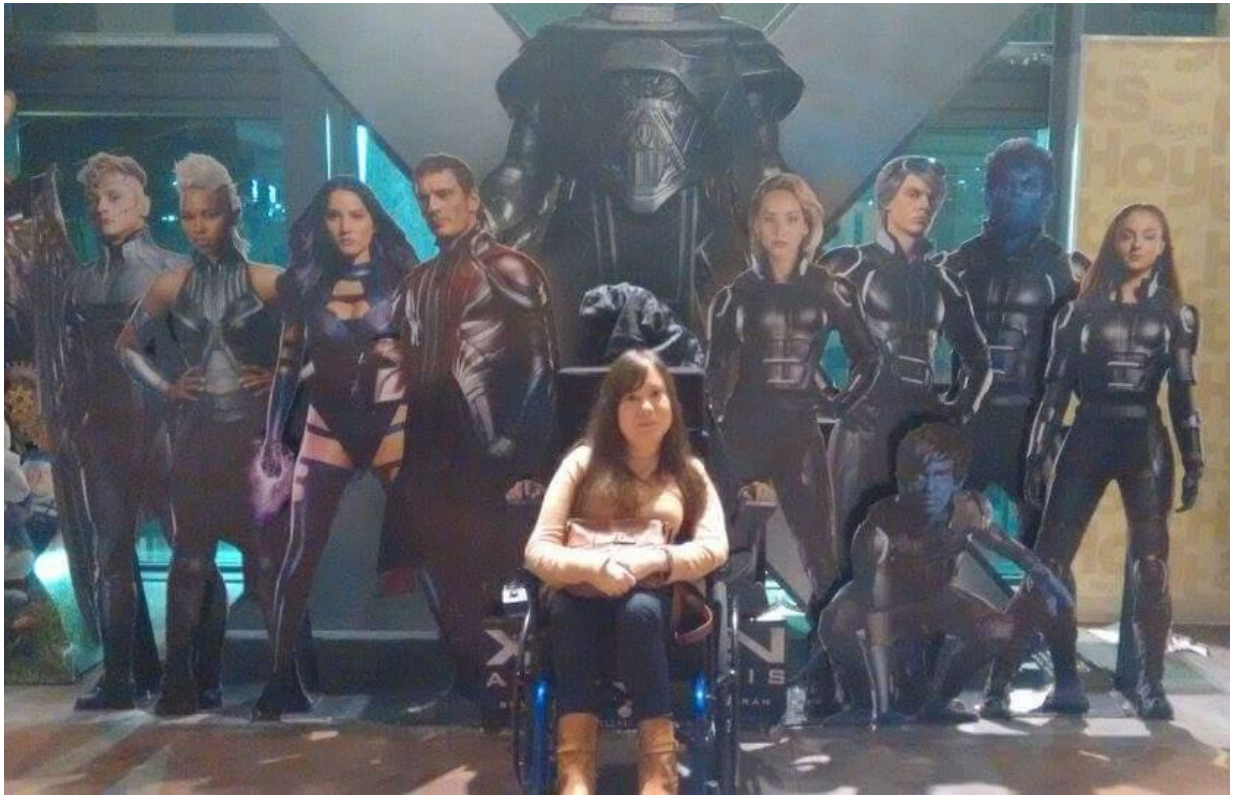
Python X

Primera Generación

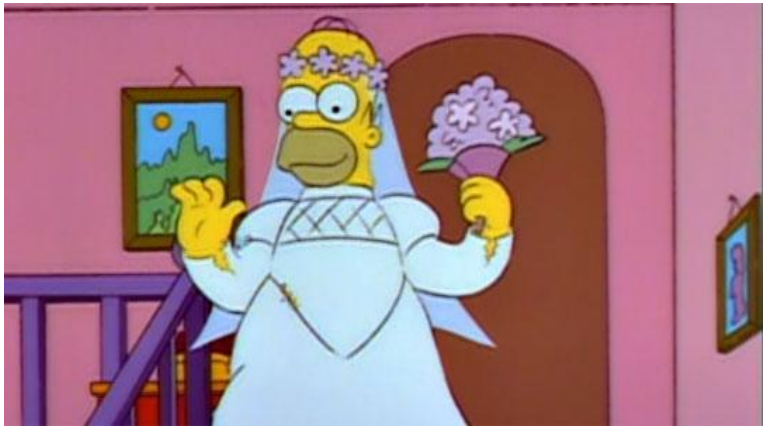
Valeria Rocha
@avaltenea



Python

- Lenguaje interpretado, de tipado dinámico pero fuerte.
- Intérprete interactivo.
- Fácil de aprender, sintaxis sencilla.
- Multiparadigma.
- Multiplataforma
- Incluye baterías. ¡Como Vale!
- Todos amamos Python.

