Python

2. Contenidos

BLOQUE 1

TEMA 1: Conceptos básicos de programación en Python

- Fundamentos de la programación de alto nivel
- Diferencias entre compilados e interpretados
- Errores frecuentes y depuración
- Generalidades sobre editores, IDEs
- Edición de texto plano. Nano

TEMA 2: Sintaxis, operadores, tipos de datos

- Operadores matemáticos y lógicos
- Tipos de datos simples
- Tipos de datos complejos

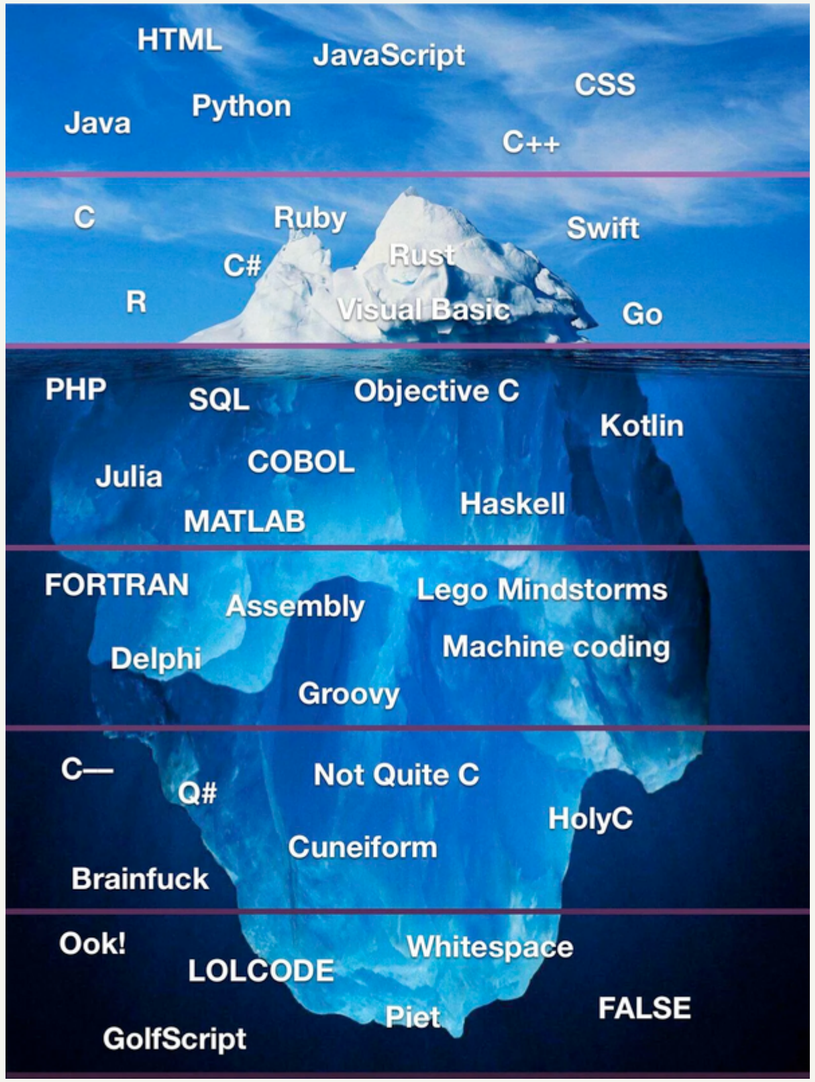
TEMA 3: Estructuras de control de flujo

- Indentación, ejecución condicional y control de variables
- Iteraciones: Tipos, bloques, recursividad
- Instrucciones break-continue-pass: Usos y depuración de errores
- Instrucciones try-except y raise: Usos y depuración de errores

Características de Python

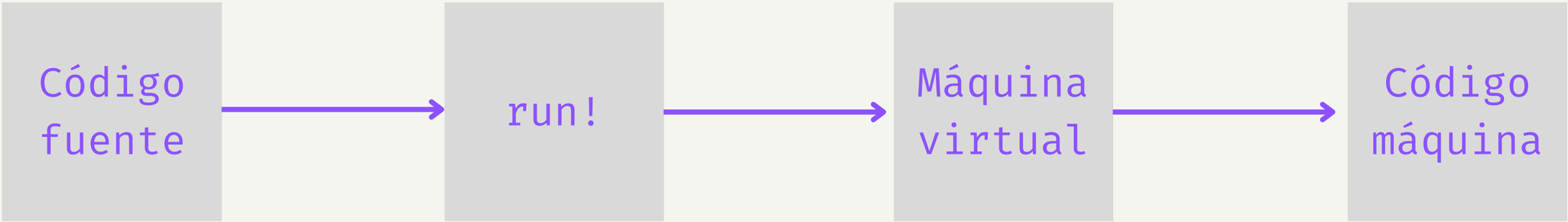
- Alto nivel
- Interpretado
- Tipado dinámico
- Multiparadigma
- Orientado a Objetos

