

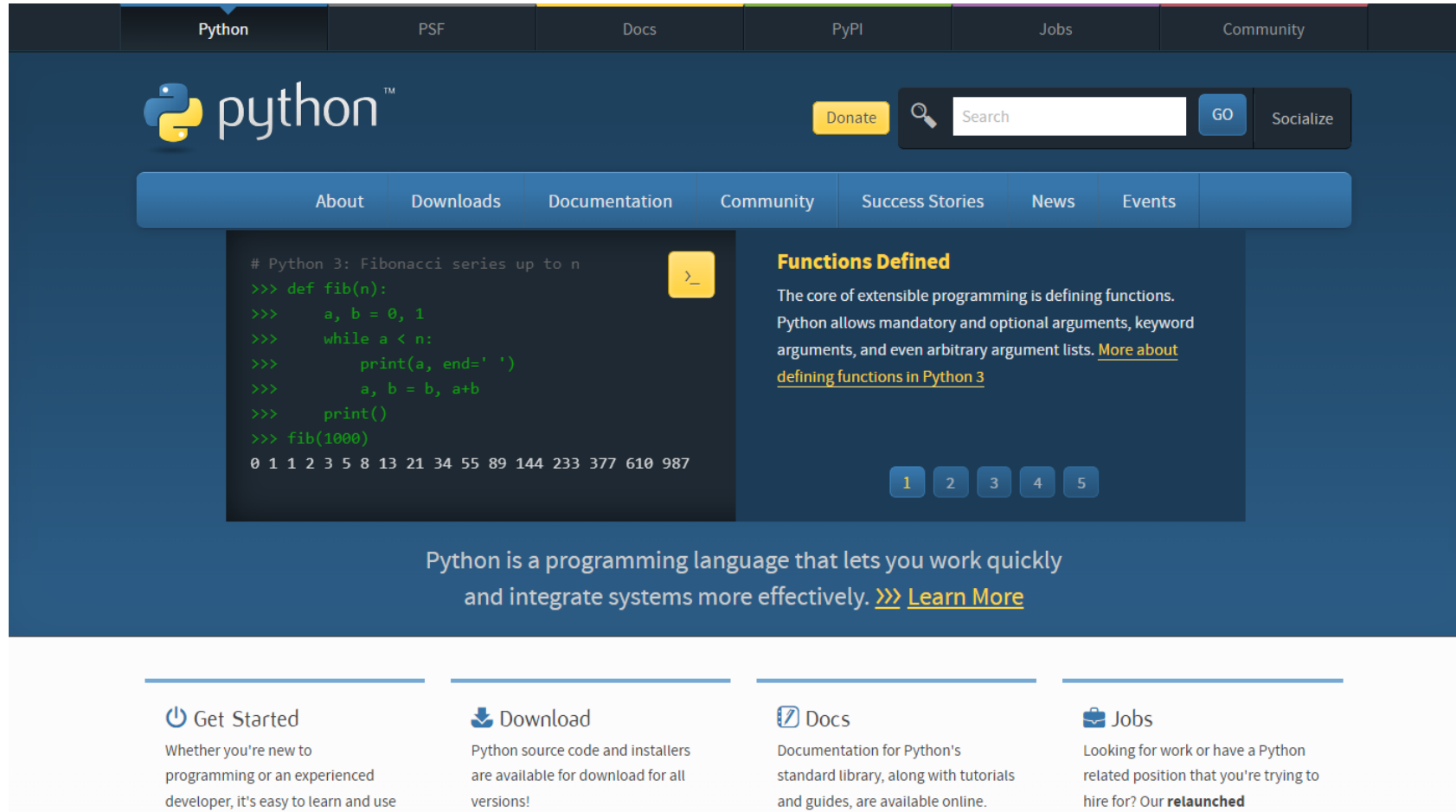
UT 1bis – Lenguaje de Programación Python

Programación de Servicios y Procesos
Curso 2024-25

Profesor: Agustín González-Quel

Python

Python es un lenguaje de programación creado por Guido van Rossum y publicado en 1991.



The screenshot shows the Python.org homepage with a dark blue header and navigation bar. The main content area features a code editor with a Fibonacci function, a 'Functions Defined' section, and a 'Get Started' button. The footer contains four columns: 'Get Started', 'Download', 'Docs', and 'Jobs'.

Python PSF Docs PyPI Jobs Community

python™ [Donate](#) [GO](#) [Socialize](#)

[About](#) [Downloads](#) [Documentation](#) [Community](#) [Success Stories](#) [News](#) [Events](#)

