Python

Filosofía

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

Web scripting | Web development | App scripting

Django

Big Data

Machine Learning

No backward compatibility

Python 2 | Python 3

pip | pip3

phyta | ipython | python3 | ipython3

REPL → Es el prompt en ejecuta, espera y repite, es por ejemplo la consola de javascript en donde ejecutamos una sentencia y se ejecuta y esta lista para la siguiente

Formas de ejecutar python
Interprete, REPL o notebooks

Type Hinting

Python no nos previene de pasar argumentos incorrectos, incluso si usamos type hinting

```
def (add_Numbers (a: int, b: int] -> int:
    return a + b
```

* Solo es informativo el int para nosotros pueden forzar o tratar de mandar parámetros correctos

Tipos de datos

int a → 42

float b → 3.14159

a + b = 45.14159

Se declaran sin tipo

a = 42

pi = 3.14159

podemos sumarlos sin problema

a + pi = 45.14159

Strings

' ' , " " , """ """ ,

Igual, simple quotes, double o triples

'hello'.capitalize() == "Hello"

"hello".replace("a", "a") == 'Hallo"

"hello".isalpha() → TRUE si todos son digitos

"hello".isdigit()

r"some,csv,values".split(",") == ["some",
"csv",
"value]

Formateo Strings

name = "Jesus"

machine = "J"

"Nice to meet you {0}. I am {1}".format(
name,
machine)

f"Nice to meet you {name}. I am {machine}"

↻ Si se usa string interpolation debemos escapearlo por ejemplo con f

---

True False → empiezan con T o F en mayuscula

si tenemos un True y lo pasamos a int = 1

si tenemos un False y lo pasamos a int = 0

si tenemos un True y lo pasamos a string = "True"

si tenemos un False y lo pasamos a str = "False"

None → similar a null

---

if - else

number = 5

if number == 5 : importar

print("Num=5")

En pyta tenema valoa, Truthy o Falsy

Falsy	Truthy
0	1, -1
""	"a"
None	True
False	

Operado ternario

"True" if x>10 else "False"

Solo hay if-else, no hay if-else if

Listas

Declaramos una lista vacia

estudiantes = [ ]

Declarama y asignama valoa a la lista

estudiantes = ['Jesus', 'Jose', 'Miguel']

Podemos asigna valoa de todos los tipos

estudiantes = ['Jesus', 14, None, True]

← Se recomienda no mezcla tipos d lista, incluso si python nos lo permite

¿Si podema pone None como valor d lista, como elimnama un elemento?

del estudiantes[0]

Ayugama un elemento al final d la lista

estudiantes.append('Elmit')

Algo muy util es acceder a elemento da l final d la lista con -1

Lista = [1, 2, 3, 4]

-4 -3 -2 -1 → si accedema con -1 accedema al 1 eve.

Podemos hacer algo como subindex con listas

Ejemplo

estudiantes = ['Jesus', 'Miguel', 'Erika', 'Lupita']

estudiantes[1:] → se saltea el primero y solo nos regresa el 1 - 3

estudiantes[-1:] → se salta el ~~primero~~ ultimo resto

estudiantes[1:-1] → se salta el primero y el ultimo regresa

len(estudiantes) → nos da la longitud

"Jesus" in estudiantes → True si esta en la lista

---

loops

Tenemos dos tipos de ~~loop~~ for.

For each-like

estudiantes = [...]

1) for student in estudiantes:
    print("Hola {}".format(student))

2) Con index

longitud iteraciones

index = 0
for index in range(0, 10, 1)

0 → start point

1 → Decide cuando sumar despues de cada iteracion

* Podemos usar el range con 1, 2 o 3 arg.
  1 → Solo num elemento
  2 → Num elem y inicio
  3 → Num elem, inicio, cuanto de salto

Diccionarios

```
Persona = {"nombre": "Juan", "edad": 34
           "ciudad": None}
```

- Podemos tener los tipos que queramos al igual que en las listas

```
personas = [{"nombre": "Juan", "edad": 34},
            {"nombre": "Ana", "edad": 34}]
```

- Podemos tener listas de mapas

```
persona.get("nombre")
persona["nombre"]
```

> Nos trae el valor de persona.nombre

```
persona.get("nombres", "Unknown")
```

↓

En este caso nos como no tenemos definido nombres, nos daría Unknown

---

Excepciones

```
try:
    ....
except Tipo:
    ....
```

```
try
    intSum = 3 + "Hello"
except Exception as error:
    Print(error)
```

Otros tipos de datos

complex

long # solo en python 2

bytes y bytearrays

tuple=(3, 5, 1, "Mark") → lista inmutable

set and frozenset

set ([3, 2, 3, 1, 5]) → 1, 2, 3, 5

→ unicamente y order

Functions

```
def nombreFuncion (params):
    body
    :
    return some
```

- Can no have any
- Can return
- Can have any.

*Los parametros tiene scope de bloque, en este caso el bloque funcion

Podemos recibir varios o ningun parametro → resulta en una lista

```
def functionEx (*vargs)
```

Podemos recibir kvargs, esto resulta en un mapa

```
def functionEx (**kvargs)
```

↓

Se llama

```
functionEx (name = "Jesus",
            age = 34,
            married = False)
```

Podemos tener valores default en nuestras funciones

```
def functionEx (name, age = 18)
```

default value

* Si no mandamos el default

Closure

Nested function, la función interna puede tener acceso a las variables de la outer function.

---

input ("Escribe tu nombre") → pide al usuario que introduzca su nombre mediante un prompt

open ("File", 'mode') → ex open ("Student.txt", 'a') → append

write (student) → escribir en archivo

close() → cerrar para no tener memory leak

readlines()

Yield ? *

---

lamda function

regular function

```
def double(x):
    return x * 2
```

lamda

```
double = lamda x: x*2
```

Anonimous Function

+Encadenamos lamda, map, filter, reduce

Se llama igual

double(10) ==20

Clase vs Función

Clase podemos definir como misma objeto, función no

Organizamos mejor el código en una clase

Declare una clase

class Student: → Para decir que la clase no hace nada, su body estará vacío
pass → Keyword reservada para la clase

student = Student()

new Student = Student()

> Se comporta igual que Java, son instancias diferentes es decir apuntan a espacios diferentes de memoria

---

Constructor se define como sigue

__init__

to String se define como sigue

__str__

Instance attributes → Se refieren a self, cada instancia es diferente self

class attributes → Serían en Java las variables estáticas, incluso las podemos acceder sin necesidad de una instancia

Herencia
class Student:
⋮
class HighSchoolStudent(Student) ← De donde heredo

En herencia podemos usar la siguiente

super()
self() > Ver si es igual que Java que debe ser la primera línea

sobre escribir metodo

Modulos son como paquetes

from file.py import clase

Si solo ponemos import tenemos que referenciar de donde usamos con el fully qualified Name

file.clase

Comentarios

multiline
"""
comen
"""

→ Se usa dentro del cuerpo del metodo, no arriba, esto para hacer algo similar al javadoc

unline #

Flask → python package

para instalarla necesitamos pip

* pip install flask

pip igual a npm de node

+ Para ver el maven de python usamos PyPI

http://pypi.python.org/pypi

Indexes de python package index

@app.route("/", methods=["GET", "POST"])

↓

Similar a Spring:

@GetMapping("/")

@PostMapping("/")

request.form.get("nombre campo")

* ~~Decorator~~
* los atributos de una clase por default se vuelven un mapa, esto se puede cambiar pero es un tema que no se toca en este punto del curso

---

Tips Tricks python

Virtual Environment

- Similar al concepto de docker a Virtual machine
- Es como si tuviéramos versión 7 y versión 8 de Java en una misma ide
- Para instalarlo:

pip install virtualenv

una vez instalado:

virtualenv <env-name>

ejemplo:

virtualenv pluralsight ← es como el nombre para

virtualenv --python=python3.7 pluralsight

versión de este virtual environment