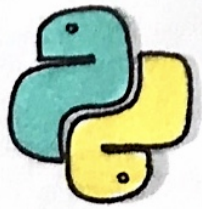
python



Elegante - Fácil - Buenas prácticas

¿Qué es un algoritmo?

→ Es una serie de pasos ordenados para resolver un problema

Comandos de la consola

cd : para moverse de carpeta en carpeta.

ls : muestra el contenido de una carpeta.

mkdir : crea carpetas.

touch : para crear archivos.

operadores aritmeticos

- suma (+)
- resta (-)
- multiplicación (*)
- división (/)
- división euclidiana (//)
- módulo (%)
- potencia (**)

variables

Una variable es un sector donde guardamos objetos, como texto, números, etc.

Las variables tienen nombre: un identificador.

identificador = objeto

OPERADOR DE ASIGNACIÓN

nombre = 'Facundo'
numero = 16

Reglas para definir los nombres de las variables:

- No puede comenzar con números
- Debe estar en minúsculas
- Las palabras dentro del mismo nombre se separan con guión bajo (_)

tipos de datos

- Números : número entero <class 'int'>
  número de punto flotante <class 'float'>

- Texto : cadena de caracteres. <class 'str'>

- Booleanos : True o False <class 'bool'>

- Listas : conjunto de datos. <class 'list'>

  numeros = [1, 3, 6]

  variable = [1, 'Facundo', 16]

Se accede a las listas mediante un índice. Los índices comienzan en cero.

numeros[0] ⟶ mostrará 1.

- Tuplas : conjunto de datos que no se puede modificar.

  <class 'tuple'>

  a = (1, 'Python', 16)

- Diccionarios: estructura de datos de llaves y valores

```
<class 'dict'>
mi_diccionario = {
    'llave1' : 1,
    'llave2' : 2,
    'llave3' : 3,
}
```

Podemos recorrer un diccionario con el ciclo FOR: el método keys devuelve las llaves; el método values devuelve los valores. El método items devuelve tanto llaves como valores.

Métodos de listas:

append	permite agregar un elemento	objetos = [1, True, 4.5] objetos.append (3) objetos=[1, True, 4.5, 3]
pop	permite borrar un elemento	objetos = [1, True, 4.5] objetos.pop(1) objetos=[1, 4.5]

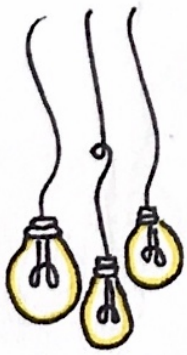
operadores lógicos

es_estudiante = True
trabaja = False
} dos variables

- es_estudiante and trabaja
  Devolverá 'True' cuando todas las variables a comparar sean 'True'
  → False: ambos deben ser verdaderos
- es_estudiante or trabaja
  Devolverá 'False' cuando todas las variables a comparar sean False.
  → True: al menos uno es verdadero

- not es_estudiante
  → False: invierte el valor de la variable

Operadores de comparación



numero1 = 2 } variables a
numero2 = 5 } comparar

¿Cómo puedo saber si el contenido de las dos variables es el mismo ?

numero1 == numero2 → False

OPERADOR DE IGUALDAD

numero1 != numero2 → True

OPERADOR DE DESIGUALDAD

> = MAYOR o igual

< = MENOR o igual

> MAYOR

< MENOR

condicionales

En Python puedes definir una serie de condicionales utilizando:

- `if` : Primera condición
- `elif`: Resto de las condiciones.
  Puede haber múltiples elif.
- `else`: se ejecuta cuando las anteriores condicionen son falsas.

```
if numero >5:
    print ('Es mayor a 5')
elif numero == 5:
    print ('Es igual a 5')
else
    print ('Es menor a 5')
```

funciones

una función es una pieza de código que se puede invocar varias veces.

Se definen con la palabra clave def.

Las funciones pueden recibir opcionalmente parámetros.

NOMBRE DE LA FUNCIÓN

```
def imprimir_mensaje():
    print('Mensaje')
```

HAY QUE TENER EN CUENTA LA INDENTACIÓN ←

```
imprimir_mensaje()
```
→ INVOCAMOS LA FUNCIÓN

bucles

nos permite repetir la ejecución de un bloque de código una cantidad determinada de veces.

ciclo while

Permite ejecutar un bloque de código mientras una condición se cumpla.

```
while condición:
    código
```

break termina la ejecución de un ciclo.

ciclo for

Permite ejecutar un bloque de código mientras se recorre un rango determinado, teniendo disponible en cada vuelta del cicdo a un elemento de ese rango.

```
for elemento in range
    código
```

continue 'salta' la iteración actual sin romper el bucle.