Lenguaje humano



10010011110011

Interpretado



Compilado

Tipos de datos y operadores

- Conocer los tipos de datos primitivos
- Conocer los diferentes tipos de operadores
- Diferenciar integers de floats

Tipos de datos

int	3
float	2.5
complex	3j
str	"I'm a string!"
bool	True/False

Operadores

Arithmetic	+,-,*,/,**,%
Comparison	<,>,>=,==,!=
Assignment	=, *=, /=
Logical	and, or, not

Números

Tipos de números

INT	3
FLOAT	3.2
COMPLEX	3J
Operadores	+,-,*,/,**,%

String

- Iterables
- Index-based
- Objetos

name

```
type(<object>)
```

argument

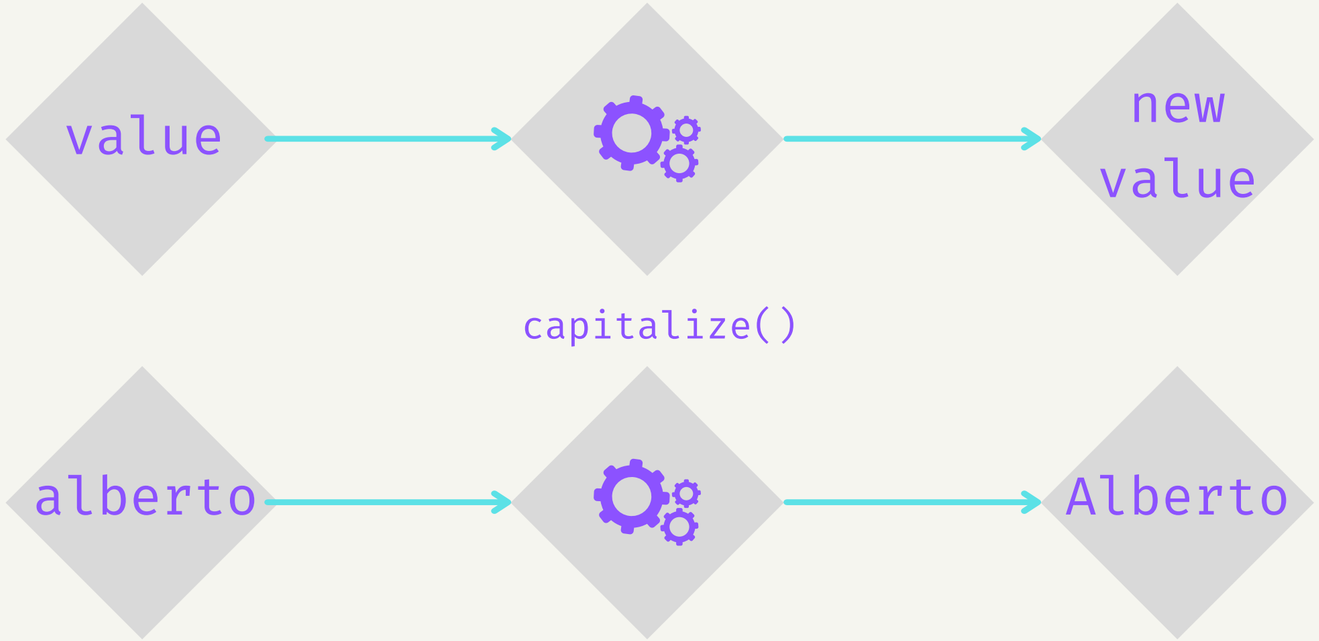
name

```
sorted(<iterable>, [reverse=True])
```

argument

optional

Los métodos son una porción de código previamente escrita que nos permite manipular las propiedades de un determinado Objeto

