IPAP | PROGRAMA ORGANISMOS PROVINCIALES

curso virtual

Anexo rápido de Python

Variables

In [133…

```
x = 7  # Las variables se crean dinamicamente...
print(x)

x ='¡Buen dia!'  # ... y pueden cambiar de tipo a lo largo del programa
print(x + '¿Cómo estás?')
```

```
7
¡Buen dia!¿Cómo estás?
```

Reglas para declarar variables

- Python hace diferencia entre mayúsulas y minúsculas.
- Las variables variable_1 y Variable_1 son DOS variables **DISTINTAS**.
- Los nombre de las variables sólo pueden contener letras, dígitos y _.
- Y **siempre** deben comenzar con letra.
- Las variables con nombre compuesto se separan con **"_"**

In [134…

```
descuento_total = 75
proceso_finalizado = False
nombre_de_usuario = "Rogelio"
```

Convenciones de estilo

“El código es leído muchas más veces de lo que es escrito” (Guido Van Roussen)

- Están especificadas en la PEP 8
- Hay guías sobre la indentación, convenciones sobre los nombres, etc.
- Algunos IDEs chequean que se respeten estas guías.
- Su adopción es MUY importante cuando se comparte el código.

Indentacion

En Python la indentacion es **obligatoria**

In [135…

```
x = 5
```

```
if x == 5:
    print("La variable X es igual a 5")
else:
    print("La variable X es distinta de 5")
```

```
La variable X es igual a 5
```

Comentarios

Los comentarios en una linea se realizan con `#`. Los comentarios multilinea con `"""`

In [136…

```
print("Mensaje generico")  # Imprime "mensaje generico"

"""
Comentario multilinea que
no se sejecuta
"""
print(2)
```

```
Mensaje generico
2
```

Tipos de datos

Tipos predefinidos: (Built-In Data Types)

- Números (enteros, flotantes y complejos)
- Booleanos
- Cadenas de texto
- Listas, tuplas, diccionarios y conjuntos

Expresiones numericas

In [137…

```
"""
Operaciones clasicas:

+ SUMA
- RESTA
* MULTIPLICACION
/ DIVISION
// DIVISION ENTERA
% MODULO
** POTENCIA
"""

x = 16
y = 4
z = x / y  # La division entre dos enteros devuelve un float
print(z)
print(y/2.0)  # Una expresion entre un entero y un flotante devuelve un flo
```

```
4.0
2.0
```

Conversiones de tipo

In [138…

```
x = 16
y = 4
z = 16 / 4
print(z)
print(int(z))  # cambia z de tipo float a tipo int
```

```
4.0
4
```

In [139…

```
x = 2  # x de tipo int
print(x)
x = float(2)  # cambia x a tipo float
print(x)
x = str(x)  # cambia x a tipo string
print(x)
```

```
2
2.0
2.0
```

La funcion input

Permite ingresadar datos desde teclado, siempre en tipo string

In [140…

```
dato = input("Ingrese algun dato: ")
type(dato)
```

```
Ingrese algun dato: 5
```

Out[140…

```
str
```

Booleanos

- Solo permiten dos valor True y False
- Operadores logicos: **and**, **or**, **not**

In [141…

```
x = True
y = False
print(x and y, x or y, not x)
```

```
False True False
```

Comparadores

Operadores relacionales: ==, !=, >, <, >=,<=

In [142…

```
print(5 > 2)
```

```
True
```

In [143…

```
print(5 == "5")
```

```
False
```

Bucles

In [144…

```
for i in range(5):  # itera desde 0 hasta 4
    print(i)
```

```
0
1
2
3
4
```

In [145…

```
x = 0
while x < 3:
    print(x)
    x = x + 1
print("Fin")
```

```
0
1
2
Fin
```

Condicionales

In [146…

```
x = 7
y = 2
if x > 2:
    print("X es mayor a Y")
elif x == 7:
    print("X vale 7")
else:
    print("No se cumplio ninguna condicion")
```

```
X es mayor a Y
```

Listas

- Una lista es una coleccion de elementos
- Las listas pueden contener **cualquier** tipo de datos, incluso otras listas

In [147…
```
lista1 = [1, 2, 3, 4, 5]
lista2 = ["texto", 123, False, [1, 2, 3]]
print(lista1)
print(lista2)
```

```
[1, 2, 3, 4, 5]
['texto', 123, False, [1, 2, 3]]
```

