, **kwds) -- Log of the survival function of the given RV.
- rv_discrete.ppf(q, *args, **kwds) -- Percent point function (inverse of cdf) at q of the given RV
- rv_discrete.isf(q, *args, **kwds) -- Inverse survival function (1-sf) at q of the given RV.
- rv_discrete.stats(*args, **kwds) -- Some statistics of the given RV
- rv_discrete.moment(n, *args, **kwds) -- n'th order non-central moment of distribution.
- rv_discrete.entropy(*args, **kwds) -- Differential entropy of the RV.
- rv_discrete.expect([func, args, loc, lb, ...]) -- Calculate expected value of a function with respect to the distribution

Continuous distributions:

- alpha -- An alpha continuous random variable.
- anglit -- An anglit continuous random variable.
- arcsine -- An arcsine continuous random variable.
- beta -- A beta continuous random variable.
- betaprime -- A beta prime continuous random variable.
- bradford -- A Bradford continuous random variable.
- burr -- A Burr continuous random variable.
- cauchy -- A Cauchy continuous random variable.
- chi -- A chi continuous random variable.
- chi2 -- A chi-squared continuous random variable.
- cosine -- A cosine continuous random variable.
- dgamma -- A double gamma continuous random variable.
- dweibull -- A double Weibull continuous random variable.
- erlang -- An Erlang continuous random variable.

- expon -- An exponential continuous random variable.
- exponweib -- An exponentiated Weibull continuous random variable.
- exponpow -- An exponential power continuous random variable.
- f -- An F continuous random variable.
- fatiguelife -- A fatigue-life (Birnbaum-Sanders) continuous random variable.
- fisk -- A Fisk continuous random variable.
- foldcauchy -- A folded Cauchy continuous random variable.
- foldnorm -- A folded normal continuous random variable.
- frechet_r -- A Frechet right (or Weibull minimum) continuous random variable.
- frechet_l -- A Frechet left (or Weibull maximum) continuous random variable.
- genlogistic -- A generalized logistic continuous random variable.
- genpareto -- A generalized Pareto continuous random variable.
- genexpon -- A generalized exponential continuous random variable.
- genextreme -- A generalized extreme value continuous random variable.
- gausshyper -- A Gauss hypergeometric continuous random variable.
- gamma -- A gamma continuous random variable.
- gengamma -- A generalized gamma continuous random variable.
- genhalflogistic -- A generalized half-logistic continuous random variable.
- gilbrat -- A Gilbrat continuous random variable.
- gompertz -- A Gompertz (or truncated Gumbel) continuous random variable.
- gumbel_r -- A right-skewed Gumbel continuous random variable.
- gumbel_l -- A left-skewed Gumbel continuous random variable.
- halfcauchy -- A Half-Cauchy continuous random variable.
- halflogistic -- A half-logistic continuous random variable.
- halfnorm -- A half-normal continuous random variable.
- hypsecant -- A hyperbolic secant continuous random variable.
- invgamma -- An inverted gamma continuous random variable.
- invgauss -- An inverse Gaussian continuous random variable.
- invweibull -- An inverted Weibull continuous random variable.
- johnsonsb -- A Johnson SB continuous random variable.
- johnsonsu -- A Johnson SU continuous random variable.

- ksone -- General Kolmogorov-Smirnov one-sided test.
- kstwobign -- Kolmogorov-Smirnov two-sided test for large N.
- laplace -- A Laplace continuous random variable.
- logistic -- A logistic (or Sech-squared) continuous random variable.
- loggamma -- A log gamma continuous random variable.
- loglaplace -- A log-Laplace continuous random variable.
- lognorm -- A lognormal continuous random variable.
- lomax -- A Lomax (Pareto of the second kind) continuous random variable.
- maxwell -- A Maxwell continuous random variable.
- mielke -- A Mielke's Beta-Kappa continuous random variable.
- nakagami -- A Nakagami continuous random variable.
- ncx2 -- A non-central chi-squared continuous random variable.
- ncf -- A non-central F distribution continuous random variable.
- nct -- A non-central Student's T continuous random variable.
- norm -- A normal continuous random variable.
- pareto -- A Pareto continuous random variable.
- pearson3 -- A pearson type III continuous random variable.
- powerlaw -- A power-function continuous random variable.
- powerlognorm -- A power log-normal continuous random variable.
- powernorm -- A power normal continuous random variable.
- rdist -- An R-distributed continuous random variable.
- reciprocal -- A reciprocal continuous random variable.
- rayleigh -- A Rayleigh continuous random variable.
- rice -- A Rice continuous random variable.
- recipinvgauss -- A reciprocal inverse Gaussian continuous random variable.
- semicircular -- A semicircular continuous random variable.
- t -- A Student's T continuous random variable.
- triang -- A triangular continuous random variable.
- truncexpon -- A truncated exponential continuous random variable.
- truncnorm -- A truncated normal continuous random variable.
- tukeylambda -- A Tukey-Lamdba continuous random variable.
- uniform -- A uniform continuous random variable.