No seas defensor de un único lenguaje. Cada uno será mejor en un contexto determinado.



Tipos de datos

Números

Enteros

```
>>> 2 + 2
4
>>> 16/2
8
```

Fracciones

```
>>> from fractions import Fraction
>>> Fraction(5,3) *2
Fraction(10, 3)
```

Flotantes

```
>>> 2 * 1.25
2.5
```

Complejos

```
>>> (2+3j) * 2
(4+6j)
```

```
>>> import math
>>> math.cos(math.pi / 4)
0.70710678118654757
```

Booleanos

True o False

Cadenas

‘Esta es una cadena’
“Esta también es una cadena”

“A la grande le puse cuca”.

Usar nombres descriptivos para las variables, por el amor de Jebus.

Cadenas: accesos

```
>>> palabra = "Pyconar"

>>> palabra[0]
'P'

>>> palabra[-1]
'r'

>>> palabra[0:5]
'Pycon'

>>>palabra[0] = m
TypeError: 'str' object does not support item assignment
```

¡Las cadenas son inmutables!

Cadenas: operaciones

```
>>> palabra = "Pyconar"

>>> len(palabra)
7

>>> 6 * 'na'
'nanananananana'
>>> 'bat' + 'man'
'batman'
```

Listas

```
>>> lista = ["Cyclope", "Wolverine", "Profesor X", "Magneto"]


>>> lista[0]

'Cyclope'


>>> lista[:]

['Cyclope', 'Wolverine', 'Profesor X', 'Magneto']
```

```
>>> lista[1] = "Tormenta"
>>> lista
['Cyclope', 'Tormenta', 'Profesor X', 'Magneto']


>>> lista.append("Jean Grey")

>>> lista

['Cyclope', 'Tormenta', 'Profesor X', 'Magneto', 'Jean Grey']



>>> lista.index("Profesor X")

2
```

Concatenar listas:

```
>>>[‘las’, ‘de’] + [‘sistemas’]
>>>[‘las’, ‘de’, ‘sistemas’]
```

List comprehensions

```
>>> lista = [1,2,3,4,5]
>>> [x **2 for x in lista]
[1, 4, 9, 16, 25]

>>> sum(x ** 2 for x in range(20))
2470
```

Diccionarios

```
>>> diccionario = {"Cyclope": "Java", "Wolverine": "Python", "Profesor X":
"Haskell", "Magneto": "Javascript"}

>>> for clave, valor in diccionario.items():
...     print(clave, valor)
...
('Wolverine', 'Python')
('Profesor X', 'Haskell')
('Cyclope', 'Java')
('Magneto', 'Javascript')
```

```
>>> diccionario["Wolverine"]
'Python'

>>> diccionario["Magneto"] = "Cobol"
>>> diccionario
{'Wolverine': 'Python', 'Profesor X': 'Haskell', 'Cyclope': 'Java', 'Magneto': 'Cobol'}

>>> diccionario.keys()
['Wolverine', 'Profesor X', 'Cyclope', 'Magneto']
>>> diccionario.values()
['Python', 'Haskell', 'Java', 'Cobol']

>>> "Bestia" in diccionario
False
```

Control de flujo

For

```
>>> lista = ['Cyclope', 'Tormenta', 'Profesor X', 'Magneto', 'Jean Grey']

>>> for mutante in lista:
...     print(mutante)

Cyclope
Tormenta
Profesor X
Magneto
Jean Grey
```

```
>>> numeros = [1,2,3,5,8]


>>> for i in range(0,5):
...     print(numeros[i])

1
2
3
5
8
```