In [148…
```
cant_elems = len(lista1)  # Devuelve la cantidad de elementos de lista1
print(cant_elems)
```

```
5
```

In [149…
```
print(lista2[0])  # Accede al primer elemento de lista2
print(lista2[2])  # Accede al tercer elemento de lista2
```

```
texto
False
```

In [150…
```
lista1 = [1, 2, 3, 4, 5]
print(lista1)
lista1[0] = 99  # Los datos de las listas pueden cambiar, es decir son muta
print(lista1)
```

```
[1, 2, 3, 4, 5]
[99, 2, 3, 4, 5]
```

In [151…
```
lista1.append(88)  # Agrega un nuevo elemento al final de lista1
print(lista1)
```

```
[99, 2, 3, 4, 5, 88]
```

Recorrer listas

In [152…
```
lista = [1, 2, 3, 4]
for elem in lista:  # Por cada elemento en la lista, lo guardo en elem y...
    print(elem)  # ... lo imprimo
```

```
1
2
3
4
```

In [153…
```
# Forma mas "tradicional"
lista = [1, 2, 3, 4]
cant_elementos = len(lista)  # Guardo en una variable la cantidad de elemen
for i in range(cant_elementos):  # Lo uso para iterar
    print(lista[i])
```

```
1
2
3
4
```

Funcion split

La funcion split permite generar una lista al separar un `string` por determina `caracter`

In [154…
```
x = "Habia una vez un elefeante rosado"
palabras = x.split(" ")  # Divide x en cada caracter espacio y genera una l
print(palabras)
```

```
['Habia', 'una', 'vez', 'un', 'elefeante', 'rosado']
```

In [155…
```
x = "Habia una vez un elefeante rosado"
palabras = x.split("a")  # Divide x en cada caracter "a" y genera una lista
print(palabras)  # El caracter usado para realizar el split es eliminado
```

```
['H', 'bi', ' un', ' vez un elefe', 'nte ros', 'do']
```

Diccionarios

- Un diccionario es un conjunto no ordenado de pares de datos: clave:valor.
- Se definen con { }
- Las claves deben ser inmutables (ej: integers y strings)

In [156…
```
edades = {"Diego": 31, "Agustina": 5, "Rogelio": 99}
print(edades)
```

```
{'Diego': 31, 'Agustina': 5, 'Rogelio': 99}
```

In [157…
```
print(edades["Rogelio"])  # Accede directamente a la edad de Rogelio
```

```
99
```

In [158…
```
edades["Timmy"] = 7  # Agrega un nuevo elemento al diccionario
print(edades)
```

```
{'Diego': 31, 'Agustina': 5, 'Rogelio': 99, 'Timmy': 7}
```

Recorrer un diccionario

In [159…
```
marcas = {"teconologia": ["Apple", "Google", "Lenovo"],
"gaseosas": ["Coca Cola", "Pepsi", "Fanta"]}

#las claves
for elem in marcas:
    print(elem)

# los valores
for elem in marcas:
    print(marcas[elem])
```

```
teconologia
gaseosas
['Apple', 'Google', 'Lenovo']
['Coca Cola', 'Pepsi', 'Fanta']
```

Funciones

In [160…

```
def suma_elems_de_lista(lista_de_entrada):
    """
    Por convencion todas las funciones deben llevar un docstring explicando
    funcionalidad

    Suma todos los elementos de la lista que se le envia por parametro
    Devuelve el total
    """
    total = 0
    for numero in lista_de_entrada:
        total = total + numero
    return total  # Devuelve el total de la multiplicacion

x = suma_elems_de_lista([1,2,3,4,5,6])  # Invocacion de la funcion
print(x)
```

```
21
```

Importar modulos

In [161…

```
import random  # Importa la libreria random que viene al instala python

print(random.randint(0, 99))  # Imprime un numero al azar entre 0 y 99
```

```
97
```

Excepciones

Si intentamos hacer 5/0 nos da error. Para evitar que el programa se detenga, utilizamos las excepciones.

Muy util para cuando intentamos abrir un archivo que no sabemos si existe o realizar operaciones que puedan fallar, como intentar conectarse a una base de datos externa.

In [162…

```
# Si intentamos hacer 5/0 nos da error
try:
    print(5/0)
except ZeroDivisionError:
    print("No se puede dividir por cero")
```

```
No se puede dividir por cero
```