Sistemes de Gestió Empresarial

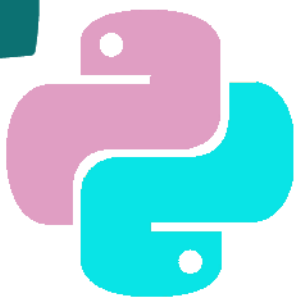
2ºGS DAM

Professor: Sergi García

02/11/2021

Python

Cassandra Sowa Candela



índex

Introducción

En este documento vamos a ver unas cuantas realizadas y sus ejemplos en Python. Vamos a comentar nuestro código y para enseñar algunos de los ejemplos que hemos realizado vamos a usar un pluggin llamado AREPL el cual nos muestra el resultado de nuestro código sin la necesidad de ejecutarlo.

Ejercicio 1

En el primer ejercicio tenemos unas cuantas cosas que hemos realizado y vamos a ver poco a poco que hemos hecho y comprobar que funciona.

```
# Creating a list
listaVieja = [1,2,3,4,5,6,7,8,9]
# Copying the list
nuevaLista = listaVieja[:]
# Adding an element into the new list
nuevaLista.append(10)
# Deleting an element from the list. Here we are deleting the element at the position 0
nuevaLista.pop(0)
#Creating a new list with the last 4 elements of listaVieja
Lista3 = listaVieja[:4:-1]
#Turning the words from a chain (separated by space) into a list
ejemplo = "This is just an example"
ejemplo2 = ejemplo.split()
```

il·lustració 1 Ejercicio 1

Diferència entre “shallow copy” i “deep copy”

Shallow copy compartix la mateixa memòria del que còpia mentres que deep copy guarda els valors en una memòria nova.

Ejercicios en Visual Studio Code

A continuación voy a explicar que hice en los casos anteriores y a demostrar que el código funciona.

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

il·lustració 2 Creación de una lista

Al principio hemos creado una lista y le hemos añadido valores. A esta lista la hemos llamado “listaVieja”.

```
nuevaLista = listaVieja[:]
```

il·lustració 3 Clonación de lista

A continuación hemos creado una lista nueva y hemos copiado los valores de “listaVieja”. A continuación podemos ver los resultados dentro de ambas listas.

```
listaVieja:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
nuevaLista:
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

il·lustració 4 Comprobación de la clonación

```
nuevaLista.append(10)
```

il·lustració 5 Añadir elemento a una lista

Con el comando “append” podemos añadir un nuevo valor a nuestra lista. En nuestro caso hemos añadido a “nuevaLista” el valor “10”.

```
nuevaLista.pop(0)
```

il·lustració 6 Eliminar elemento de una lista

Con el comando “pop” podemos eliminar un elemento de la lista, para ello tenemos que hacer referencia a la posición del elemento que queremos eliminar. La lista llamada “nuevaLista” se ve así:

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

il·lustració 7 Resultado de la lista “nuevaLista”

```
Lista3 = listaVieja[:4:-1]
```

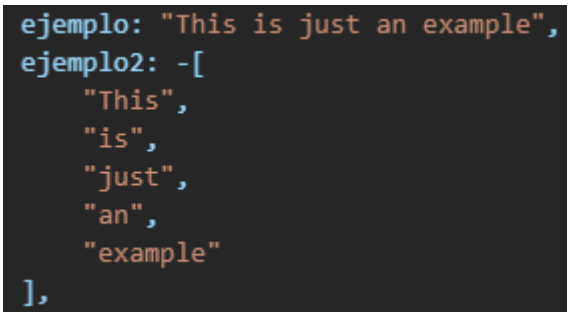
il·lustració 8 Creación de una nueva lista con los últimos 4 elementos de otra.

Para crear una nueva lista añadiendo los últimos 4 elementos de una lista tenemos que añadir “[:4:-1]”. El 4 hace referencia a cuantos elementos vamos a copiar y el -1 hace referencia a que vamos a empezar por el final.

```
ejemplo = "This is just an example"
ejemplo2 = ejemplo.split()
```

il·lustració 9 Convertir las palabras de una cadena en una lista

Para finalizar hemos creado una variable con una frase de ejemplo y hemos creado una lista con cada palabra.



il·lustració 10 Comprobación de como se han guardado las palabras

Ejercicio 2

Para comenzar creamos una función a la cual le pasamos 2 números y devuelve una suma.