If

```
if variable == condicion1:
        #Hacer algo
elif variable == condicion2:
        #Hacemos otra cosa
else
        #Sino hacemos esto otro
```

While

```
>>> while variable < 10:
...     print variable
...     variable += 1
...
0
1
2
3
4
```

Funciones

```
>>> def holaMundo():
...     print "Hola mutantes de la PyConAr"

>>> holaMundo()
Hola mutantes de la PyConAr

>>> holaMundo
<function holaMundo at 0x7f5dd66f47d0>
```

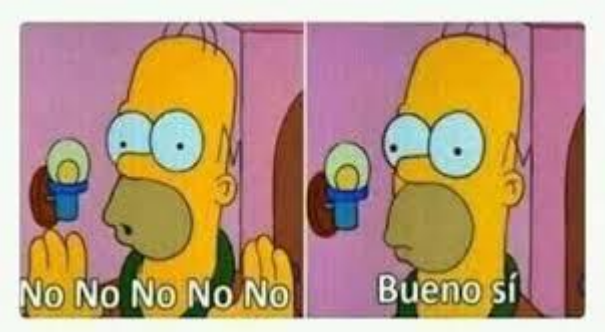
Funciones

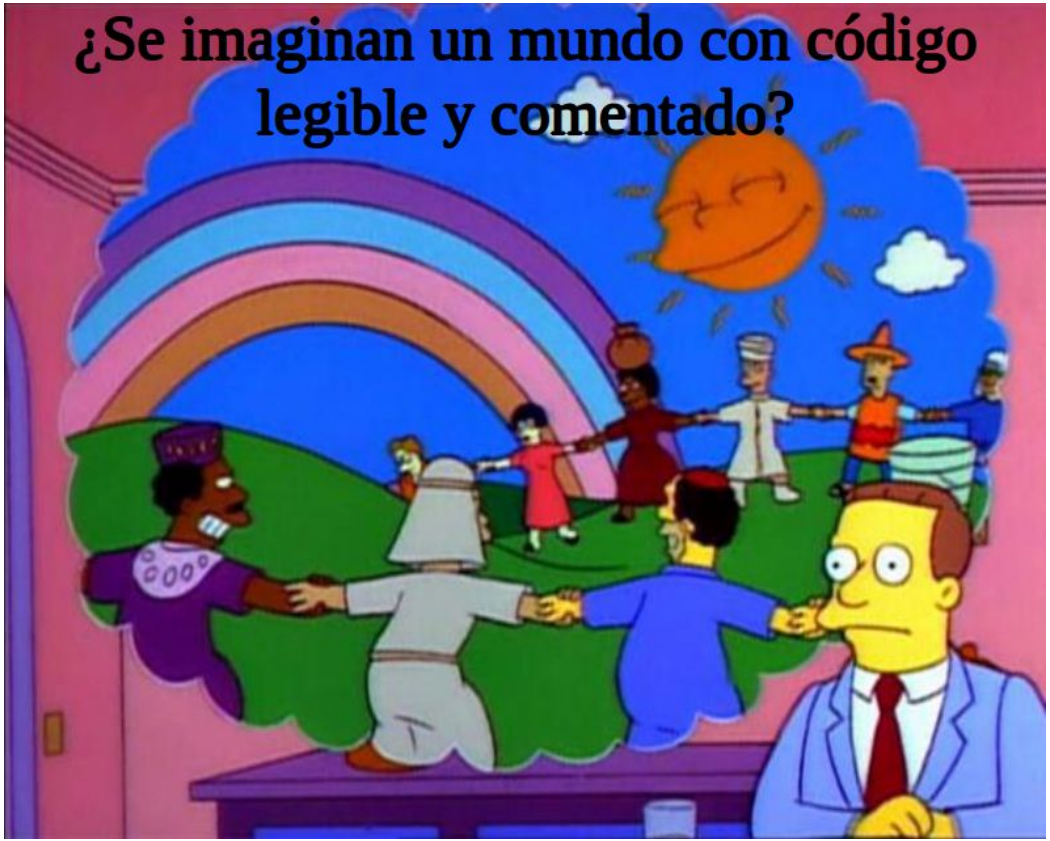
```
>>> def holaMundo():
...     print "Hola mutantes de la PyConAr"

>>> holaMundo()
Hola mutantes de la PyConAr

>>> holaMundo
<function holaMundo at 0x7f5dd66f47d0>
```

