Repaso de la clase 1

Taller de Python FIFA

Python como calculadora

```
# Se suma con "+"
1 + 2
>> 3
```

```
# Se resta con "-"
1 - 2
>> -1
```

```
# Se divide con "/"
2/1
>> 2.0
```

```
# Se multiplica con "*"
2*4
>> 8
```

```
# Se eleva a una potencia con "**"
9**(1/2)
>> 3.0
```

OBSERVACIÓN:

Se usan los paréntesis igual que en la calculadora para *especificar el orden de las operaciones.*

Tipos de objeto

1. Los números enteros (**INT:** `1, 2, 3`)
   y los números "reales" (**FLOAT:** `1.0, 2.17`) son
   dos *tipos de objeto* en Python.

2. Otros tipos de objeto:

   STR: strings, textos, `'Hola FIFA'`

   BOOL: booleanos, `True` o `False`

   LIST: listas, `['Napo', 'Muzza', 2001]`

¿Por qué sirve conocer de qué tipo es un objeto?

Tipos de objeto

Ejemplos:

1. Suma y resta de enteros:

```
1 + 1 - 2
>> -1
```

2. Suma de listas:

```
['Papas'] + ['Ketchup', 'Mostaza']
>> ['Papas', 'Ketchup', 'Mostaza']
```

Para distintos objetos, distintas operaciones.

3. Resta de listas:

```
['Papas'] - ['Ketchup', 'Mostaza']
>> TypeError: unsupported operand type(s)
for -: 'list' and 'list'
```

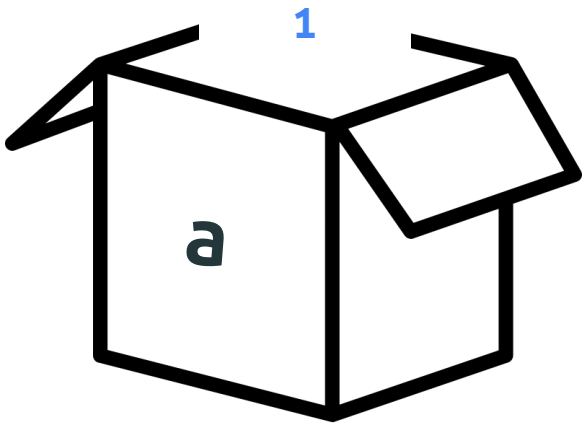
Variables

Si vamos a usar un **objeto** varias veces en un programa, conviene **guardarlo bajo un nombre**.

```
a = 1
b = 2 + a


print(a)
print(b)

>> 1
>> 3
```



El “=” asigna un objeto a una variable!

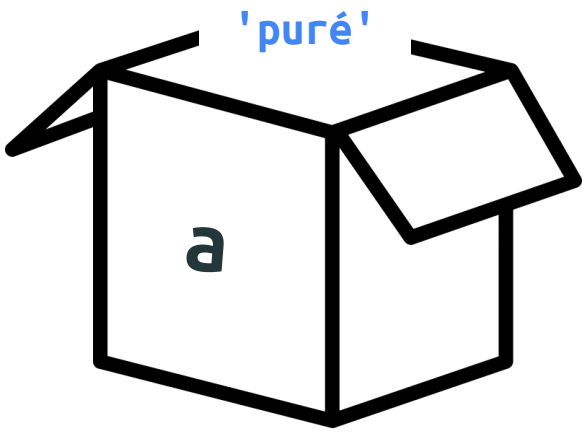
Variables

Si vamos a usar un **objeto** varias veces en un programa, conviene **guardarlo bajo un nombre**.

```
a = 1
b = 2 + a
a = 'puré'

print(a)
print(b)

>> puré
>> 3
```



Pueden usar pythontutor.com para visualizar el orden de ejecución y ver cómo cambian las variables!

Tipo *bool*

Le podemos preguntar a Python si las cosas son verdaderas (`True`) o falsas (`False`):

¿`a` es un entero?

```
a = 1
b = a
type(a) == int
>> True
```

Tipo *bool*

Le podemos preguntar a Python si las cosas son verdaderas (`True`) o falsas (`False`):

¿**a** es un entero *y* es mayor a **b**?

```
a = 1
b = 2
(type(a) == int) and (a > b)
>> False
```

`if`: *Condicionales*

uso del tipo ***bool*** →

```
# Programa que verifica que la temperatura de una muestra está
dentro de los rangos necesarios para empezar a medir.

temp_ideal = 20  # [Celsius]
tol = 3  # [Celsius], tolerancia de la muestra

if temp <= temp_ideal - tol:
     print('La muestra está debajo de la tolerancia.')
     print('Pausar el experimento y calentar la muestra.')

print(f'La temperatura de la muestra es {temp}')
```