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987
```

Functions Defined

The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)

[1](#) [2](#) [3](#) [4](#) [5](#)

Python is a programming language that lets you work quickly and integrate systems more effectively. [>>> Learn More](#)

Get Started
Whether you're new to programming or an experienced developer, it's easy to learn and use

Download
Python source code and installers are available for download for all versions!

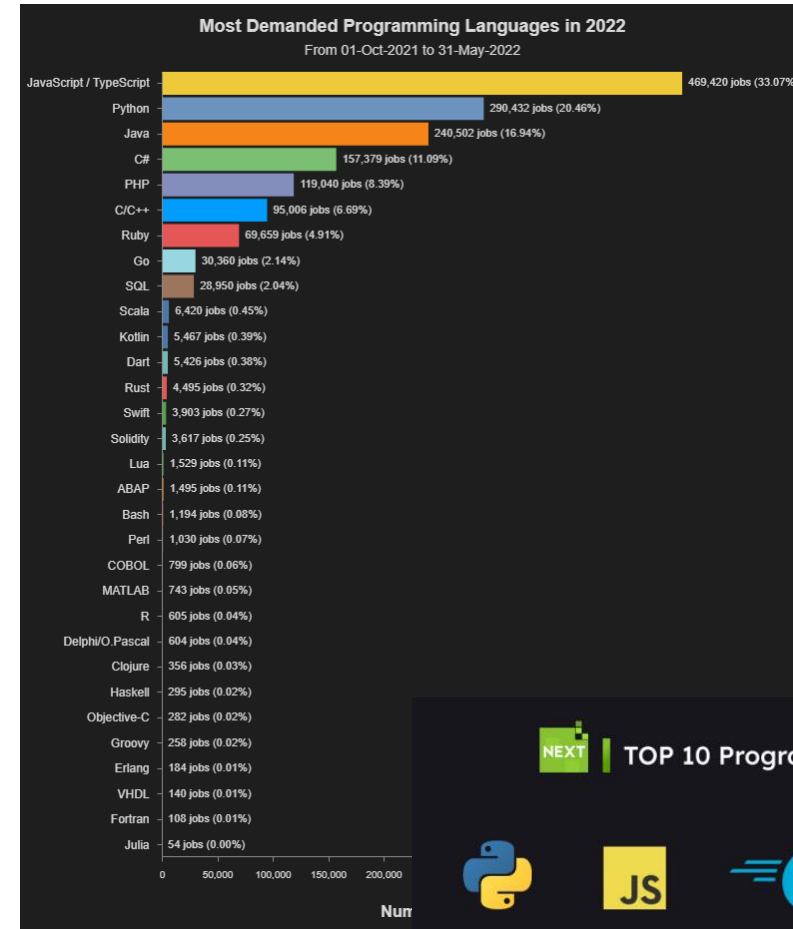
Docs
Documentation for Python's standard library, along with tutorials and guides, are available online.

Jobs
Looking for work or have a Python related position that you're trying to hire for? Our **relaunched**

Características

- Muy usado y demandado en el mercado.
- Lenguaje interpretado.
- Orientado a Objetos.
- Multiplataforma.
- Tipado dinámico.
- Especialmente para servidor, pero también permite programación cliente web.
- Open Source con una gran comunidad.
- No llaves, bloques delimitados por tabulador.

```
IDLE Shell 3.9.10
File Edit Shell Debug Options Window Help
Python 3.9.10 (tags/v3.9.10:f2f3f53, Jan 17 2022, 15:14:21) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = 1
>>> type(a)
<class 'int'>
>>> a = 1.0
>>> type(a)
<class 'float'>
>>> a = '1'
>>> type(a)
<class 'str'>
>>> |
```



Más características

- Se utiliza para:
 - Desarrollo software en general
 - Desarrollo web: sobre todo en el servidor. En la parte cliente con django (<https://www.djangoproject.com/>)
 - Sistemas cercanos a las matemáticas, Inteligencia Artificial, procesamiento de imágenes, ...
- ¿Por qué Python?
 - Python funciona en diferentes plataformas (Windows, Mac, Linux, Raspberry Pi, etc.).
 - Python tiene una sintaxis sencilla similar a la de la lengua inglesa.
 - Python tiene una sintaxis que permite a los desarrolladores escribir programas con menos líneas que otros lenguajes de programación.
 - Python se ejecuta en un sistema de intérprete, lo que significa que el código puede ejecutarse tan pronto como se escribe. Esto significa que la creación de prototipos puede ser muy rápida.
 - Python puede tratarse de forma procedimental, orientada a objetos o funcional.
- Sintaxis de Python comparada con otros lenguajes de programación
 - Python fue diseñado para facilitar la lectura, y tiene algunas similitudes con el idioma inglés con influencia de las matemáticas.
 - Python utiliza nuevas líneas para completar un comando, a diferencia de otros lenguajes de programación que suelen utilizar punto y coma o paréntesis.
 - Python se basa en la sangría, utilizando espacios en blanco, para definir el ámbito de aplicación, como el de los bucles, las funciones y las clases. Otros lenguajes de programación suelen utilizar corchetes para este fin.

Revisar equipo personal: Instalación

- Intérprete. Instalaremos 3.11 ó 3.12
Descarga e instalación
- Gestor de paquetes: PIP o Conda.
 - Yo prefiero PIP
- Entornos de Desarrollo:
 - Visual Studio Code - <https://code.visualstudio.com/>
 - Spyder - <https://www.spyder-ide.org/>
 - PyCharm - <https://www.jetbrains.com/es-es/pycharm/>

Variables y tipos

- Comentarios
 - # Comienza la línea por
 - """ Comentario """
- Variables
 - Sirven para almacenar datos y tienen distintos tipos
 - Entero (int): 23
 - Real (float): 12.34
 - Lógico o booleano (bool): True | False
 - Cadena de caracteres (str): "Hola Paco"
- No se declaran, pero tienen que existir para usarlas:

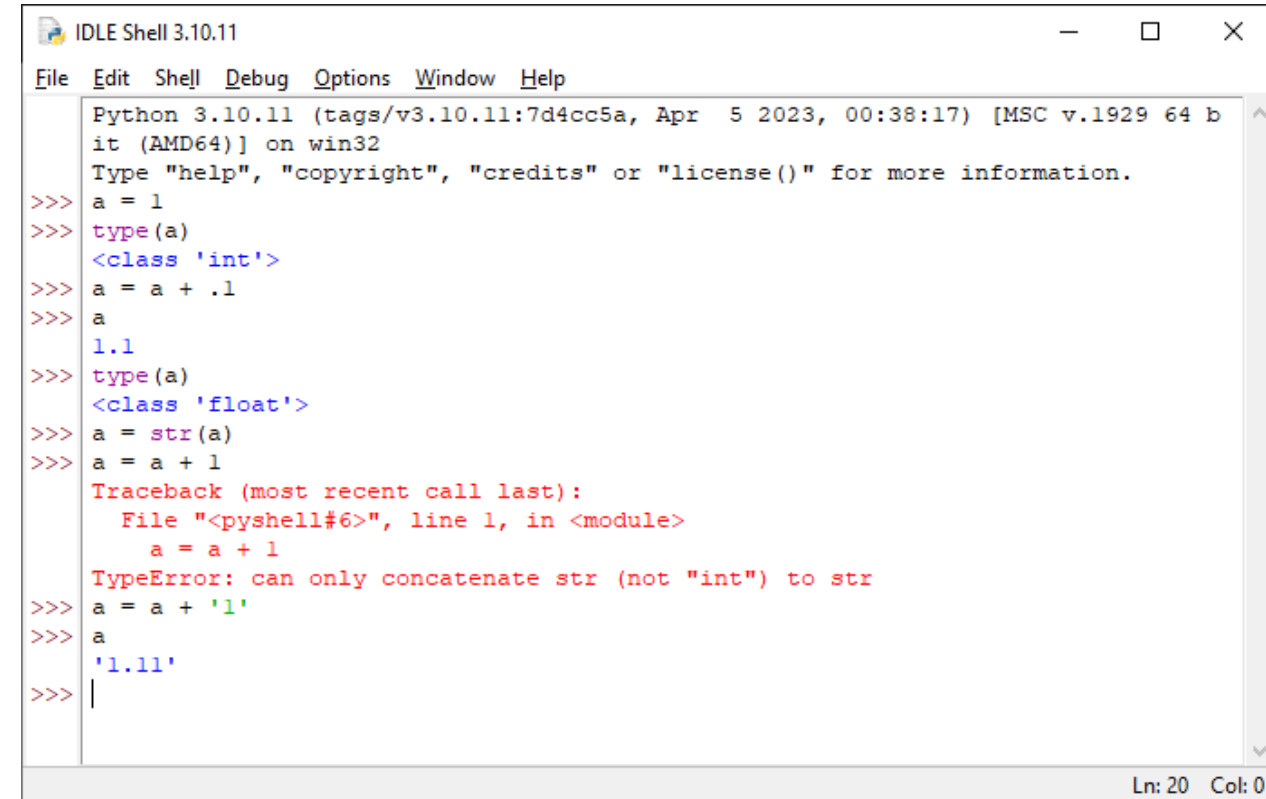
`b = a + 1` → **ERROR**

`a = 0`

`b = a + 1` → **OK**

- Pueden convertirse de un tipo a otro

Int
float
str
bool