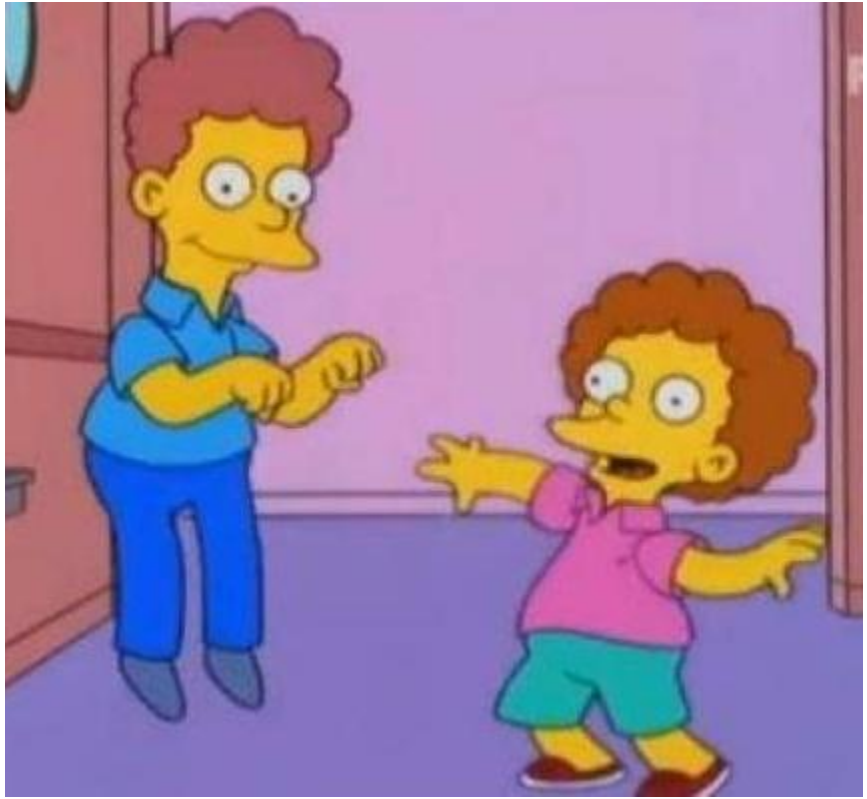
Valeria, ¿acaso has usado print como debugger?



