1-python_tour

October 22, 2020

1 Tour de Python

1.1 Instrucciones

- En **tu carpeta** (la que está dentro de `alumnos`) crea un archivo llamado `1-python_tour`.
- En la primera celda pon el formato `Heading 1` y teclea `Tour de Python`.
- Luego teclea (**no copies y pegues**) el código mostrado aquí y ve ejecutándolo.

1.2 Ayuda en ipython

[1]:
```
str?
```

Presiona `Tab` en la siguiente celda

[5]:
```
str(".")
```

[5]:
```
'.'
```

1.3 Variables

Asignación le decimos a la computadora que la variable `x` tiene *asignado* el valor `1`.

[6]:
```
x = 1
```

[7]:
```
x
```

[7]:
```
1
```

[8]:
```
print (x)
```

```
1
```

Asignación doble

[9]:
```
a, b = 3, 4
print ("El valor de a es ", a)
print ("El valor de b es ", b)
```

```
El valor de a es  3
El valor de b es  4
```

Los **tipos** de variables "básicos" son los siguientes:

Enteros

[10]:
```
x = 3
x
```

[10]:
```
3
```

Flotantes

[11]:
```
y = 3.5
y
```

[11]:
```
3.5
```

Números Complejos de la forma $a + bι$

[12]:
```
z = 3 + 5j
z
```

[12]:
```
(3+5j)
```

Booleanos

[13]:
```
a = True
a
```

[13]:
```
True
```

Python también soporta **notación científica**

[14]:
```
x = 1.2e34
x
```

[14]:
```
1.2e+34
```

[15]:
```
x = 1e-12 + 3.56e4j
x
```

[15]:
```
(1e-12+35600j)
```

Nota como cambiamos tres veces el tipo y el valor de la variable `x`, el hecho de *que se pueda* hacer no implica ** que debas de hacerlo **, se considera como una pésima práctica de programación...

Cadenas de texto

[19]:
```
#Poner la "U" para el encoding UTF-8
cadena = u"Esto es una cadena de texto con acentos y otros símbolos UTF-8 ¿?¡!"
cadena
```

[19]:
```
'Esto es una cadena de texto con acentos y otros símbolos UTF-8 ¿?¡!'
```

[20]:
```
print(cadena)
```

```
Esto es una cadena de texto con acentos y otros símbolos UTF-8 ¿?¡!
```

Ejercicio Usa la ayuda en línea de ipython e investiga como: - Capitalizarla. - Convertirla a minúsculas. - Convertir a mayúsculas. - Eliminar los espacios. de la cadena "Anita lava la tina". Aplica las cuatro operaciones una tras otra.

Es posible ser explícito en la asignación del tipo de la variable

[21]:
```
ejercicio = "anita lava la tina"
```

[22]:
```
str.capitalize(ejercicio)
```

[22]:
```
'Anita lava la tina'
```

[23]:
```
ejercicio.lower()
```

[23]:
```
'anita lava la tina'
```

[24]:
```
str.upper(ejercicio)
```

[24]:
```
'ANITA LAVA LA TINA'
```

[26]:
```
#Quitamos los espacios
str.replace(ejercicio," ","")
```

[26]:
```
'anitalavalatina'
```

[27]:
```
x = int(2)
y = float(3)
z = complex(1+2j)
a = str("Hola")
```

[28]:
```
print(x, y, z, a)
```

```
2 3.0 (1+2j) Hola
```

1.4 Operaciones

Ejercicio:

¿Puedes identificar las operaciones?

[29]:
```
x = 3
y = 2
```

[30]:
```
x+y
```

[30]: 5

[31]:
```
x-y
```

[31]: 1

[32]:
```
x*y
```

[32]: 6

[33]:
```
x**y
```

[33]: 9

[34]:
```
x/y
```

[34]: 1.5

[35]:
```
y%x
```

[35]: 2

Ejercicio:

Repite las operaciones pero ahora usa el siguiente valor (y tipo) de `y`.

¿Qué sucede? ¿Cuál es el tipo del valor de salida?

[36]:
```
y = 4.1
```

Ejercicio:

Repite las operaciones pero ahora usa el valor (y tipo) de `y`.

¿Qué sucede? ¿Cuál es el tipo de valor de salida?

[37]:
```
y = 4 + 3j
```

Ejercicio:

¿Qué pasa en lo siguiente?

