Un micro tutorial de Python

Números

```
>>> 2+2
4
>>> (50 - 5*6) / 4
5
>>> 2309874209847209 * 120894739
279251639722309561933451
>>> 3 * 3.75 / 1.5
7.5
>>> (2+3j) * (8-4j)
(28+16j)
```

Cadenas

```
>>> 'Secuencia de caracteres'
'Secuencia de caracteres'
>>> "Hola" + " mundo"
'Hola mundo'
>>> "Eco " * 4
'Eco Eco Eco Eco '
>>> saludo = 'Hola mundo'
>>> saludo[0], saludo[-2]
('H', 'd')
>>> saludo[2:5]
'la '
```

Listas

```
>>> a = [100, 'huevos', 'sal']
>>> a
[100, 'huevos', 'sal']
>>> a[0]
100
>>> a[-2:]
['huevos', 'sal']
>>> a + ['oro', 9]
[100, 'huevos', 'sal', 'oro', 9]
>>> a[0] = "manteca"
>>> a
['manteca', 'huevos', 'sal']
```

Conjuntos

```
>>> f = set("abracadabra")
>>> f
set(['a', 'r', 'b', 'c', 'd'])
>>> f & set(['a','e','i','o','u'])
set(['a'])
```

Diccionarios

```
>>> dias = {"Ene": 31, "Jul": 30}
>>> dias
{'Ene': 31, 'Jul': 30}
>>> dias["Ene"]
31
>>> dias["Ago"] = 31
>>> dias["Jul"] = 31
>>> dias
{'Ago': 31, 'Ene': 31, 'Jul': 31}
>>> "Mar" in dias
False
>>> dias.keys()
(['Ago', 'Ene', 'Jul']
>>> dias.values()
[31, 31, 31])
```

if

```
if <expresion>:
    <suite>
elif <expresion>:
    <suite>
else:
    <suite>
```

Una <expresion> es algo que evalúa siempre a Verdadero o Falso
Operadores lógicos: `or`, `and`, `not`
Comparadores: `< > == != in is`
Una <suite> es un bloque de código, de una o más lineas, delimitado por la sangría.

while

```
while <expresion>:
    <suite>
```

for

```
>>> bichos = ["pulgas", "piojos"]
>>> for bich in bichos:
...     print "Mata-" + bich
...
Mata-pulgas
Mata-piojos
```

List comprehensions

```
>>> vec = [3, 7, 12, 0, 3, -13]
>>> [x**2 for x in vec]
[9, 49, 144, 0, 9, 169]
>>> [x**2 for x in vec if x <= 7]
[9, 49, 0, 9, 169]
```

Excepciones

```
>>> 5 / 0
Traceback (most recent call last):
  File "<stdin>", line 1, in ...
ZeroDivisionError: int ... by zero

>>> try:
...     5 / 0
... except ZeroDivisionError:
...     print "oops!"
...
oops!
```

```
try:
    <suite>
except [Excepcion1, ...]:
    <suite>
finally:
    <suite>
else:
    <suite>
```

Si hay una excepción en la suite del `try`, se ejecuta la suite del `except`. Si no hubo ninguna excepción, se ejecuta la suite del `else`. Y siempre se ejecuta la suite del `finally`.

```
>>> raise ValueError("Ejemplo!")
Traceback (most recent call last):
  File "<stdin>", line 1, in ...
ValueError: Ejemplo!
```

Funciones

```
>>> def alcuadrado(n):
...     res = n ** 2
...     return res
...
>>> alcuadrado(3)
9
```

```
>>> def func(a, b=0, c=7):
...     return a, b, c
...
>>> func(1)
(1, 0, 7)
>>> func(1, 3)
(1, 3, 7)
>>> func(1, 3, 9)
(1, 3, 9)
>>> func(1, c=9)
(1, 0, 9)
>>> func(b=2, a=-3)
(-3, 2, 7)
```

Clases

```
>>> class Posicion:
...   def __init__(self, x, y):
...     self.x = x
...     self.y = y
...   def distancia(self):
..      x = self.x**2 + self.y**2
..      return math.sqrt(x)
...
>>> p1 = Posicion(3, 4)
>>> p1.x
3
>>> p1.dist()
5.0
>>> p2 = Posicion(7, 9)
>>> p2.y
9
>>> p1.y
4
```

Módulos

- Funciones, clases, y/o código suelto, todo en un archivo
- Es un .py normal, sólo lo importamos y directamente lo usamos
- Fácil, rápido, ¡y funciona!

Armamos un pos.py que contiene la clase definida arriba:

```
>>> import pos
>>> p = pos.Posicion(2, 3)
>>> p.x
2
```

El Zen de Python

Lindo es mejor que feo.
Explícito es mejor que implícito.
Simple es mejor que complejo.
Complejo es mejor que complicado.
Plano es mejor que anidado.
Espaciado es mejor que denso.
La legibilidad es importante.
Los casos especiales no son lo suficientemente especiales como para romper las reglas.
Sin embargo la practicidad le gana a la pureza.
Los errores nunca deberían pasar silencioosamente.
A menos que se silencien explícitamente.
Frente a la ambigüedad, evitar la tentación de adivinar.
Debería haber una, y preferiblemente sólo una, manera obvia de hacerlo.
Aunque no sea obvia a menos que seas holandés (GvR).
Ahora es mejor que nunca.
A pesar de que nunca es muchas veces mejor que ya mismo.
Si la implementación es difícil de explicar, es una mala idea.
Si la implementación es fácil de explicar, quizás sea una buena idea.
Los espacios de nombres son una gran idea, ¡hagamos más!

por Tim Peters

python™

¿La serpiente te asusta?

¡Dejate atrapar!

Programá en ...

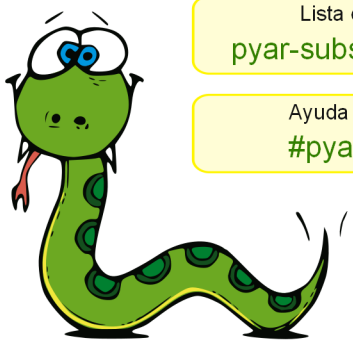
python™

*En **PyAr** te vamos a ayudar:*

Python Argentina
http://www.python.org.ar

Lista de correo, mandá mail a:
pyar-subscribe@decode.com.ar

Ayuda instantánea, entrá al chat:
#pyar (irc.freenode.org)



PyAr