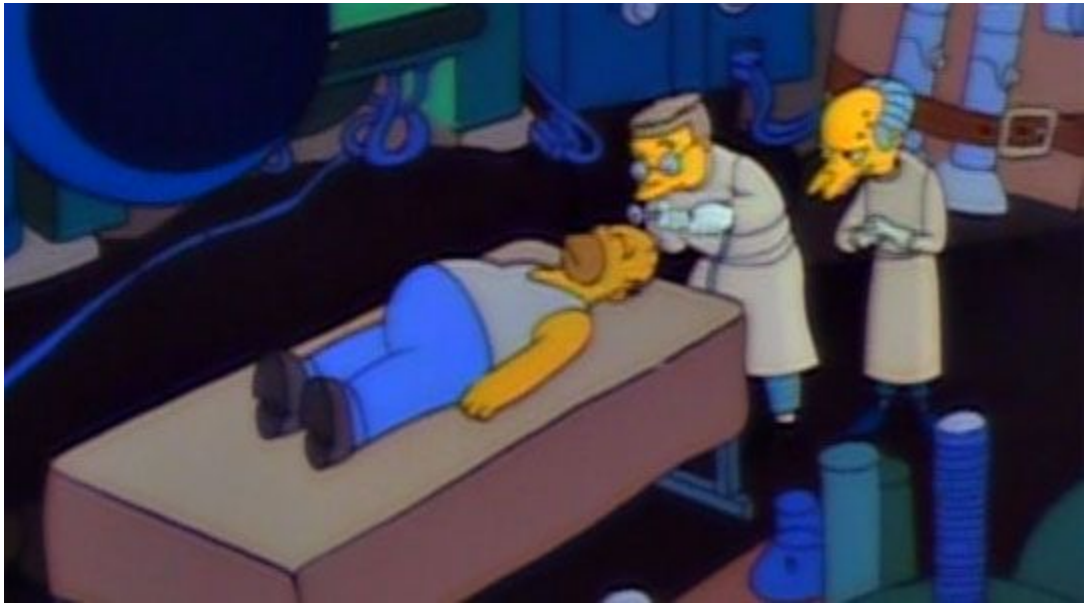

¿Se imaginan un mundo con código
legible y comentado?

Testea que tu código haga lo que tiene que hacer antes de hacer commit.
No le das una mamadera a un bebé sin antes verificar la temperatura.

Clases en Python



¡Construyamos nuestro propio mutante!



Clase Mutante

nombre: developer

experiencia: 5

reclutable: si/no

poderes:

- Programar en Python.
- Arreglar bugs.
- Romper producción.
- Resolver problemas.

```
class Mutante:

    reclutable = True

    def __init__(self, nombre, experiencia):

        self.nombre = nombre

        self.experiencia = experiencia

        self.poderes = []


    def saludar(self):

        print "Hola mi nombre es " + self.nombre


    def esReclutable(self):

        return self.reclutable == True
```

```
>>> miMutante = Mutante("Developer", 5)


>>> miMutante.saludar()

Hola mi nombre es Developer


>>> miMutante.esReclutable()

True


>>> miMutante.experiencia

5
```

```
def agregarPoder(self, poder):
    self.poderes.append(poder)

def verPoderes(self):
    for poder in self.poderes:
        print poder
```

```
>>> miMutante.agregarPoder("Resolver bugs")

>>> miMutante.agregarPoder("Programar en Python")

>>> miMutante.agregarPoder("Romper produccion")
```

```
>>> miMutante.verPoderes()

Resolver bugs

Programar en Python

Romper produccion
```

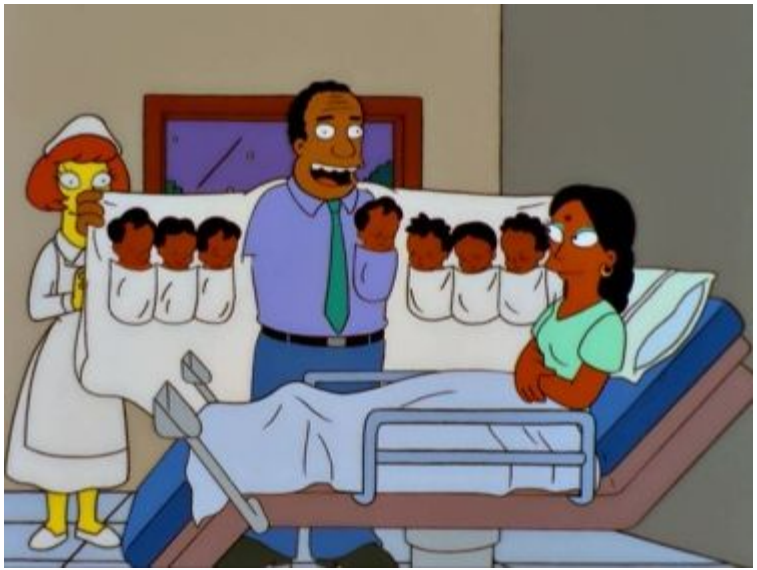
Herencia múltiple

```
class MiClase(ClasePadre):

    #Definición de la clase.
```

