Table of Contents

Módulo 2: Introducción a Python

En este curso usaremos Python 3.6. Lo obtendremos a través de Anaconda 3.6 (https://www.anaconda.com/download/), esta distribución de Python contiene todos los paquetes para cómputo científico.

Introducción a objetos

Python es un lenguaje orientado a objetos. Es decir que todas las cosas con las que trabajamos en Python son objetos particulares con características y métodos específicos.

In [ ]:
```
# un número entero
type(2)
```

In [ ]:
```
# puede multiplicarse
2*3
```

In [ ]:
```
# una cadena de texto
type("Hola")
```

In [ ]:
```
# también, pero no significa lo mismo
"Hola"*3
```

In [ ]:
```
print("Suma de enteros (2+2): ", 2 + 2)
print("Suma de cadenas ('2'+'2'): ", "2" + "2")
```

Las clases pueden tener métodos.

In [ ]:
```
"Hola, ¿cómo estás?".find("es")
```

In [ ]:
```
"Hola, ¿cómo estás?".find("adios")
```

Además de los métodos, que son propios del objeto, existen funciones que reciben objetos específicos.

In [ ]:
```
# El valor absoluto de un número
abs(-10)
```

In [ ]:
```
# ¿El valor absoluto de una palabra?
abs("Hola")
```

También existen "constructores" de objetos que reciben objetos de otra clase.

In [ ]:
```
str(1)
```

In [ ]:
```
str(1)*3
```

In [ ]:
```
int("1")
```

In [ ]:
```
int("1")*3
```

Pero no siempre es legal.

In [ ]:
```
int("Hola")
```

Ejercicio: Imprime un gúgol (un 1 seguido por 100 ceros).

In [ ]:
```
"1" + "0"*100
```

Listas

Una lista es un objeto que contiene objetos. Se define con corchetes: [ ]

In [ ]:
```
type([])
```

Las listas pueden contener cualquier tipo de objeto.

In [ ]:
```
[1, 2, 3, "Hola", ",", "¿cómo estás?", 16]
```

Vamos a asignarla a una variable.

In [ ]:
```
lista_magica = [1, 2, 3, "Hola", ",", "¿cómo estás?", 16]
print(lista_magica)
```

Podemos acceder a sus elementos a través de corchetes.

In [ ]:
```
lista_magica[3]
```

Una vez que accedemos al elemento, ya no tenemos una lista sino el objeto que estaba en esa posición.

In [ ]:
```
type(lista_magica[3])
```

El primer elemento es el 0.

In [ ]:
```
lista_magica[0]
```

In [ ]:
```
[1, 2, 3, 4][0]
```

Si nos pasamos Python nos lo hará saber amablemente.

In [ ]:
```
lista_magica[10]
```

In [ ]:
```
len(lista_magica)
```

También podemos acceder a un rango de una lista (*slice*).

In [ ]:
```
lista_magica[1:5]
```

In [ ]:
```
lista_magica[4:]
```

In [ ]:
```
lista_magica[:4]
```

El tamaño de la lista se obtiene con la función `len`.

In [ ]:
```
len(lista_magica)
```

Para agregar un elemento a la lista, se usa el método `append`.

In [ ]:
```
lista_magica.append("nuevo elemento")
print(lista_magica)
```

También se pueden sumar listas, con lo que se obtiene una nueva lista con los elementos de las 2.

In [ ]:
```
print(lista_magica + lista_magica)
```

Para obtener los valores únicos de una lista, usamos `set`.

In [ ]:
```
set(lista_magica + lista_magica)
```

In [ ]:
```
type(set(lista_magica + lista_magica))
```

Para regresarlo a lista, usamos el constructor `list`.

In [ ]:
```
list(set(lista_magica + lista_magica))
```

Si la lista tiene puros números, podemos sumar sus elementos.

In [ ]:
```
sum([1.2,2.5,3.2])
```

Una cadena de caracteres también puede usarse como una lista.