[38]:
```
a = 3
b = "hola"
```

[39]:
```
a + b
```

```
␣
↪---------------------------------------------------------------------------

       TypeError                                 Traceback (most recent call␣
↪last)

       <ipython-input-39-bd58363a63fc> in <module>
   ----> 1 a + b


       TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

[40]:
```
str(a) + b
```

[40]:
```
'3hola'
```

[41]:
```
b + b
```

[41]:
```
'holahola'
```

[42]:
```
 a * b
```

[42]:
```
'holaholahola'
```

Ejercicio:

¿Qué puedes deducir de lo siguiente?

[43]:
```
a, b, c = 1.0, 2.0, 3.0
print (a)
print (b)
print (c)
```

```
1.0
2.0
3.0
```

[44]:
```
a + b/c
```

[44]:
```
1.6666666666666665
```

[45]:
```
(a+b)/c
```

[45]:
```
1.0
```

También *modificadores* de una variable

[46]:
```
x= 0
x += 1
```

[47]:
```
print (x)
```

```
1
```

[48]:
```
x -= 2
```

[49]:
```
print(x)
```

```
-1
```

[50]:
```
x *= 3.14
print (x)
```

```
-3.14
```

[51]:
```
x /= 5*a
```

[52]:
```
print(x)
```

```
-0.628
```

1.5 Condicionales

[53]:
```
if x > 3:
    print("Esto no se debería de imprimir")
    print("Nota que es muy peligroso hacer esto con flotantes...")
```

[54]:
```
x = 5
```

[55]:
```
if x > 3:
    print("Esto SI se debería de imprimir")
```

```
Esto SI se debería de imprimir
```

Los siguientes operadores lógicos se pueden utilizar

[56]:
```
x == 1
```

[56]:
```
False
```

[57]:
```
x > 1
```

[57]:
```
True
```

[58]:
```
x < 1
```

[58]: False

[59]:
```
x >= 1
```

[59]: True

[60]:
```
x != 1
```

[60]: True

[61]:
```
x > 10 or x <= 5
```

[61]: True

[62]:
```
x > 10 and x <= 5
```

[62]: False

[63]:
```
print(x)
```

```
5
```

[64]:
```
#Usar u para el encoding
if x >= 5 and x < 10:
    print(u"¡Hola!")
else:
    print(u"Nunca me verás")
```

```
¡Hola!
```

Ejercicio

Usando los *widgets* que proporciona el notebook, programa una interfaz que indique (imprimiendo) si el número que se introduce es par o impar.

[65]:
```
def par(x):
    if x%2:
        print("par")
    else:
        print("impar")
```

[68]:
```
from ipywidgets import interact
interact(par,x=10);
```

```
interactive(children=(IntSlider(value=10, description='x', max=30, min=-10), Output()), _dom_cl
```

1.6 Bucles

Los bucles `for` en python, utilizan *iteradores*:

[69]:
```
for i in (1,2,3,4,5):
    print(i)
```

```
1
2
3
4
5
```

[70]:
```
for x in u"Hola Como estás?":
    print(x)
```

```
H
o
l
a

C
o
m
o

e
s
t
á
s
?
```

El que sigue no es el mejor ejemplo de un uso de `while`, pero es con fines ilustrativos

[71]:
```
i = 1
while i > 0 and i <= 5:
    print(i)
    i += 1
```

```
1
2
3
4
5
```

[72]:
```
#(inicio, fin, salto) --> no incluye el final
for i in range(3,10, 2):
    print (i)
```

```
3
5
7
9
```

Ejercicio:

Calcula el valor de la suma $\sum_{k=0}^{1000} \left(\frac{1}{k}\right)^2$.

[73]:
```
suma=0
for i in range(1,1001):
    suma+=(1/i)**2
print(suma)
```

```
1.6439345666815615
```

1.7 Estructuras de datos *built-in*

1.7.1 Listas

[74]:
```
a_list = [1, 2, 3, 4, 5]
a_list
```

[74]:
```
[1, 2, 3, 4, 5]
```

[75]:
```
another_list = [1, 1.3, 4.5, 2+1j, [1,2]]
another_list
```

[75]:
```
[1, 1.3, 4.5, (2+1j), [1, 2]]
```

[76]:
```
a_list[3]
```

[76]:
```
4
```

[77]:
```
another_list[3]
```

[77]:
```
(2+1j)
```

[78]:
```
another_list[4]
```

[78]:
```
[1, 2]
```

[79]:
```
a_list[-1]
```

[79]:
```
5
```

[80]:
```
a_list[1:]
```

[80]:
```
[2, 3, 4, 5]
```

[81]:
```
a_list[2:4]
```

[81]:
```
[3, 4]
```

[82]:
```
a_list[:4]
```

[82]:
```
[1, 2, 3, 4]
```

[83]:
```
n=5
```

[84]:
```
#Agregar
a_list.append(n)
```

[85]:
```
a_list
```

[85]:
```
[1, 2, 3, 4, 5, 5]
```

[86]:
```
#Sacar
a_list.pop()
```

[86]:
```
5
```

[87]:
```
a_list
```

[87]:
```
[1, 2, 3, 4, 5]
```

[88]:
```
a_list.append(3)
print(a_list)
a_list[1] = 3
print(a_list)
a_list.append(5)
print(a_list)
```

```
[1, 2, 3, 4, 5, 3]
[1, 3, 3, 4, 5, 3]
[1, 3, 3, 4, 5, 3, 5]
```

[89]:
```
a_list.remove?
```

[90]:
```
a_list.remove(3) #borra la primer ocurrencia del valor en la lista
```

[91]:
```
a_list
```

[91]:
```
[1, 3, 4, 5, 3, 5]
```

[92]:
```
#Podemos hacer una lista vacía
empty_list = []
```