```
def suma(a, b):
    return a+b

variable = suma(5,2)
```

il·lustració 11 Recibir 2 números y devolver su suma

A continuación Hemos creado otra función la cual multiplica por 2 cada valor dentro de la lista.

```
list = [1,2,3,4,5]
def double_array(list):
    for i in range(len(list)):
        list[i] = list[i] * 2

double_array(list)
```

il·lustració 12 Doblar el valor de los elementos de una lista

Para finalizar hemos creado una función similar a la anterior pero con la diferencia de que los valores multiplicados serán añadidos a una nueva lista.

```
oldList = [1,2,3,4,5]
DoubledList = []

def double_array1(oldList):
    for i in oldList:
        # Here we are making the operation, multiplying each value
        i = i * 2
        # We are printing the result so we can see it
        print(i)
        # With this line we are adding the new value into the list
        DoubledList.append(i)

# We call the function:
double_array1(oldList)
```

il·lustració 13 Doblar el valor de los elementos de una lista y guardarlos en una nueva lista

Ejercicio 3

En el siguiente ejercicio comenzamos con la idea de poder guardar nombres de usuarios y contraseñas dentro de listas o diccionarios. Vamos a mostrar como lo haríamos dentro de una lista y como lo haríamos dentro de un diccionario. Además, las contraseñas estarán guardadas en un formato Hash.

Para comenzar importamos la librería “hashlib”. Además para realizar las encriptaciones de manera más rápida y cómoda vamos a crear una función para encriptar nuestras contraseñas.

```python
import hashlib

# Function to encrypt
def encryptPass(password):
    return hashlib.sha256(password.encode()).hexdigest();
```

il·lustració 14 Función para encriptar contraseñas

Ahora vamos a pasar a ver el ejemplo de como guardaríamos nombres de usuarios y contraseñas dentro de una lista.

```python
list = [
{"username1",encryptPass("password1")},
{"username2",encryptPass("password2")},
{"username3",encryptPass("password3")},
{"username4",encryptPass("password4")},
{"username5",encryptPass("password5")}]
```

il·lustració 15 Guardar nombres y contraseñas dentro de una lista

En el caso de realizarlo con un diccionario sería:

```python
dict = {
    "cassandra": encryptPass("password1"),
    "napsta": encryptPass("password2"),
    "marlon": encryptPass("password3"),
    "vida": encryptPass("password4"),
    "george": encryptPass("password5"),

}
```

il·lustració 16 Guardar nombres y contraseñas dentro de un diccionario

Las contraseñas no se guardan como el texto que les pasamos. Se guardan ya encriptadas.

```
dict: -{
    cassandra: "0b14d501a594442a01c6859541bcb3e8164d183d32937b851835442f69d5c94e",
    george: "8b2c86ea9cf2ea4eb517fd1e06b74f399e7fec0fef92e3b482a6cf2e2b092023",
    marlon: "5906ac361a137e2d286465cd6588ebb5ac3f5ae955001100bc41577c3d751764",
    napsta: "6cf615d5bcaac778352a8f1f3360d23f02f34ec182e259897fd6ce485d7870d4",
    vida: "b97873a40f73abedd8d685a7cd5e5f85e4a9cfb83eac26886640a0813850122b"
```

il·lustració 17 Resultado de las contraseñas encriptadas

En el caso de comprobar la contraseña, lo que hace la gente es comparar las contraseñas. En los sitios webs o aplicaciones lo que hacen es encriptar la contraseña que les mandes y la comparan con el valor de la contraseña encriptada que ya tienen.

```
print (dict["cassandra"] == encryptPass("casspass"))
print (dict["cassandra"] == encryptPass("password1"))
```

il·lustració 18 Comprobación de contraseñas

El primer caso nos da “False” y el segundo caso nos devuelve “True”.

Ejercicio 4

Is

```
#Example 1 how does "is" work
ejemplo1 = [1,2,3]
ejemplo1b = [3,4,5]
# using "is" equals to using "==", We just compare if its equal
resultado = ejemplo1 is ejemplo1b
resultado2 = ejemplo1 is ejemplo1
```

il·lustració 19 Ejemplo de "is"

Utilizamos "is" para comparar valores. En nuestro paso tenemos 2 listas con distintos valores. "resultado" será igual a "false" y "resultado2" será igual a "true".

Not