- vonmises -- A Von Mises continuous random variable.
- wald -- A Wald continuous random variable.
- weibull_min -- A Frechet right (or Weibull minimum) continuous random variable.
- weibull_max -- A Frechet left (or Weibull maximum) continuous random variable.
- wrapcauchy -- A wrapped Cauchy continuous random variable.
- Multivariate -- distributions
- multivariate_normal -- A multivariate normal random variable.
- Discrete -- distributions
- bernoulli -- A Bernoulli discrete random variable.
- binom -- A binomial discrete random variable.
- boltzmann -- A Boltzmann (Truncated Discrete Exponential) random variable.
- dlaplace -- A Laplacian discrete random variable.
- geom -- A geometric discrete random variable.
- hypergeom -- A hypergeometric discrete random variable.
- logser -- A Logarithmic (Log-Series, Series) discrete random variable.
- nbinom -- A negative binomial discrete random variable.
- planck -- A Planck discrete exponential random variable.
- poisson -- A Poisson discrete random variable.
- randint -- A uniform discrete random variable.
- skellam -- A Skellam discrete random variable.
- zipf -- A Zipf discrete random variable.

Statistical functions -- Several of these functions have a similar version in scipy.stats.mstats which work for masked arrays.

- **describe(a[, axis])** -- Computes several descriptive statistics of the passed array.
- gmean(a[, axis, dtype]) -- Compute the geometric mean along the specified axis.
- hmean(a[, axis, dtype]) -- Calculates the harmonic mean along the specified axis.

- **kurtosis(a[, axis, fisher, bias])** -- Computes the kurtosis (Fisher or Pearson) of a dataset.
- **kurtosistest(a[, axis])** -- Tests whether a dataset has normal kurtosis
- **mode(a[, axis])** -- Returns an array of the modal (most common) value in the passed array.
- **moment(a[, moment, axis])** -- Calculates the nth moment about the mean for a sample.
- **normaltest(a[, axis])** -- Tests whether a sample differs from a normal distribution.
- **skew(a[, axis, bias])** -- Computes the skewness of a data set.
- **skewtest(a[, axis])** -- Tests whether the skew is different from the normal distribution.
- tmean(a[, limits, inclusive]) -- Compute the trimmed mean.
- tvar(a[, limits, inclusive]) -- Compute the trimmed variance
- tmin(a[, lowerlimit, axis, inclusive]) -- Compute the trimmed minimum
- tmax(a[, upperlimit, axis, inclusive]) -- Compute the trimmed maximum
- tstd(a[, limits, inclusive]) -- Compute the trimmed sample standard deviation
- tsem(a[, limits, inclusive]) -- Compute the trimmed standard error of the mean.
- nanmean(x[, axis]) -- Compute the mean over the given axis ignoring nans.
- nanstd(x[, axis, bias]) -- Compute the standard deviation over the given axis, ignoring nans.
- nanmedian(x[, axis]) -- Compute the median along the given axis ignoring nan values.
- **variation(a[, axis])** -- Computes the coefficient of variation, the ratio of the biased standard deviation to the mean.
- cumfreq(a[, numbins, defaultreallimits, weights]) -- Returns a cumulative frequency histogram, using the histogram function.
- **histogram2(a, bins)** -- Compute histogram using divisions in bins.
- **histogram(a[, numbins, defaultlimits, ...])** -- Separates the range into several bins and returns the number of instances in each bin.
- itemfreq(a) -- Returns a 2-D array of item frequencies.

