Seguridad y Python

Introducción básica a Python 3

Programación Orientada a Objetos

Alcance de las variables

- nonlocal: “not a global or local variable”
  - Permite el acceso a variables en funciones anidadas

PYTHON

```
global_var = 0

def method():
    def method2():
        nonlocal value # Hace referencia a la variable value de method
        global global_var # Hace referencia a la variable global
        value = 100
        global_var = 100

    value = 10
    method2()
    print(value) # 100. Si hicieramos print(global_var), también sería 100

method()
```

Clases: Inicialización, Uso

PYTHON

```
class Punto:
    def __init__(self, x, y): # Constructor: Inicializa var. de instancia
        self.x = x
        self.y = y

p = Punto(2.5, 3.0)
print(p.x)
print(p.y)

# OJO, entre def y __init__ hay un espacio!!!!
# OJO, __init__ tiene dos subrayados a cada lado (_ _ init _ _)!
```

Clases: Métodos de instancia

PYTHON

```
class Punto:
    def __init__(self, x, y): # Constructor: Inicializa var. de instancia
        self.x = x
        self.y = y

    def moverHaciaX(self, x): # Método de instancia
        self.x += x
    def moverHaciaY(self, y): # Método de instancia
        self.y += y

    def moverHacia(self, x, y): # Método de instancia
        self.moverHaciaX(x)
        self.moverHaciaY(y)

p = Punto(2.5, 3.0)
p.moverHacia (-1.0, 1.0)
```

Clases: Variables y métodos de clase

PYTHON

```
class Punto:

    instancesCreated = 0 # “Variable” de clase

    def __init__(self, x, y): # Constructor: Inicializa var. de instancia
        self.x = x
        self.y = y
        type(self).instancesCreated += 1 # Accediendo a la “Var.” de clase

p = Punto(2.5, 3.0)
p2 = Punto(1.5, 1.0)
print(Punto.instancesCreated)
print(p.instancesCreated)
```

Las “variables y métodos de clase” son un concepto complejo dentro de Python, y no se utilizarán en esta asignatura

Ver https://stackoverflow.com/questions/68645/are-static-class-variables-possible

Clases: Variables y métodos de clase

PYTHON

```
class Punto:

    def __init__(self, x, y): # Constructor: Inicializa var. de instancia
        self.x = x
        self.y = y

    @classmethod
    def classmethod(cls): # Método de clase
        return "Este metodo solo puede acceder a las variables de clase"

    @staticmethod
    def staticmethod():  # Método de instancia
        return "Este metodo no puede acceder ni a las var. de clase ni a self"

print(Punto. classmethod())
print(Punto. staticmethod())
```

Clases: Herencia

PYTHON

```
class Punto3D(Punto):

    instancesCreated = 0 # Debemos redefinir las “variables” de clase

    def __init__(self, x, y, z): # Constructor de la subclase
        super().__init__(x,y) # Llamada al constructor de la superclase
        self.z = z
        type(self).instancesCreated += 1

p = Punto(2.5, 3.0)
p3 = Punto3D(1.0, 0.0, 1.0)
print(isinstance(p, Punto)) # Is object x an instance/child of class Y?
print(issubclass(p3.__class__, Punto)) # is class X an instance/child of
class Y?
```

Estructuras (vacias)

PYTHON

```
class Estructura:
    pass

s = Estructura()
s.campo1 = 10
s.campo2 = "cadena"
s.campo3 = Punto(2.5, 3.0)
```

Iteradores

PYTHON

```
class Reverse:
    def __init__(self, data):
        self.data = data # Sobre que iteramos
        self.it = len(data) # Índice inicial del iterador
    def __iter__(self):
        return self # Sobre quien iteramos
    def __next__(self):
        if self.it == 0:
            raise StopIteration
        self.it = self.it - 1
        return self.data[self.it]

rev = Reverse('reverso')
rev.it = len(rev.data) # En nuestras clases, DEBEMOS iniciar el iterador
for char in rev:
    print(char)
```

Referencias bibliográficas

Bibliografía básica

- “Python 3 documentation”

  https://docs.python.org/3/tutorial/