Python professional

Módulo

Un módulo es cualquier archivo de Pyhton. Generalmente, contiene código que puedes reutilizar.

Paquete

Una carpeta que contiene módulos. Siempre posee el archivo **init**.py

Cómo organizar un proyecto python

```
|-README
|-.gitignore
|-venv
|-package
  |-module
  |-module
  |-module
```

Tipado

Los tipados es una clasificación de los lenguajes de programación, tenemos cuatro tipos:

- Estático
- Dinámico
- Débil
- Fuerte

> El tipado del lenguaje depende de cómo trata a los tipos de datos.

El tipado estático es el que levanta un error en el tiempo de compilación. El tipado dinámico levantan el error en tiempo de ejecución.

El tipado débil es el que hace un cambio en un tipo de dato para poder operar con el.

> Python es un lenguaje de tipado 🐰 Dinámico y 💪 Fuerte

Como ejecutar mypy

```
mypy {filename}.py --check-untyped-defs
```

Closures

Es una forma de acceder a variables de otros scopes a través de una nested function. Se retorna la nested function y esta recuerda el valor que imprime, aunque a la hora de ejecutarla no este dentro de su alcance.

Reglas para encontrar un closure:

1. Debemos tener una nested function
2. La nested function debe referenciar un valor de un scope superior
3. La función que envuelve la nested debe retornarla también

> Cuando tenemos una clase que tiene solo un método Cuando trabajamos con decoradores

Ejemplo:

```
def make_multiplier(x):

  def multiplier(n):
    return x * nested

  return multiplier

times10 = make_multiplier(10)
times4 = make_multiplier(4)

print(times10(3)) #OUTPUT: 30
print(times4(5)) #OUTPUT: 20
print(times10(times4(2))) #OUTPUT:80
```

Decorador

Función que recibe como parámetro otra función, le añade cosas, y retorna una función diferente.

Patrón común:

```
def decorator(func):
  def envoltura():
    print('Esto se añade a mi función original')
    func
  return envoltura

def saludo():
  print('¡Hola!')
saludo = decorator(saludo)

saludo()
```

Solución built-in:

```
def decorator(func):
  def envoltura():
    print('Esto se añade a mi función original')
    func()
  return envoltura

@decorator
def saludo():
  print('¡Hola!')

saludo()
```

Ejemplo:

```
def mayusculas(func):
  def envoltura(texto):
    return func(texto).upper()
  return envoltura

@mayusculas
def mensaje(nombre):
  return f'{nombre}, recibiste un mensaje'

print(mensaje('Cesar'))
```

Sets

Una colección desordenada de elementos únicos e inmutables.

Operaciones básicas de sets

- Unión: |
- Intersección: &
- Diferencia: -
- Diferencia simétrica: ^

Datetime

Abreviaciones

Código	Significado
%Y	Year
%m	Month
%d	Day
%H	Hour

Código	Significado
%M	Minute
%S	Second

Ejemplo

```
from datetime import datetime

my_datetime = datetime.now()
print(my_datetime)

my_str = my_datetime.strftime("%d/%m/%Y")
print(my_str)
```