- percentileofscore(a, score[, kind]) -- The percentile rank of a score relative to a list of scores.
- scoreatpercentile(a, per[, limit, ...]) -- Calculate the score at a given percentile of the input sequence.
- relfreq(a[, numbins, defaultreallimits, weights]) -- Returns a relative frequency histogram, using the histogram function.
- binned_statistic(x, values[, statistic, ...]) -- Compute a binned statistic for a set of data.
- binned_statistic_2d(x, y, values[, ...]) -- Compute a bidimensional binned statistic for a set of data.
- binned_statistic_dd(sample, values[, ...]) -- Compute a multidimensional binned statistic for a set of data.
- obrientransform(*args) -- Computes the O'Brien transform on input data (any number of arrays).
- signaltonoise(a[, axis, ddof]) -- The signal-to-noise ratio of the input data.
- bayes_mvs(data[, alpha]) -- Bayesian confidence intervals for the mean, var, and std.
- **sem(a[, axis, ddof])** -- Calculates the standard error of the mean (or standard error of measurement) of the values in the input array.
- zmap(scores, compare[, axis, ddof]) -- Calculates the relative z-scores.
- **zscore(a[, axis, ddof])** -- Calculates the z score of each value in the sample, relative to the sample mean and standard deviation.
- sigmaclip(a[, low, high]) -- Iterative sigma-clipping of array elements.
- threshold(a[, threshmin, threshmax, newval]) -- Clip array to a given value.
- trimboth(a, proportiontocut[, axis]) -- Slices off a proportion of items from both ends of an array.
- trim1(a, proportiontocut[, tail]) -- Slices off a proportion of items from ONE end of the passed array distribution.
- **f_oneway(*args)** -- Performs a 1-way ANOVA.
- **pearsonr(x, y)** -- Calculates a Pearson correlation coefficient and the p-value for testing non-correlation.

- spearmanr(a[, b, axis]) -- Calculates a Spearman rank-order correlation coefficient and the p-value to test for non-correlation.
- pointbiserialr(x, y) -- Calculates a point biserial correlation coefficient and the associated p-value.
- kendalltau(x, y[, initial_lexsort]) -- Calculates Kendall's tau, a correlation measure for ordinal data.
- **linregress(x[, y])** -- Calculate a regression line
- **ttest_1samp(a, popmean[, axis])** -- Calculates the T-test for the mean of ONE group of scores.
- **ttest_ind(a, b[, axis, equal_var])** -- Calculates the T-test for the means of TWO INDEPENDENT samples of scores.
- **ttest_rel(a, b[, axis])** -- Calculates the T-test on TWO RELATED samples of scores, a and b.
- **kstest(rvs, cdf[, args, N, alternative, mode])** -- Perform the Kolmogorov-Smirnov test for goodness of fit.
- **chisquare(f_obs[, f_exp, ddof, axis])** -- Calculates a one-way chi square test.
- power_divergence(f_obs[, f_exp, ddof, axis, ...]) -- Cressie-Read power divergence statistic and goodness of fit test.
- **ks_2samp(data1, data2)** -- Computes the Kolmogorov-Smirnov statistic on 2 samples.
- mannwhitneyu(x, y[, use_continuity]) -- Computes the Mann-Whitney rank test on samples x and y.
- tiecorrect(rankvals) -- Tie correction factor for ties in the Mann-Whitney U and Kruskal-Wallis H tests.
- rankdata(a[, method]) -- Assign ranks to data, dealing with ties appropriately.
- ranksums(x, y) -- Compute the Wilcoxon rank-sum statistic for two samples.
- wilcoxon(x[, y, zero_method, correction]) -- Calculate the Wilcoxon signed-rank test.
- kruskal(*args) -- Compute the Kruskal-Wallis H-test for independent samples

- friedmanchisquare(*args) -- Computes the Friedman test for repeated measurements
- ansari(x, y) -- Perform the Ansari-Bradley test for equal scale parameters
- bartlett(*args) -- Perform Bartlett's test for equal variances
- levene(*args, **kwds) -- Perform Levene test for equal variances.
- shapiro(x[, a, reta]) -- Perform the Shapiro-Wilk test for normality.
- anderson(x[, dist]) -- Anderson-Darling test for data coming from a particular distribution
- anderson_ksamp(samples[, midrank]) -- The Anderson-Darling test for k-samples.
- **binom_test(x[, n, p])** -- Perform a test that the probability of success is p.
- fligner(*args, **kwds) -- Perform Fligner's test for equal variances.
- mood(x, y[, axis]) -- Perform Mood's test for equal scale parameters.
- boxcox(x[, lmbda, alpha]) -- Return a positive dataset transformed by a Box-Cox power transformation.
- boxcox_normmax(x[, brack, method]) -- Compute optimal Box-Cox transform parameter for input data.
- boxcox_llf(lmb, data) -- The boxcox log-likelihood function.
- entropy(pk[, qk, base]) -- Calculate the entropy of a distribution for given probability values.