In [ ]:
```
print('Hola'[1])
print('Hola'[0:2])
```

Condicionales

Los condicionales regresan *booleanos*, es decir Verdadero o Falso si se cumple una cierta condición. Dependiendo de la pregunta que se quiera hacer, existen diversos operadores.

Comparación

In [ ]:
```
# Igualdad
print("1 == 1: ", 1 == 1)
print("1 == 2: ", 1 == 2)
print("1 != 1: ", 1 != 1)
print("1 != 2: ", 1 != 2)
# Desigualdad
print("1 <= 2: ", 1 <= 2)
print("1 >= 2: ", 1 >= 2)
print("1 < 2: ", 1 < 2)
print("1 > 2: ", 1 > 2)
```

In [ ]:
```
lista_magica == lista_magica
```

In [ ]:
```
# Igualdad
print("'hola' == 'hola': ", "hola" == "hola")
print("'hola' == 'adios': ", "hola" == "adios")
print("'hola' != 'hola': ", "hola" != "hola")
print("'hola' != 'adios': ", "hola" != "adios")
```

In [ ]:
```
"hola" == "hola"
```

In [ ]:
```
# Contención
"hola" in "hola, ¿cómo estás?"
```

In [ ]:
```
"adios" in "hola, ¿cómo estás?"
```

In [ ]:
```
"hola" not in "hola, ¿cómo estás?"
```

Podemos usar comparaciones de igualdad y de contención en otros objetos.

In [ ]:
```
'Hola' in lista_magica
```

In [ ]:
```
'Adios' in lista_magica
```

In [ ]:
```
lista_magica[-1]
```

Ejercicio: Dado un número entero n, revisa si es un gúgol.

```
In [ ]: n = 100
        print(n == int("1" + "0"*100))

        n = 10000000000000000000000000000000000000000000000000000000000000000
        000000000000000000000000000000000000
        print(n == int("1" + "0"*100))
```

AND OR

Mezclar condicionales

```
In [ ]: True & True
```

```
In [ ]: True & False
```

```
In [ ]: False & False
```

```
In [ ]: True | True
```

```
In [ ]: True | False
```

```
In [ ]: False | False
```

IF ELSE

Nos permite realizar una acción si se cumple alguna condicional. ¡En Python los espacios/tabulaciones importan!

```
In [ ]: if 1 == 1:
            print("eso es una tautología")
        else:
            print("eso es una contradicción")
```

```
In [ ]: if 1 == 1:
            print("eso es una tautología")
        else:
            print("eso es una contradicción")
```

Podemos usar cualquier cosa que regrese Verdadero o Falso.

```
In [ ]: if "Hola" in lista_magica:
            print("sí está")
```

In [ ]:
```
if "Hola" in lista_magica:
    if 1 in lista_magica:
        print("están los dos")
```

In [ ]:
```
if ("Hola" in lista_magica) & (10 in lista_magica):
    print("están los dos")
```

Ejercicio: Escribe código para, dado un número entero `n`, si es un gúgol imprima "éxito" y si no imprima "fracaso".

In [ ]:
```
n = 10000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
if n == int("1" + "0"*100):
    print("Éxito")
else:
    print("Fracaso")
```

Ciclos

Los ciclos nos permiten repetir una acción muchas veces.

- `for` : nos permite iterar sobre un objeto
- `while` : nos permite repetir una acción hasta que una condición se cumpla.

Un ejemplo de objeto iterable es obtenida con la función `range`

In [ ]:
```
range(0,10)
```

In [ ]:
```
type(range(0,10))
```

In [ ]:
```
for i in range(0, 5):
    print(i)
```

Una lista también es un objeto iterable.

In [ ]:
```
for a in lista_magica:
    print(a)
```

Para usar `while`, necesitamos una condición.

In [ ]:
```
n = 0
while n < 5:
    print(n)
    n = n + 1
```

In [ ]:
```
n
```

Ejercicio:

1) Crea un ciclo que, a partir de una variable entera `n = "1"` , le agregue ceros hasta llegar a un gúgol.