```
IDLE Shell 3.10.11
File Edit Shell Debug Options Window Help
Python 3.10.11 (tags/v3.10.11:7d4cc5a, Apr 5 2023, 00:38:17) [MSC v.1929 64 b
it (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> a = 1
>>> type(a)
<class 'int'>
>>> a = a + .1
>>> a
1.1
>>> type(a)
<class 'float'>
>>> a = str(a)
>>> a = a + 1
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    a = a + 1
TypeError: can only concatenate str (not "int") to str
>>> a = a + '1'
>>> a
'1.11'
>>> |
```

Ln: 20 Col: 0

Variables y tipos

Aritméticas

+ Suma: $b+4$

- Resta: $a-b$

- Cambio de signo: $-a$

* Multiplicación

** Exponente

/ División

// División entera

% Resto de la división entera $17\%3 = 2$

Lógicas

and

or

not

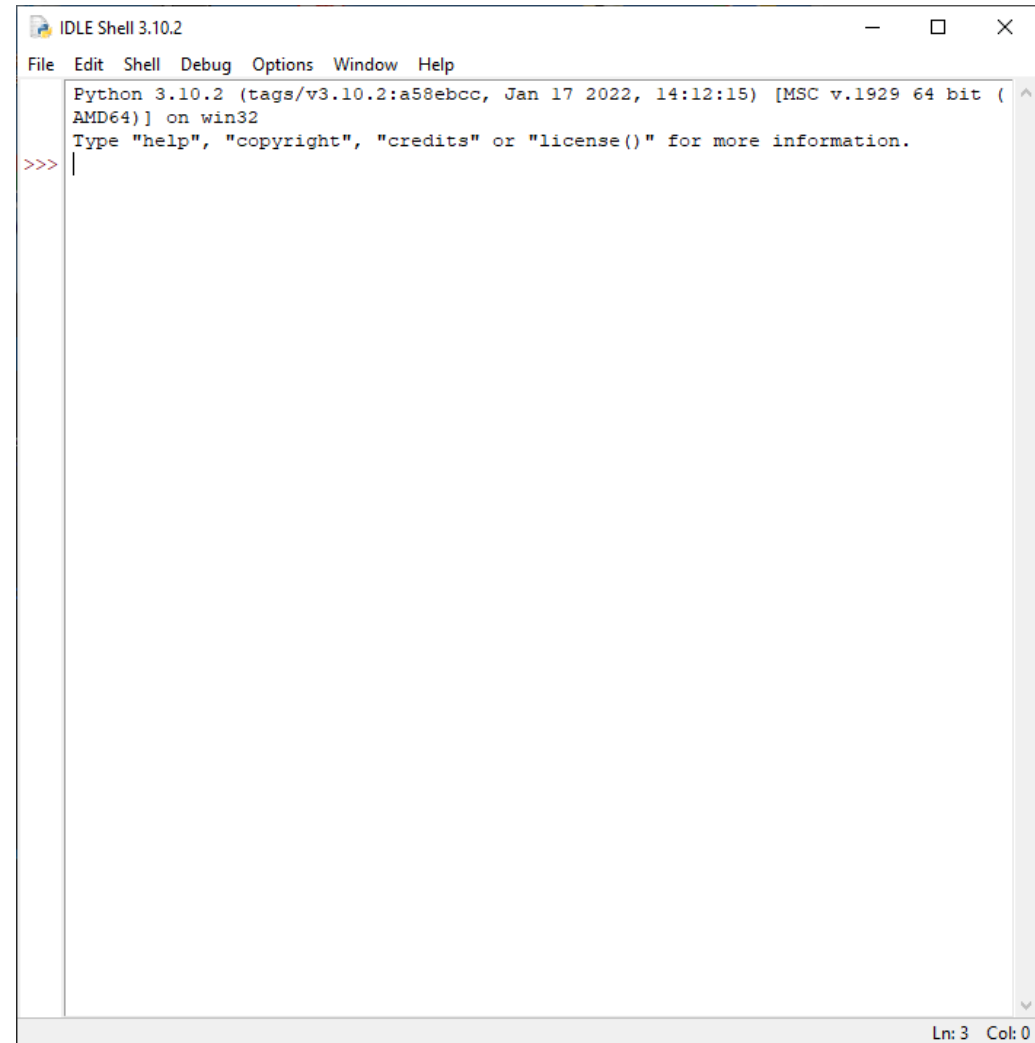
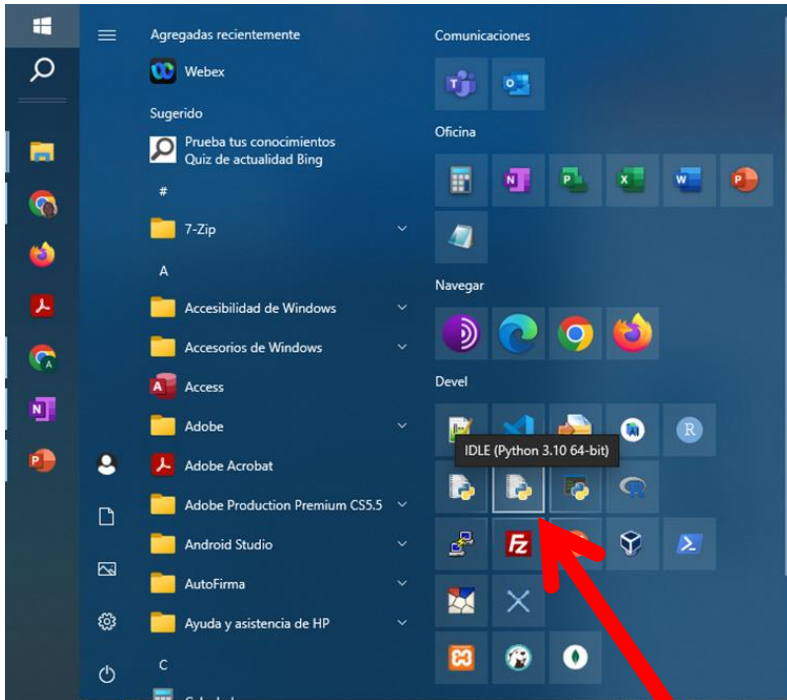
==

!=

>, >=

<, <=

Arrancamos el IDLE para probar



Entrada/Salida

- Funciones

- `input()`
- `print()`

- Estructura

- `Variable = input("mensaje")`
- Variable es tipo string.

- Ejemplos:

```
name = input('Introduce tu nombre: ')\nprint('Hola ', name)\nprint('Hola {}'.format(name))
```

```
num = input('Introduce un número: ')
```

...

```
print('Num cuadrado:{}'.format(num*num))
```

`format()`

- Método de la clase string
- Recibe como parámetro placeholders, nombres de variable y cadenas de definición de formato.

Ejemplo:

```
txt1 = "My name is {fname}, I'm {age}".format(fname =\n"John", age = 36)
```

```
txt2 = "My name is {0}, I'm {1}".format("John",36)
```

```
txt3 = "My name is {}, I'm {}".format("John",36)
```

```
pi = 3.141592
```

```
print("Pi: {:.3f}".format(pi))
```

<https://www.geeksforgeeks.org/python-string-format-method/>

Control de flujo

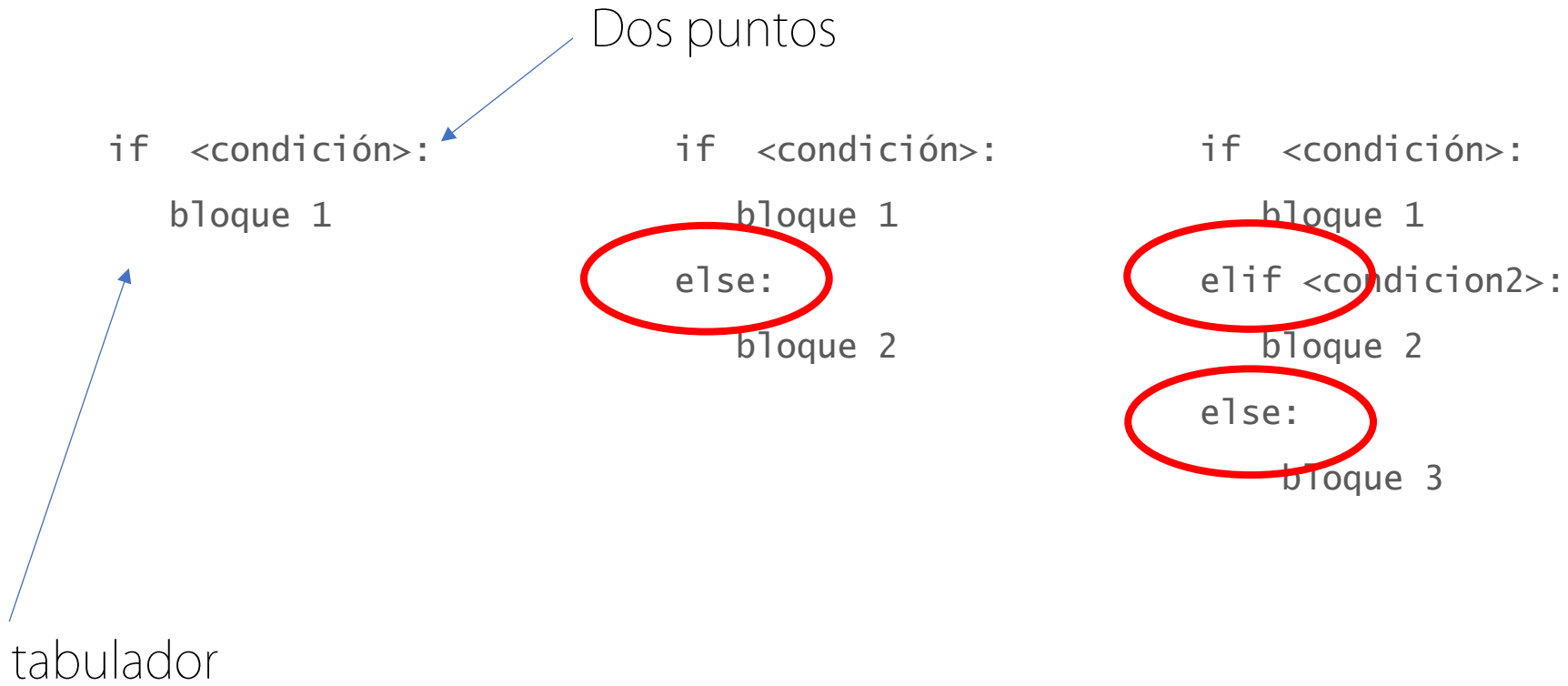
Control de flujo

- if
- for
- while

Aspectos comunes

- El bloque interior está delimitado por tabuladores
- El inicio del bloque lo marcan 2 puntos (:)
- No hay do-while, solo while
- Nuevo concepto: iterable.

If – cómo se escribe



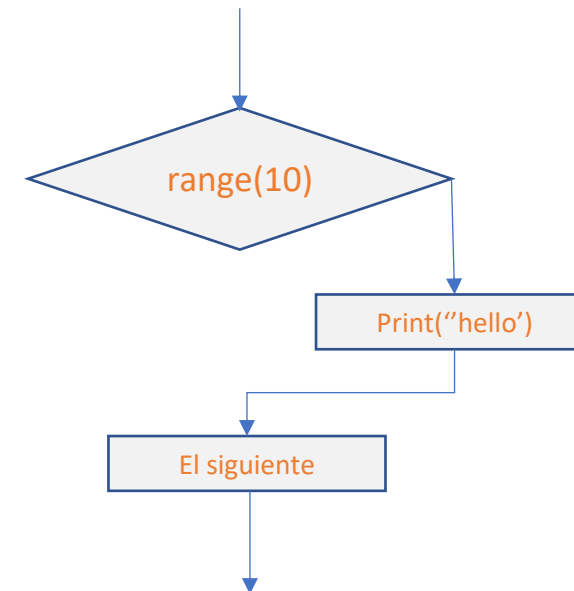
Bucle for / range()

```
for i in range(10):
```

```
    print('Hello')  
    print(i)
```

```
print("fuera del for")
```

“range” es una función que devuelve una lista de valores que definen cuantas veces se ejecuta el bloque de código



Bucle for / función range()

`range(start, stop, step=1)`

- Start: valor inicial
- Stop: valor en el que se detiene (no llega hasta el valor)
- Step: salto

Statement	Values generated
<code>range (10)</code>	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
<code>range (1, 10)</code>	1, 2, 3, 4, 5, 6, 7, 8, 9
<code>range (3, 7)</code>	3, 4, 5, 6
<code>range (2, 15, 3)</code>	2, 5, 8, 11, 14
<code>range (9, 2, -1)</code>	9, 8, 7, 6, 5, 4, 3

While

- Repite un bloque de sentencias hasta que no se cumple la condición
while <condición>:
 <Bloque>

```
temp = 0
while temp!=-1000:
    temp = eval(input('Enter a temperature (-1000 to quit): '))
    if temp != -1000:
        print('In Fahrenheit that is', 9/5*temp+32)
# fuera del while
```

Listas

- Conjunto de valores no necesariamente del mismo tipo con un mismo nombre de variable

`l1 = [1,4,5,7,8]`

`l2 = ['ana', 'perez', 21, 9.8]`

- Funciones

- `len(l)`
- `sum(l)`
- `max(l)`
- `min(l)`

- Métodos: `l.sort()`

Method	Description
<code>append (x)</code>	adds <code>x</code> to the end of the list
<code>sort ()</code>	sorts the list
<code>count (x)</code>	returns the number of times <code>x</code> occurs in the list
<code>index (x)</code>	returns the location of the first occurrence of <code>x</code>
<code>reverse ()</code>	reverses the list
<code>remove (x)</code>	removes first occurrence of <code>x</code> from the list
<code>pop (p)</code>	removes the item at index <code>p</code> and returns its value
<code>insert (p,x)</code>	inserts <code>x</code> at index <code>p</code> of the list

Bucles for y listas

```
lst = ["pan", "café", "yogur", "fruta", "avena"]  
for i in range(len(lst)):  
    print(lst[i])
```

```
lst = ["pan", "café", "yogur", "fruta", "avena"]  
for d in lst:  
    print(d)
```

- No necesitamos recorrer la lista con un índice
- Es más, hay funciones para tener el índice y el elemento a la vez.

```
for i,elem in enumerate(lst):  
    print(i, elem)
```


Más de listas

- Para copiar listas

`m = l` **REFERENCIA**

`m = l [:]` **COPIA DE OBJETOS**

- Si queremos empezar a trabajar con una lista, hay que crearla antes, aunque sea vacía

`l = []` #Es el equivalente a `x = 0`

- Operadores parecidos a string

- `+` - concatena
- `*` - repite la lista
- operador `in`: `x in list` # devuelve True/False

```
l = [4,5,6]
```

```
l[1] = 2                    # [4,2,6]
```

```
l.insert(2,-3)            # [4,2,-3,6]
```

```
del l[1]                   # [4,-3,6]
```

```
l = l*3                    # [4,-3,6, 4,-3,6 4,-3,6]
```

```
l = l[2:4]                 # [6,4]
```

```
l.append(5)                # [6,4,5]
```

Cadenas de caracteres (string)

Tipo de datos que sirve para almacenar texto

```
v = "Pepe"
```

```
nombre = input("Introduce tu nombre:  
)
```

len(cad) Devuelve la longitud

```
v = 'Pepe'
```

```
len(v) → 4
```

+

Concatena dos cadenas "Hola" + "Mundo"

Repite una cadena "Hola"*4

```
v = 'Pepe'
```

```
v*4 → PepePepePepePepe
```

in

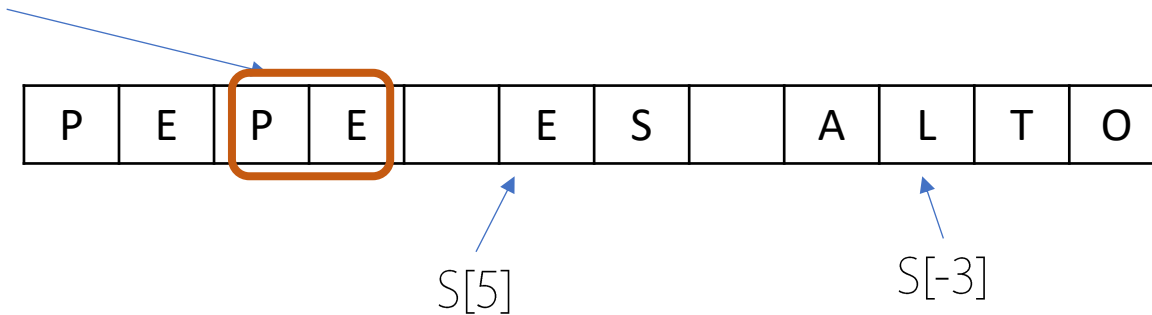
Devuelve True si una string está contenida en otra

Ej: name in texto → True/False

name not in texto

Más operaciones

- Ver un carácter concreto dentro de una cadena
- `s[5]`: El carácter que ocupa la posición 6 (empieza a contar por 0)
`s = "PEPE ES ALTO"`
`s[0]` → Letra inicial
`S[len(s)-]` → Letra final
- `s[-4]`: El carácter a 4 posiciones antes del final
- `S[2:4]`: Una subcadena desde `s[2]` a `s[4-1]`



- `s.funcion()`
 - `name = "paco"`
 - `name.upper()`
 - `name.count("co")`
- CUIDADO CON LA NOTACIÓN !!

Method	Description
<code>lower ()</code>	returns a string with every letter of the original in lowercase
<code>upper ()</code>	returns a string with every letter of the original in uppercase
<code>replace (x, y)</code>	returns a string with every occurrence of x replaced by y
<code>count (x)</code>	counts the number of occurrences of x in the string
<code>index (x)</code>	returns the location of the first occurrence of x
<code>isalpha ()</code>	returns True if every character of the string is a letter

Todos los métodos de la clase string

Note: All string methods returns new values. They do not change the original string.

Method	Description
<code>capitalize()</code>	Converts the first character to upper case
<code>casefold()</code>	Converts string into lower case
<code>center()</code>	Returns a centered string
<code>count()</code>	Returns the number of times a specified value occurs in a string
<code>encode()</code>	Returns an encoded version of the string
<code>endswith()</code>	Returns true if the string ends with the specified value
<code>expandtabs()</code>	Sets the tab size of the string
<code>find()</code>	Searches the string for a specified value and returns the position of where it was found
<code>format()</code>	Formats specified values in a string
<code>format_map()</code>	Formats specified values in a string
<code>index()</code>	Searches the string for a specified value and returns the position of where it was found
<code>isalnum()</code>	Returns True if all characters in the string are alphanumeric
<code>isalpha()</code>	Returns True if all characters in the string are in the alphabet
<code>isascii()</code>	Returns True if all characters in the string are ascii characters
<code>isdecimal()</code>	Returns True if all characters in the string are decimals
<code>isdigit()</code>	Returns True if all characters in the string are digits
<code>isidentifier()</code>	Returns True if the string is an identifier
<code>islower()</code>	Returns True if all characters in the string are lower case
<code>isnumeric()</code>	Returns True if all characters in the string are numeric

<code>isprintable()</code>	Returns True if all characters in the string are printable
<code>isspace()</code>	Returns True if all characters in the string are whitespaces
<code>istitle()</code>	Returns True if the string follows the rules of a title
<code>isupper()</code>	Returns True if all characters in the string are upper case
<code>join()</code>	Converts the elements of an iterable into a string
<code>ljust()</code>	Returns a left justified version of the string
<code>lower()</code>	Converts a string into lower case
<code>lstrip()</code>	Returns a left trim version of the string
<code>maketrans()</code>	Returns a translation table to be used in translations
<code>partition()</code>	Returns a tuple where the string is parted into three parts
<code>replace()</code>	Returns a string where a specified value is replaced with a specified value
<code>rfind()</code>	Searches the string for a specified value and returns the last position of where it was found
<code>rindex()</code>	Searches the string for a specified value and returns the last position of where it was found
<code>rjust()</code>	Returns a right justified version of the string
<code>rpartition()</code>	Returns a tuple where the string is parted into three parts
<code>rsplit()</code>	Splits the string at the specified separator, and returns a list
<code>rstrip()</code>	Returns a right trim version of the string
<code>split()</code>	Splits the string at the specified separator, and returns a list
<code>splitlines()</code>	Splits the string at line breaks and returns a list
<code>startswith()</code>	Returns true if the string starts with the specified value
<code>strip()</code>	Returns a trimmed version of the string
<code>swapcase()</code>	Swaps cases, lower case becomes upper case and vice versa
<code>title()</code>	Converts the first character of each word to upper case
<code>translate()</code>	Returns a translated string
<code>upper()</code>	Converts a string into upper case
<code>zfill()</code>	Fills the string with a specified number of 0 values at the beginning

Otros tipos

Tuplas