Contingency table functions:

- **chi2_contingency(observed[, correction, lambda_])** -- Chi-square test of independence of variables in a contingency table.
- contingency.expected_freq(observed) -- Compute the expected frequencies from a contingency table.
- contingency.margins(a) -- Return a list of the marginal sums of the array a.
- fisher_exact(table[, alternative]) -- Performs a Fisher exact test on a 2x2 contingency table.

Plot-tests:

- ppcc_max(x[, brack, dist]) -- Returns the shape parameter that maximizes the probability plot correlation coefficient for the given data to a one-parameter family of distributions.
- ppcc_plot(x, a, b[, dist, plot, N]) -- Returns (shape, ppcc), and optionally plots shape vs.
- probplot(x[, sparams, dist, fit, plot]) -- Calculate quantiles for a probability plot, and optionally show the plot.
- boxcox_normplot(x, la, lb[, plot, N]) -- Compute parameters for a Box-Cox normality plot, optionally show it.

Masked statistics functions:

Statistical functions for masked arrays (scipy.stats.mstats):

- scipy.stats.mstats.argstoarray
- scipy.stats.mstats.betai
- scipy.stats.mstats.chisquare
- scipy.stats.mstats.count_tied_groups
- scipy.stats.mstats.describe
- scipy.stats.mstats.f_oneway
- scipy.stats.mstats.f_value_wilks_lambda
- scipy.stats.mstats.find_repeats
- scipy.stats.mstats.friedmanchisquare
- scipy.stats.mstats.kendalltau
- scipy.stats.mstats.kendalltau_seasonal
- scipy.stats.mstats.kruskalwallis
- scipy.stats.mstats.kruskalwallis
- scipy.stats.mstats.ks_twosamp
- scipy.stats.mstats.ks_twosamp
- scipy.stats.mstats.kurtosis
- scipy.stats.mstats.kurtosistest

- scipy.stats.mstats.linregress
- scipy.stats.mstats.mannwhitneyu
- scipy.stats.mstats.plotting_positions
- scipy.stats.mstats.mode
- scipy.stats.mstats.moment
- scipy.stats.mstats.mquantiles
- scipy.stats.mstats.msign
- scipy.stats.mstats.normaltest
- scipy.stats.mstats.obrientransform
- scipy.stats.mstats.pearsonr
- scipy.stats.mstats.plotting_positions
- scipy.stats.mstats.pointbiserialr
- scipy.stats.mstats.rankdata
- scipy.stats.mstats.scoreatpercentile
- scipy.stats.mstats.sem
- scipy.stats.mstats.signaltonoise
- scipy.stats.mstats.skew
- scipy.stats.mstats.skewtest
- scipy.stats.mstats.spearmanr
- scipy.stats.mstats.theilslopes
- scipy.stats.mstats.threshold
- scipy.stats.mstats.tmax
- scipy.stats.mstats.tmean
- scipy.stats.mstats.tmin
- scipy.stats.mstats.trim
- scipy.stats.mstats.trima
- scipy.stats.mstats.trimboth
- scipy.stats.mstats.trimmed_stde
- scipy.stats.mstats.trimr
- scipy.stats.mstats.trimtail
- scipy.stats.mstats.tsem
- scipy.stats.mstats.ttest_onesamp

- scipy.stats.mstats.ttest_ind
- scipy.stats.mstats.ttest_onesamp
- scipy.stats.mstats.ttest_rel
- scipy.stats.mstats.tvar
- scipy.stats.mstats.variation
- scipy.stats.mstats.winsorize
- scipy.stats.mstats.zmap
- scipy.stats.mstats.zscore

Univariate and multivariate kernel density estimation (scipy.stats.kde):

- gaussian_kde(dataset[, bw_method]) -- Representation of a kernel-density estimate using Gaussian kernels.

For many more stat related functions install the software R and the interface package rpy.

Statistical functions

describe	**-- Descriptive statistics**
gmean	**-- Geometric mean**
hmean	**-- Harmonic mean**
kurtosis	**-- Fisher or Pearson kurtosis**
kurtosistest	**--**
mode	**-- Modal value**
moment	**-- Central moment**
normaltest	**--**
skew	**-- Skewness**
skewtest	**--**
tmean	**-- Truncated arithmetic mean**
tvar	**-- Truncated variance**
tmin	**--**
tmax	**--**
tstd	**--**
tsem	**--**
nanmean	**-- Mean, ignoring NaN values**
nanstd	**-- Standard deviation, ignoring NaN values**

nanmedian -- Median, ignoring NaN values

variation -- Coefficient of variation

histogram _

linregress

ttest_1samp

ttest_ind

ttest_rel

kstest

chisquare

ks_2samp

chi2_contingency

contingency.expected_freq

contingency.margins