2) Usa tu condición anterior para ver si lo lograste

In [ ]:
```
n = "1"

while n != "1" + "0"*100:
    n += '0'



# for
n = "1"
for i in range(100):
    n += '0'


if n == "1" + "0"*100:
    print("Éxito")
else:
    print("Fracaso")
```

In [ ]:
```
lista_numeros = []

for i in range(0,10):
    lista_numeros.append(i)

print(lista_numeros)
```

In [ ]:
```
auxiliar = [1,2,3,4]
lista_numeros = []

for i in auxiliar:
    lista_numeros.append(i)

print(lista_numeros)
```

También podemos crear listas directamente con un ciclo de la siguiente manera.

In [ ]:
```
[i for i in range(0,10)]
```

In [ ]:
```
[i*2 for i in range(0,10)]
```

In [ ]:
```
[x for x in lista_magica]
```

Funciones

In [ ]:
```
def ponle_ceros(n, numero_ceros):
    n_obj = str(n)

    while (n_obj != str(n) + "0"*numero_ceros):
        n_obj += '0'
    return int(n_obj)
```

In [ ]:
```
print(ponle_ceros)
```

In [ ]:
```
ponle_ceros(1, 100)
```

In [ ]:
```
ponle_ceros(2,10)
```

Ejercicio: Crea una función que reciba una lista e imprima todos los elementos de la lista.

In [ ]:
```
def imprime_lista(lista):
    for i in lista:
        print(i)
```

In [ ]:
```
imprime_lista(lista_magica)
```

Ejercicio:

1. Crea una lista con los nombres de tus compañeros.
2. Crea una función que, dada una lista de nombres, imprima por cada una de las iniciales cuántas personas comparten esa inicial.

In [ ]:
```
lista_nombres = [
    'Carlos',
    'Erick',
    'Irving',
    'Cynthia',
    'Estefania',
    'Armando'
]


def cuenta_ini(lista):
    lista_iniciales = [x[0] for x in lista]
    lista_unica = list(set(lista_iniciales))
    for inicial in lista_unica:
        n = 0
        for nombre in lista:
            if nombre[0] == inicial:
                n += 1
        print(n," nombres comienzan con la letra ", inicial)


# def iniciales_únicas(lista):
#     lista_iniciales = [x[0] for x in lista]
#     return list(set(lista_iniciales))

# def cuenta_inicial(inicial, lista):
#     n = 0
#     for nombre in lista:
#         if nombre[0] == inicial:
#             n += 1
#     print(n," nombres comienzan con la letra ", inicial)


# def imprime_cuenta(lista_unica, lista_repetida):
#     for inicial in lista_unica:
#         cuenta_inicial(inicial, lista_repetida)


# def cuenta_iniciales(lista_nombres):
#     lista_iniciales_unica = iniciales_únicas(lista_nombres)
#     imprime_cuenta(lista_iniciales_unica, lista_nombres)

# cuenta_iniciales(lista_nombres)
```

In [ ]:
```
iniciales_únicas(lista_nombres)
```

In [ ]:
```
cuenta_ini(lista_nombres)
```

Diccionarios

In [ ]:
```
{}
```

In [ ]:
```
type({})
```

In [ ]:
```
dicc = {'llave': 'algún valor'}
```

In [ ]:
```
dicc
```

In [ ]:
```
dicc['llave']
```

In [ ]:
```
calificaciones = {
    'fulanito': 7,
    'perenganita': 9,
    'Juan': 7,
    'Gaby': 9,
    'Irving': None
}
```

In [ ]:
```
calificaciones['fulanito']
```

Los diccionarios tienen dos listas principales: llaves y valores.

In [ ]:
```
calificaciones.keys()
```

In [ ]:
```
calificaciones.values()
```

Podemos iterar por las llaves para encontrar los valores.

