[Introducción](#_dsdbv945at5v)

[Documentación y Herramientas](#_bo842qwcyqm9)

[Librerías de python](#_sasz23ydbzcm)

[PARTE I](#_ytfq6ty6lb6v)

[Listas](#_b7vm0uj20fs1)

[Tuplas](#_aa46xe2qbvf5)

[Diccionarios](#_60zg4qklm0v2)

[Conjuntos](#_ufaygl41iapc)

[Condiciones](#_jbofvg3n9yz4)

[Patrones de diseño](#_xlhrqblc5qw4)

[Funciones I](#_lad7iavs6y9d)

[Argumentos](#_qrgvj3kv0gbj)

[Manejo de errores](#_b769b820fr5e)

[Archivos](#_cy0e82c7o7i1)

[Gestionar los permisos de los archivos](#_pgep0za2w0ag)

[PyQT (Visualización)](#_kyayabsjmu4)

[Pandas](#_jrf2irikconf)

# Introducción

Consultas relacionado con dudas de Lanbide: erandio@lanbide.eus

ChatGPT tiene un delay de 18 meses. 6 para entrenar , 6 verificar y 6 para liberar

Stackoverflow es la plataforma más actualizada por ello para la resolución de muchos problemas

# Documentación y Herramientas

Documentación del curso: [bit.ly/laner\_python](http://bit.ly/laner_python)

Pizarra: [bit.ly/laner\_meet](http://bit.ly/laner_meet)

Herramientas si se usa windows

Instalar Anaconda: <https://www.anaconda.com/download/success> (instala librerías por defecto y las más habituales pero existen más y es bueno cotillearlas)

* Spider (específico ciencia de datos): Entorno de desarrollo específico para ciencia de datos también vale VSC
* Jupiter notebook
* Google collab
* VSC
* Python Tutor (compilador): <https://pythontutor.com/python-compiler.html#mode=edit>
* Documentación de strings methods: <https://www.w3schools.com/python/python_ref_string.asp>

# Librerías de python

* Oficial Python: <https://docs.python.org/3/library/index.html>
* