Python

@edervs04

edervs

Creado en 1991 por Guido Van Rossum

Actualmente está en la versión 3.8

Multi-Paradigma: Orientado a Objetos, Estructura, Funcional y Logíca (con magia).

Interpretado. Tipado dinámico. No puntos y coma (;).

Altamente extensible con un núcleo pequeño y todo se basa en sus bibliotecas estándar.

Tipos de Variables

Tipos de variables: Todos los que ya conocemos y:

Set: {1,2,3,4}

List: [1,2,3,¨a¨]

Tuple: (1,2,3)

Dict: {“age”: 20}

Operadores binarios de booleanos

True and False

False or 1

not True

1 == 1

0 != []

Operadores binarios de números

```
2 ** 10
```

```
10 // 3
```

Estructuras de Control

```
if expresión booleana:
    sentencias
```

Notemos la indentación.

4 espacios o tab.

```
while expresión booleana:
    sentencias
```

```
for variable in iterable:
    sentencias
```

Todos son foreachs

Funciones

```
def nombre(argumentos):
    sentencias
```

```
def nombre(argumento=valor):
    sentencias
```

```
def sin_definir():
    pass
```

```
try:
    sentencias
except error:
    sentencias
finally:
    sentencias
```

```
try:
    sentencias
except:
    sentencias
finally:
    sentencias
```

Ejercicio 0.0.7. Una matriz se puede representar como una arreglo de 2 dimensiones. En *Python* se puede representar como listas dentro de una lista. Se puede definir una matriz de la siguiente manera:

```
# Matriz de 3x3
matriz = [[1,1,0], [0,1,1], [1,0,1]]
```

Define una función que reciba una matriz y que regrese el espejo en vertical de esa matriz. El espejo del ejemplo anterior sería:

```
matriz_espejo = [[0,1,1], [1,1,0], [1,0,1]]
```

El código debe de estar dentro de una función y regresar una matriz que sea el espejo en vertical de la matriz que se recibe.

```
def matriz_espejo_vertical(matriz):
    # TODO: Código para regresar la matriz
    #       espejo en vertical
    return matriz_espejo
```

Programación orientada a objetos

```
class nombre:
    atributo = 0
    def __init__(self, argumento, argumento2):
        self.atributo = argumento
        self.atributo2 = argumento2
    def otro_metodo(self):
        return self.atributo
```

```python
class Animal:
    def __init__(self, edad):
        self.edad = edad

    def respirar(self):
        print("Respirando")

class Persona(Animal):
    def __init__(self, nombre, edad, planeta="Tierra"):
        Animal.__init__(self, edad)

        self.nombre = nombre
        self.respirar()
        self.presentarse()

    def presentarse(self):
        print("Mi nombre es", self.nombre,
              "tengo", self.edad, "y soy del planeta",
              self.planeta)

    def __str__(self):
        # Se debe de regresar un str
        return self.nombre + " de " + str(self.edad)

persona1 = Persona("Dexter", 10)
print(persona1)
```

Gracias

@edervs04

edervs