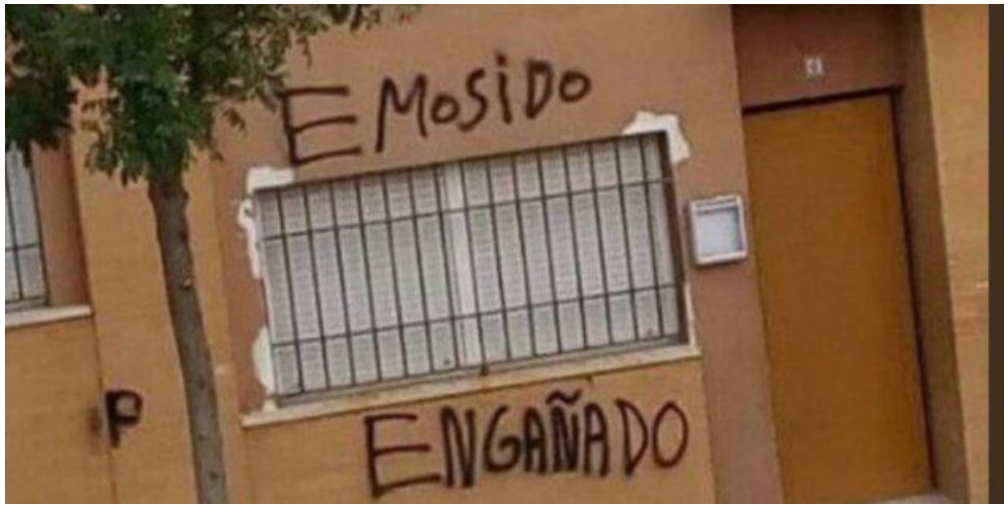
```
class MiClase(ClaseA, ClaseB):

    #Definición de la clase.
```



Módulos

```
>>> import nombrearchivo
>>> mutante = nombrearchivo.Mutante("Developer", 5)
```



Y aún hay más...

NumPy: funcionalidades matemáticas avanzadas.

PyGame: juegos, trabajar con imágenes, animaciones.

Sympy: evaluaciones algebraica, calcular números complejos.

SciPi: algoritmos y herramientas matemáticas.

Request: librería HTTP.

Llegues donde llegues, la humildad ante todo.
¡En algún momento vos tampoco sabias lo que era un for!

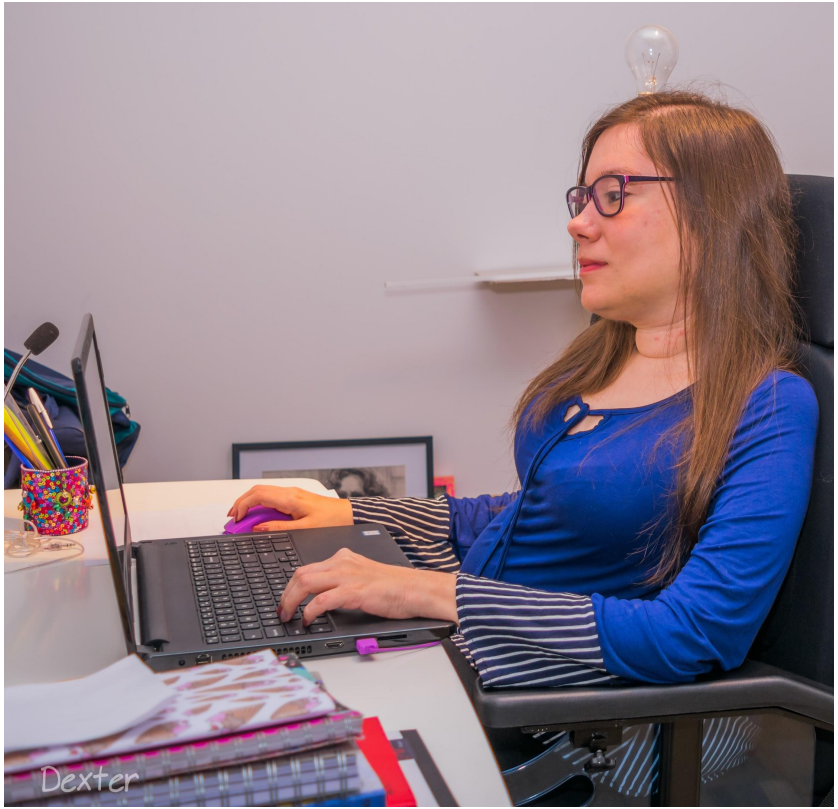
Valeria Rocha

[LAS] DE SISTEMAS

FIUBA

valeria.mrb@gmail.com

@avaltenea



¡Gracias PyConAr!

¡Gracias @lasdesistemas!

¡Gracias @cynpya!