```
empty_list
```

[92]: []

[93]:
```
for element in a_list:
    print(element)
```

```
1
3
4
5
3
5
```

[94]:
```
for r in range(5):
    print(r)
```

```
0
1
2
3
4
```

List comprehensions En matemáticas, es normal describir las listas (siendo estrictos conjuntos) de la siguente manera:

- $S = \{x^2 \mid x\ en\ \{0 \dots 9\}\}$
- $V = (1, 2, 4, 8, \dots, 2^{12})$
- $M = \{x \mid x\ en\ S\ y\ x\ es\ impar\}$

En python existe una técnica muy poderosa de creación/manipulación de listas que permite escribirlas casi igual:

[95]:
```
S = [x**2 for x in range(10)]
V = [2**x for x in range(13)]
M = [x for x in S if x%2 == 0]
print(S)
print(V)
print(M)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
[0, 4, 16, 36, 64]
```

Las *list comprehensions* pueden ser utilizadas también con cadenas:

[98]:
```
palabras = 'Anita lava la tina'.split()
print(palabras)
transformacion = [[w.upper(), w.lower(), len(w)] for w in palabras]
```

```
print("==============================")
for t in transformacion:
    print(t)
```

```
['Anita', 'lava', 'la', 'tina']
==============================
['ANITA', 'anita', 5]
['LAVA', 'lava', 4]
['LA', 'la', 2]
['TINA', 'tina', 4]
```

[99]:
```
transformacion
```

[99]:
```
[['ANITA', 'anita', 5],
 ['LAVA', 'lava', 4],
 ['LA', 'la', 2],
 ['TINA', 'tina', 4]]
```

Es posible usar varios `for` en una expresión, el siguiente bloque muestra los tripletes *pitagóricos* (aquellos a, b y c que cumplen con $a^2 + b^2 = c^2$)

[100]:
```
n = 25
[(x, y, z) for x in range(1, n) for y in range(x, n) for z in range(y, n) if␣
 ↪x*x + y*y == z*z]
```

[100]:
```
[(3, 4, 5), (5, 12, 13), (6, 8, 10), (8, 15, 17), (9, 12, 15), (12, 16, 20)]
```

1.7.2 Diccionarios

Los diccionarios o *mapas* son otra estructura de datos importante en python

[101]:
```
diccionario = {'a':1, 'b':2, 'coral': 'amarillo'}
diccionario
```

[101]:
```
{'a': 1, 'b': 2, 'coral': 'amarillo'}
```

[102]:
```
diccionario.keys()
```

[102]:
```
dict_keys(['a', 'b', 'coral'])
```

[103]:
```
diccionario.values()
```

[103]:
```
dict_values([1, 2, 'amarillo'])
```

[104]:
```
diccionario['a']
```

[104]:
```
1
```

[105]:
```
diccionario['coral']
```

[105]:
```
'amarillo'
```

Ejercicio Usando la ayuda de ipython responde: - ¿Cómo puedo agregar elementos a un diccionario? - ¿Cómo puedo eliminar elementos a un diccionario? - ¿Cómo cambia el método `pop` respecto a la listas? - ¿Cómo puedo iterar en un diccionario?

[ ]:
```

```

1.8 Funciones

Principio DRY: * Don't repeat yourself*

[106]:
```
from math import sqrt, atan
def polar(x,y):
    """
    x, y son un par ordenado
    """
    r = sqrt(x**2 + y**2)
    theta = atan(y/x)
    return r, theta
```

[107]:
```
polar?
```

[185]:
```
polar??
```

[108]:
```
r, theta = polar(3,4)
print("La coordenada r es: " , r)
print("El ángulo theta es: ", theta)
```

```
La coordenada r es:  5.0
El ángulo theta es:  0.9272952180016122
```

Ejercicio

Crea una función que devuelva las coordenadas (x, y, z) a partir de las coordenadas esféricas (r, θ, ϕ). Guárdala a disco, en el archivo `transformaciones.py`

[109]:
```
%%file transformaciones.py

def transformacion(r,theta,fi):
    from math import  sin, cos
    """
    r,theta,fi son las coordenadas esféricas

    """
    x=r*sin(theta)*cos(fi)
    y=r*sin(theta)*sin(fi)
```

```
    z=r*cos(theta)
    print(x,y,z)
```

```
Overwriting transformaciones.py
```

[110]:
```
transformacion??
```

Ejercicio

Usando la función recién creada, usa la librería `interact` para pedir los valores (r, θ, ϕ).

[111]:
```
from transformaciones import transformacion
from ipywidgets import interact
interact(transformacion,r=10,theta=180,fi=180);
```

```
interactive(children=(IntSlider(value=10, description='r', max=30, min=-10), IntSlider(value=18
```