Clase0

October 7, 2020

1 Bienvenidos la curso de Python Intermedio A

1.1 Repaso rápido de lo básico de Python

Lo primero que debemos hacer es decargar python. Lo podemos hacer de la página oficial o desde el source.

Página oficial

Source code

La documentación que vamos a usar en todo el curso es la documentación oficial de Python(https://docs.python.org/es/3/

1.1.1 Comentarios

[1]:
```
# comentario de una linea

"""comentario
de varias
lineas
.... sigue..."""
```

[1]:
```
'comentario\nde varias \nlineas\n… sigue…'
```

1.1.2 Variables y tipos de datos

[2]:
```
variable1 = "esto es un string"
variable2 = 123  # integer
variable3 = 2.4  # float
variable4 = True  # o False boolean
variable5 = None  # Nada o null

variable_lista = [1, "hola", 2.3, False]
variable_tupla = (1, "2", 2.3, False)
variable_dict = {"key1": 2, "key2": {"otra_key": [1,32,"2"]}}
variable_set = {"a", "b", "c"}
```

1.1.3 Tipos de datos mutables e inmutables

Algunos tipos de datos en Python son mutables que significan que su valor puede cambiar en tiempo de ejecución. Los inmutables, no pueden cambiar.

1.1.4 Operadores aritméticos

```
+ Suma
- Resta
* Multiplicación
** Exponente
/ División
// División entera
% Módulo
```

1.1.5 Flujo de Control

OPERADORES RELACIONALES (DE COMPARACIÓN)

Símbolo	Significado	Ejemplo	Resultado
==	Igual que	5 == 7	Falso
!=	Distinto que	rojo != verde	Verdadero
<	Menor que	8 < 12	Verdadero
>	Mayor que	12 > 7	Falso
<=	Menor o igual que	12 <= 12	Verdadero
>=	Mayor o igual que	4 >= 5	Falso

OPERADORES LÓGICOS

Operador	Ejemplo	Resultado*	
and (y)	5 == 7 and 7 < 12	0 y 0	Falso
	9 < 12 and 12 > 7	1 y 1	Verdadero
	9 < 12 and 12 > 15	1 y 0	Falso
or (o)	12 == 12 or 15 < 7	1 o 0	Verdadero
	7 > 5 or 9 < 12	1 o 1	Verdadero

if

[9]:
```
x = int(input("Please enter an integer: "))
if x < 0:
    x = 0
    print('Negative changed to zero')
elif x == 0:
    print('Zero')
elif x == 1:
    print('Single')
else:
```

```
    print('More')
```

```
Please enter an integer: 10
More
```

for

[10]:
```
words = ['cat', 'window', 'defenestrate']
for w in words:
    print(w, len(w))
```

```
cat 3
window 6
defenestrate 12
```

while

[11]:
```
year = 2001
while year <= 2020:
    print(f"Informes del Año {year}")
    year += 1
```

```
Informes del Año 2001
Informes del Año 2002
Informes del Año 2003
Informes del Año 2004
Informes del Año 2005
Informes del Año 2006
Informes del Año 2007
Informes del Año 2008
Informes del Año 2009
Informes del Año 2010
Informes del Año 2011
Informes del Año 2012
Informes del Año 2013
Informes del Año 2014
Informes del Año 2015
Informes del Año 2016
Informes del Año 2017
Informes del Año 2018
Informes del Año 2019
Informes del Año 2020
```

1.1.6 Funciones

Podemos crear funciones, usando la palabra reservada `def` y podemos setear parámetros dentro del paréntesis. Una función puede no tener parámetros.

[12]:
```
def print_mensaje():
    print("Hola a todos!!!")

def print_hello(nombre):
    print(f"hola {nombre}")


print_mensaje()
print_hello("Emmanuel")
```

```
Hola a todos!!!
hola Emmanuel
```

Cuando se envían argumentos a una función, y estos se reciben los parámetros en el orden que fueron defindos, estos son `argumentos por posición`

[13]:
```
def foo(a, b):
    return a * b

res = foo(2, 3)
print(res)
```

```
6
```

También se puede enviar los parámetros con sus nombres, ya no estamos obligados a llamar a la función respetando el orden con el que fueron definidos

[14]:
```
res = foo(b=3, a=2)
print(res)
```

```
6
```

Podemos usar parámetros por defecto, dónde le asignamos el valor del parámetro en la defincón de la función, de esta forma, ese argumento ya va a tener un valor, por lo que podemos ignorarlo durante la llamada

[15]:
```
def bar(a, b=3):
    return a * b

print(bar(2))
print(bar(3, 4))
print(bar(b=3, a=4))
```

```
6
12
12
```

1.1.7 Ejemplo

[16]:
```
def iva():
    iva_percent = 0.21
    precio = int(input("Cual es el precio del producto"))
    print(f"El impuesto es de ${precio * iva_percent}")

iva()
```

```
Cual es el precio del producto131
El impuesto es de $27.509999999999998
```

[17]:
```
def iva(precio, iva_percent = 0.21):
    return precio * iva_percent

impuesto_a_pagar = iva(100)

print(f"El impuesto es de ${impuesto_a_pagar}")
```

```
El impuesto es de $21.0
```

2 POO

Este es un paradigma de programación que cambia la forma de procesar datos. Con la POO ya no tenemos una secuencia de comandos sino que tenemos ahora entidades u objetos, que tienen un comportamiento y características, y permite la interacción entre objetos, por ejmplo mediante el envío de mensajes. Las tres características principales de la POO son: Encapsulación, Polimorfismo y Herencia.

2.1 Clases

Representa una abstracción una generalización. La clase es una encapsulación porque constituye una encapsula los datos que conforman los objetos como los procedimientos que permiten manipularlos.

2.2 Objetos

Cada objeto tiene una identidad, un tipo y un valor. Un objeto representa alguna entidad de la vida real, y que pertenecen al problema con el que nos estamos enfrentando, y con el que podemos interactuar.

[18]:
```
# espacio para mostrar codigo en vivo
```

3 Módulos

En Python, cada uno de nuestros archivos .py se denominan módulos. Estos módulos, a la vez, pueden formar parte de paquetes. Un paquete, es una carpeta que contiene archivos .py. Pero,

para que una carpeta pueda ser considerada un paquete, debe contener un archivo de inicio llamado **init**.py. Este archivo, no necesita contener ninguna instrucción. De hecho, puede estar completamente vacío.

[19]:
```
# espacio para practicar
```

[ ]:
```

```