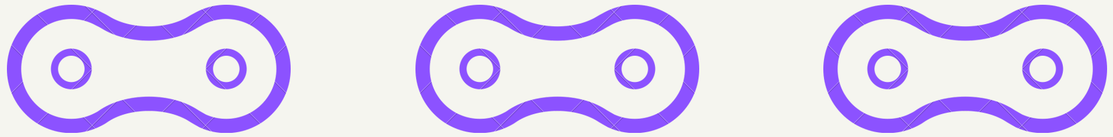


https://www.w3schools.com/python/python_ref_string.asp

Iterables

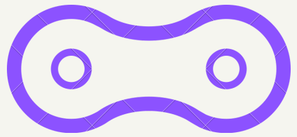
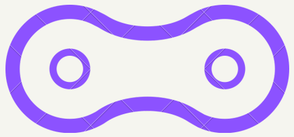
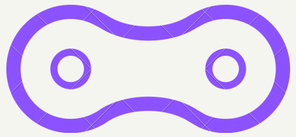
F O O

"F" "O" "O"



Iterables

F O O

"F"	"O"	"O"
		
0	1	2

if

o.
logical

and, or, not

o.
compar

<,>,>=,==,!=

```
if <condition>:
    this happens



if 1 == 1:
    print("we know!")
```

```
if <condition_1>:
    result 1
elif <condition_2>:
    result 2

if 1 != 1:
    print("we don't know!")
elif 1 == 1:
    print("This is the correct!")
```

```
if <condition_1>:
    result 1
else:
    result 2

if 1 != 1:
    print("we don't know!")
else:
    print("This will always happen")
```

List, Sets & Tuples

- Colección de datos ordenada
- Index-based
- Objetos

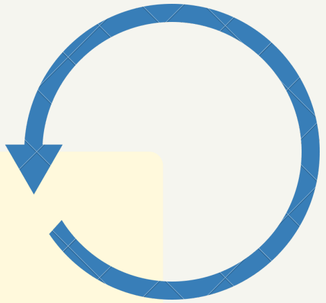
By Position

```
test = ["a", "b", "c", "d"]
test[0] # --> "a"
test[-1] # --> "d"
test[-1] == test[3] # --> True
test[1:3] # --> ["b", "c"]
```

Useful functions

```
test = [1,2,3,4]
sum() # --> 10
max() # --> 4
min() # --> 1
range(1:5) == test # --> True
```

while & for loops



```
while <some_condition>:
    result

while 1 == 1:
    print("A terrible mistake!")
```

```
for <element in iterable>:
    result

for letter in "I'm iterable":
    print(f"that {letter}")
```

```
for num in [1,2,3,4]:
    if num % 2 == 0:
        pass
        print(num)
        continue
    else:
        break
print("Out of the loop")
```