`if`: *Condicionales*

Indentado: solo corre si es verdadera la condición

```
# Programa que verifica que la temperatura de una muestra está
dentro de los rangos necesarios para empezar a medir.

temp_ideal = 20  # [Celsius]
tol = 3  # [Celsius], tolerancia de la muestra

if temp <= temp_ideal - tol:
    print('La muestra está debajo de la tolerancia.')
    print('Pausar el experimento y calentar la muestra.')

print(f'La temperatura de la muestra es {temp}')
```

dos puntos!!

corre siempre

`if`: *Condicionales*

Solo corre si **es falsa** la condición anterior y **es verdadera** la condición del *elif*.

```
# Programa que verifica que la temperatura de una muestra está
dentro de los rangos necesarios para empezar a medir.

temp_ideal = 20  # [Celsius]
tol = 3  # [Celsius], tolerancia de la muestra

if temp <= temp_ideal - tol:
    print('La muestra está debajo de la tolerancia.')
    print('Pausar el experimento y calentar la muestra.')

elif temp >= temp_ideal + tol:
    print('La muestra está encima de la tolerancia.')
    print('Pausar el experimento y enfriar la muestra.')

print(f'La temperatura de la muestra es {temp}')
```

`if`: *Condicionales*

Solo corre si **no se cumplió ninguna condición antes**.
No prueba otra condición.

```
# Programa que verifica que la temperatura de una muestra está
dentro de los rangos necesarios para empezar a medir.

temp_ideal = 20  # [Celsius]
tol = 3  # [Celsius], tolerancia de la muestra

if temp <= temp_ideal - tol:
    print('La muestra está debajo de la tolerancia.')
    print('Pausar el experimento y calentar la muestra.')

elif temp >= temp_ideal + tol:
    print('La muestra está encima de la tolerancia.')
    print('Pausar el experimento y enfriar la muestra.')

else:
    print('Seguir con el experimento.')

print(f'La temperatura de la muestra es {temp}')
```

Listas

Las listas sirven para almacenar varios elementos en una variable:

```
mis_objetos_favoritos = [42, True, 'Milanesa']
```

Dado que las listas están ordenadas, se accede a los elementos usando su índice*:

```
animales = ['nemo', 'koala', 'kiwi']
print(animales[1])
```

```
>> koala
```

***Comenzando a contar desde el 0 !!**

Listas

Las listas sirven para almacenar varios elementos en una variable:

```
mis_objetos_favoritos = [42, True, 'Milanesa']
```

Dado que las listas están ordenadas, se accede a los elementos usando su índice*:

```
animales = ['nemo', 'koala', 'kiwi']
print(animales[1])
print(animales[-1])

>> koala
>> kiwi
```

for: *Bucles*

Indentado:
El bloque de código que se repite.

INPUT:

```
animales = ['nemo', 'koala', 'kiwi']

for animal in animales:
    print(f'oh! es un {animal}?')
    print('No, es batman')

print(f'La longitud de la lista "animales" es {len(animales)}')
```

dos puntos!!

OUTPUT:

```
>> oh! es un nemo?
>> No, es batman
>> oh! es un koala?
>> No, es batman
>> oh! es un kiwi?
>> No, es batman
>> La longitud de la lista "animales" es 3
```

`for`: *Bucles*

`animal:`

Variable **iteradora**

Fuera del `for` *no está definida.*

Cambia en cada iteración según el orden de la lista.

INPUT:

```
animales = ['nemo', 'koala', 'kiwi']

for animal in animales:
      print(f'oh! es un {animal}?')
      print('No, es batman')

print(f'La longitud de la lista "animales" es {len(animales)}')
```

OUTPUT:

```
>> oh! es un nemo?
>> No, es batman
>> oh! es un koala?
>> No, es batman
>> oh! es un kiwi?
>> No, es batman
>> La longitud de la lista "animales" es 3
```

for: *Bucles*

INPUT:

```
animales = ['nemo', 'koala', 'kiwi']

for animal in animales:
     print(f'oh! es un {animal}?')
     print('No, es batman')

print(f'La longitud de la lista "animales" es {len(animales)}')
```

Salió del loop

OUTPUT:

```
>> oh! es un nemo?
>> No, es batman
>> oh! es un koala?
>> No, es batman
>> oh! es un kiwi?
>> No, es batman
>> La longitud de la lista "animales" es 3
```

Corre una vez

range(start, stop, step)

Es una función que permite iterar sobre secuencias de números equiespaciados de forma rápida.

```
for i in range(1, 10, 2):
    print(i)
```

```
>> 1
>> 3
>> 5
>> 7
>> 9
```

No incluye el último número.

No devuelve una lista!

```
print(range(5))
print(type(range(5))
```

```
>> range(0, 5)
>> <class 'range'>
```

...pero se puede convertir en una lista:

```
print(list(range(5)))
```

```
>> [0, 1, 2, 3, 4]
```