```
t = (1, 'Angel', 3.5)
```

Se acceden como listas `t[2]`

Son invariables

Set

```
s = {2,3,4,5}
```

No se acceden como listas (ERROR)

Operadores (¡¡¡solamente estos!!!)

`add`

`remove`

`in`

Funciones

- Permiten definir un conjunto de operaciones que se van a repetir mucho en un programa.
- Ahorra escribir código
- La estructura del código es más limpia.

```
def suma(a,b):  
    result = a + b  
    return(result)
```

```
def suma(a = 1, b= 2):  
    ...
```

```
c = suma()  
d = suma(34)
```

- Función main y parámetros

```
from absl import app  
from absl import flags
```

```
FLAGS = flags.FLAGS  
flags.DEFINE_string('file', './fichero.txt',  
                    'Fichero donde ...')
```

```
def main(argv):  
    file = FLAGS.file  
  
    # Cuerpo de la función main
```

```
if __name__ == '__main__':  
    app.run(main)
```

```
>> python filename.py --file "miscosas.txt"  
>> python filename.py --help
```

Ejercicios

Lista3: recorrer la lista siguiente:



```
lista = [(1, "Nombre"), (2, "Apellidos"), (3, 21), (4, 10)]
```

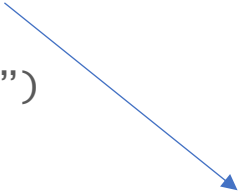
Hacer una función que recibe como parámetro cada elemento de la lista y para cada tupla formada por (a,b):

- Imprimir a
- Si b es string imprimirlo igual y si es entero imprimir el doble
- Devuelve el 1 si ha impreso string y 0 si ha impreso el doble de un entero.

El programa principal presentará en pantalla el número de string impresas cuando haya procesado la lista completa

Leer y escribir ficheros

```
f = open("demofile.txt", MODE)
f.read()
f.write("Texto a escribir")
f.close()
```



- "r" - Read - Default value. Opens a file for reading, error if the file does not exist
- "a" - Append - Opens a file for appending, creates the file if it does not exist
- "w" - Write - Opens a file for writing, creates the file if it does not exist
- "x" - Create - Creates the specified file, returns an error if the file exists

Comando with – Gestores de contexto

With

- Es un comando de los denominados gestores de contexto.
- Abren un bloque dentro del cual se ejecutan acciones. Al entrar del bloque se empiezan a usar recursos y al salir del bloque se liberan

Ejemplo

```
with open('fichero', 'w') as ofile:  
    ofile.write('Hola!')
```

```
file = open('fichero', 'w')  
file.write('Hola!')  
file.close()
```

Tipo de datos dict (similar a JSON)

- Una lista almacena muchos valores, incluso de tipos diversos.
- ¿Y si necesito estructurar datos de tipos variados?

Nombre

Apellido

Edad

Nota.

```
mydict = { 'nombre': 'Ana',  
           'apellidos': 'Perez',  
           'edad': 21,  
           'nota': 9.8  
}
```

```
mydict = dict(nombre='Ana', apellidos='Perez', edad=21,nota=9.8)
```

```
mydict['nombre']
```

Además, listas de dict

```
>>> mydict1 = { 'nombre': 'Ana', 'apellido':'Perez','edad':21}
>>> mydict2 = { 'nombre': 'Ignacio', 'apellido':'Gil','edad':19}
>>> ld = [mydict1, mydict2]
>>> ld[1]['edad']
19
```

Y los campos de un dict pueden ser una lista, claro...