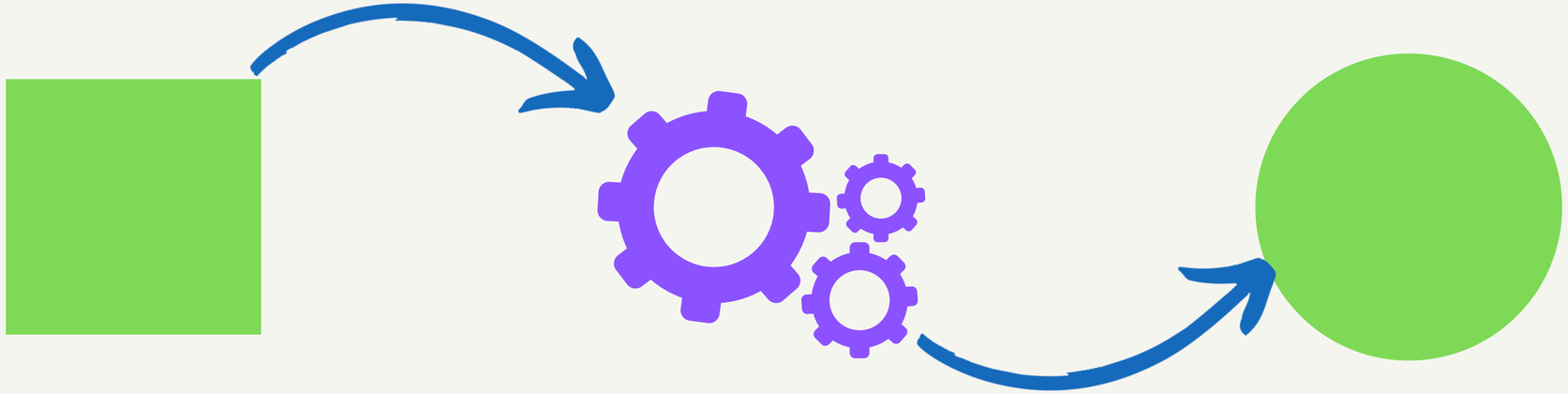
Lists Comprehension

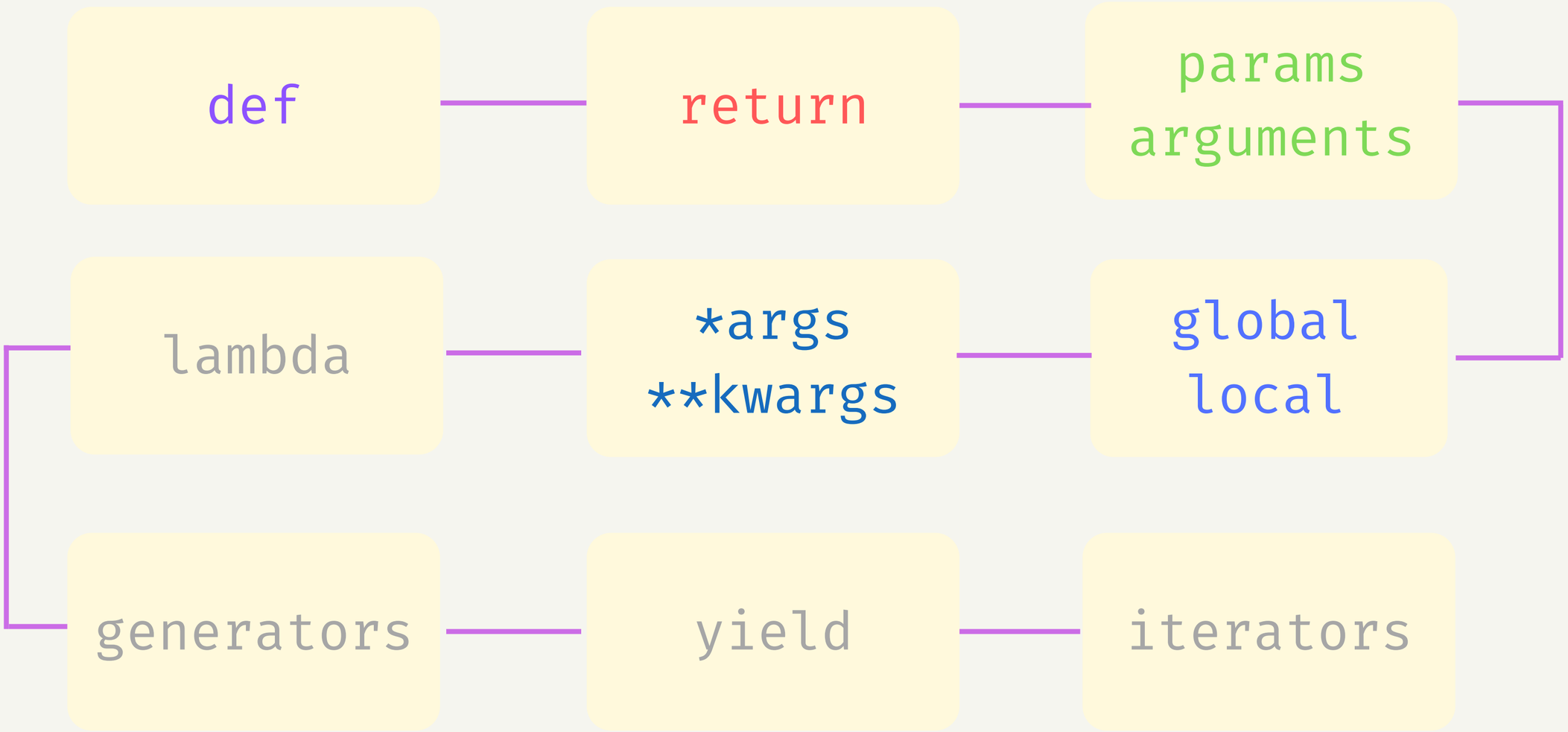
```
test_1 = ["a", "b", "c", "d"]
test_2 = [ltr.upper() for ltr in test_1]
test_2 # --> ["A", "B", "C", "D"]
```

"spread" operator

```
test_1 = ["a", "b", "c", "d"]
test_2 = (*test_1)
test_2 # --> ("a", "b", "c", "d")
```

Functions




def
return
params
arguments
lambda
*args
**kwargs
global
local
generators
yield
iterators

Functions

```
def <name>():
    # action
```

```
def print_something():
    print("I'm a function")
```

Functions

Params

```
def <name>():
    return param
```

Devolver

```
def double(num):
    return num * 2
```