```
#Example 2 how does "not" work
ejemplo2 = True
#With "not" we are saying "no", we are denying.
ejemplo2 = not True
```

il·lustració 20 Ejemplo de "not"

Usamos "not" para negar. En este ejemplo, "ejemplo2" guardará de resultado "false".

In

```
#Example 3 how does "in" work
ejemplo3 = [2,3,4,5,6]
#We use in to say "in case 4 is inside the list"
booleanoEjemplo = 4 in ejemplo3
```

il·lustració 21 Ejemplo de "in"

En este ejemplo estamos usando "in" para buscar dentro de la lista. Estamos buscando si 4 se encuentra dentro de la lista. Por ello el resultado que obtendremos será "true"

Ejercicio 5

Para pasar varios parametros por consola es tan sencillo como útilizart la orden “input”

```
# Int
age = input("What's your age?")
# String
name = input("Whats your name?")
```

il·lustració 22 Pasando parámetros distintos por consola

En el siguiente caso podemos ver como sobrecargamos una función, con *parameters no especificamos cuantos datos vamos a pasar a la función.

```
def add (*parameters):
    sum = 0
    for i in parameters:
        sum+=i
    return sum;

print( add(1,2))
print( add(1,2,6,2,4))
print( add(1,2,4))
```

il·lustració 23 Sobrecarga de una función

Ejercicio 6

En el siguiente ejercicio hemos creado una lista que contiene altura y peso.

```
list = [(1.56,65),
(1.93, 99),
(1.75,64),
(1.56,46),
(1.84,75)]
```

il·lustració 24 Lista con alturas y pesos

Nos interesa ordenar esta lista por altura, mostrando primero la mayor altura. En el caso de que la altura sea la misma el programa mostrará primero quién pese menos.

```
def sortCondition(element):
    return -element[0], element[1]

# To sort the list we use "sort" and we use "key" to say which values we would like
list.sort(key = sortCondition)
print(list)
```

il·lustració 25 Ordenando la lista

Para realizar esto creamos una función y elegimos la posición en la lista que mostrará. Al añadir “-“ delante de “element” estamos transformando todos los valores en negativos, por ello, si tenemos un 4 y un 2, al añadir “-“ se convertirán en “-4” y “-2” por ello, 4 se mostrará como el valor más pequeño.

Para organizar la lista tenemos que usar el comando “sort” y para definir como queremos organizar la lista usaremos “key”.

Què és Key?

Key és una funció que torna un valor per a ajudar-nos a ordenar llistes.

Ejercicio 7

Para el último ejercicio hemos creado una clase llamada “Car” y le hemos añadido de atributos el color y la matrícula (licencePlate)

Con “self” hacemos referencia, en java sería “this”.

También hemos añadido 3 metodos distintos.

El primer método imprime la matricula y el color del coche.

El siguiente método sirve para cambiar manualmente el color del coche

El último metodo te devuelve que el coche tiene sólo 4 puertas.

```
class Car:

    def __init__(self, licencePlate, color):
        self.licencePlate = licencePlate
        self.color = color

    def print(self):
        print (str(self.licencePlate) + " "+ self.color)

    def changeColor(self, color):
        self.color = color

    def getNumberOfDoors(self):
        return 4
```

il·lustració 26 Creación de una clase con parámetros y métodos

Para continuar añadimos los posibles colores que se pondrán de manera aleatoria a los coches e hicimos que le aparezca al usuario una pregunta de cuantas instancias de coches quiere crear. Dependiendo de las isntancias que se pongan se irán añadiendo las matrículas y el color aleatorio de cada coche.

Para elegir de manera aleatoria tendremos que importar la librería “random”.

```
def __main():
    listOfCars = []
    colors = ["red", "blue", "pink", "black", "white"]

    n = int(input("Please enter the number of car instances:").strip());
    for i in range (n):
        licencePlate = i+1
        color = colors[random.randint(0, len(colors) - 1)]
        listOfCars.append(Car(licencePlate, color))
    print10Instances(listOfCars)
```

il·lustració 27 Añadiendo matrículas y color de manera aleatoria

Para finalizar vamos a imprimir los 10 primeros coches en caso de que tengamos más de 10 coches y en caso de tener menos pues mostraremos todos ellos.

```
def print10Instances(listOfCars):
    limit = min(len(listOfCars), 10)
    for i in range(limit):
        listOfCars[i].print();
```

il·lustració 28 Mostrando coches por pantalla