```
>>> md = { 'nombre': 'Ana', 'apellidos': 'Perez','mods': ['PSP','FOL']}
>>> md['mods']
['PSP', 'FOL']
>>> md['mods'][0]
'PSP'
```

Librerías de Python

- No somos nadie sin librerías
- Más habituales
 - Numpy, pandas, time, json, ...
- Usaremos
 - Threads, aiosync, multiprocessing, websocket, ...

Cargo la librería en mi código

uso:

```
# sin cambio
```

```
import numpy
```

```
numpy.array(0)
```

```
# con un nombre para usarla
```

```
import numpy as np
```

```
np.array
```

```
#carga 1 clase de la libreria
```

```
from multiprocessing import Process
```

```
Process.run(fn)
```

ACTIVIDAD:

Instalar numpy

Programa que multiplique 2 matrices de 3x3 con y sin numpy.



Manejo de excepciones

```
try:
    #Bloque con el código que puede dar error
except:
    #Acción a realizar en caso de error
finally:
    #Acción a realizar siempre
```

El sistema incorpora una serie de códigos de error

- Referencia para Python 3.10
<https://docs.python.org/3/library/exceptions.html>

Algunos errores

- OverflowError, ZeroDivisionError, RuntimeError, OSError, TypeError,

```
s = eval(input("Introduce n: "))
a = 300
res = "NaN"
try:
    res = a/s
except:
    print("div cero")
    exit(0)
finally:
    print(res)
```

Python Orientado a Objeto

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

```
class Dog:
    species = "Canis familiaris"

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def edad(self):
        return(f"{self.name} tiene {self.age} años")
```

```
miPerro = Dog("cuchicuchi", 2)
```

```
miPerro.edad()
```

Ejercicio de POO



- Crear en un fichero banco.py con una clase cuenta
- Clase cuenta
 - Datos
 - Nombre del cliente
 - Número de cuenta
 - Saldo
 - Movimientos: Lista de: fecha (string), "tipo de movimiento" (string), cantidad (float).
Tipo de movimiento será uno de: "apertura", "ingreso", "pago", "cierre"
 - Métodos
 - Creación del objeto: Parametro de entrada (nombre, fecha) →
 - Crea un número de cuenta de 10 dígitos usando `str(hash(nombre))[-10:]`
 - Pone un movimiento inicial de apertura
 - **ingreso** → acepta la cantidad, la suma al saldo y registra el movimiento
 - **reintegro** → acepta la cantidad, la suma al saldo y registra el movimiento
 - **verSaldo** → devuelve el saldo.
 - **verCliente** → devuelve el nombre del cliente.
- Desde otro fichero importar la clase y:
 - Hacer una función transferencia
 - En el programa main
 - Crear 2 clientes.
 - Ingresar 10.000 € a cada uno
 - Imprimir por pantalla el nombre de cada cliente y el saldo de su cuenta.
 - Transferir 4500€ de una cuenta a la otra
 - Imprimir por pantalla el nombre de cada cliente y el saldo de su cuenta.

Más de OO, herencia y demás.

Variables de Clase

- Cualquier variable definida fuera de función puede ser usada como variable de clase.
 - Si se accede con un objeto, su valor será el local al objeto.
 - Si se accede con CLASE.VAR, su valor será el de clase

Ver ejemplo

```
class AlumnoDAM:
    nota = 0

    def __init__(self, aname, aage):
        self.name = aname
        self.age = aage

    def setNota(self, n):
        self.nota = n
        if AlumnoDAM.nota == 0:
            AlumnoDAM.nota = n
        else:
            AlumnoDAM.nota = (AlumnoDAM.nota+n)/2

def main():
    a = AlumnoDAM("Ana", 21)
    b = AlumnoDAM("Luis", 20)
    a.setNota(9)
    b.setNota(6)
    print(a.nota, ' ', b.nota, ' ', AlumnoDAM.nota)
main()
```

name y age
son locales a
cada objeto.

Clases abstractas

Python no soporta directamente clases abstractas pero existe una librería que permite hacerlo.

- Librería abc (abstract base class).
<https://docs.python.org/3/library/abc.html>
- Una clase es abstracta si incorpora un método abstracto.
 - Se marca con un "decorator"
 - *Un decorator es un patrón de diseño en Python que permite al usuario añadir nuevas funcionalidades a un objeto existente sin modificar su estructura.*

Nota para programadores Java:

Si queremos hacer una lista de polígonos en Java y que cada polígono sepa calcular su área necesitamos una estructura de herencia como la reflejada en el ejemplo.

Sin embargo, en Python, dado que las listas pueden ser de tipos variados podríamos prescindir de la clase base y el programa funcionaría igual.

Es decir, en un lenguaje no tipado son menos necesarias las clases abstractas.

```
class Poligono(abc.ABC):
    @abc.abstractmethod
    def area(self):
        pass

class Triangulo(Poligono):
    def area(self):
        print("tengo 3 caras")

class Pentagono(Poligono):
    def area(self):
        print("tengo 5 caras")

def main():
    l = []
    l.append(Triangulo())
    l.append(Pentagono())

    for p in l:
        p.area()
```

Ampliación

- List comprehension:
 - <https://docs.python.org/3.10/tutorial/datastructures.html#list-comprehensions>
 - <https://www.geeksforgeeks.org/python-list-comprehension/>
- Context Managers:
 - https://book.pythontips.com/en/latest/context_managers.html
- Algunas librerías:
 - Pandas: <https://pandas.pydata.org/>
 - Numpy: <https://numpy.org/>
 - Matplotlib: <https://matplotlib.org/>
 - Requests: <https://pypi.org/project/requests/>
 - FastAPI: <https://fastapi.tiangolo.com/>
 - ...

UT 1bis – Lenguaje de Programación Python

Fin