In [ ]:
```
for k in calificaciones.keys():
#     print(k)
    print("La calificación de", k, "es", calificaciones[k])
```

In [ ]:
```
def cuenta_ini_dicc(lista):
    dicc_ini = {}
    lista_iniciales = [x[0] for x in lista]
    lista_unica = list(set(lista_iniciales))
    for inicial in lista_unica:
        n = 0
        for nombre in lista:
            if nombre[0] == inicial:
                n += 1
        dicc_ini[inicial] = n
    return dicc_ini
#         print(n," nombres comienzan con la letra ", inicial)
```

In [ ]:
```
cuenta_ini_dicc(lista_nombres)
```

Archivos

Lectura

En Python, para leer un archivo debemos abrirlo. Existen varias opciones según no que queramos hacer al momento de abrir el archivo.

- r.- read. Sólo para leer
- a.- append. Agrega las líneas a un archivo
- w.- write. Escribir

In [2]:
```
archivo = open('datos/nombres.txt')
```

In [3]:
```
type(archivo)
```

Out[3]:
```
_io.TextIOWrapper
```

In [6]:
```
archivo = open('datos/nombres.txt')
nombres = archivo.read()
archivo.close()
```

Los saltos de línea son codificados como "\n".

In [7]:
```
nombres
```

Out[7]:
```
'Carlos\nErick\nIrving\nCynthia\nEstefania\nArmando'
```

In [8]:
```
print(nombres)
```

```
Carlos
Erick
Irving
Cynthia
Estefania
Armando
```

Podemos tomar ese string y dividirlo usando `split` .

In [9]:
```
nombres.split('\n')
```

Out[9]:
```
['Carlos', 'Erick', 'Irving', 'Cynthia', 'Estefania', 'Armando']
```

Para leer individualmente desde el principio podemos usar `readlines` .

In [13]:
```
archivo = open('datos/nombres.txt')
nombres = archivo.readlines()
archivo.close()
```

In [14]:
```
print(nombres)
```

```
['Carlos\n', 'Erick\n', 'Irving\n', 'Cynthia\n', 'Estefania\n', 'Arma
ndo']
```

Para no estar manejando archivos abiertos, podemos usar `with`

In [15]:
```
with open('datos/nombres.txt') as f:
    nombres = f.readlines()

print(nombres)
```

```
['Carlos\n', 'Erick\n', 'Irving\n', 'Cynthia\n', 'Estefania\n', 'Arma
ndo']
```

Escritura

Para escribir, usamos el método `write` .

In [16]:
```
with open('datos/resultado.txt') as f:
    f.write('Hola')
```

```
---------------------------------------------------------------------
------
UnsupportedOperation                         Traceback (most recent call
last)
<ipython-input-16-73a2ade2804b> in <module>()
      1 with open('datos/resultado.txt') as f:
----> 2     f.write('Hola')

UnsupportedOperation: not writable
```

El archivo no existe, tenemos que poner la opción para escribir.

In [21]:
```
with open('datos/resultado.txt', 'w') as f:
    f.write('Hola')
```

In [22]:
```
with open('datos/resultado.txt', 'a') as f:
    f.write('Adios')
```

In [20]:
```
with open('datos/resultado.txt', 'w') as f:
    f.write('Adios')
```

Para escribir en líneas separadas hay que escribirlas explícitamente

In [23]:
```
with open('datos/resultado.txt', 'w') as f:
    f.write('Hola\n')
    f.write('Adios')
```

Ejercicio: Usando el diccionario `calificaciones` . Escribir un archivo que por cada persona en el diccionario escriba una línea que diga: "La calificación de `nombre` es `calificación` ".

In [ ]:
```
with open('datos/califs.txt', 'w') as f:
    for k in calificaciones.keys():
        f.write("La calificación de " + k + " es " + str(calificacion
es[k]) + '\n')
```

Ejercicio (Tarea):

- ¿Cuántas palabras tiene la Ilíada?
- ¿Cuántas de ellas son únicas?
- ¿Cuáles son las 20 palabras más usadas?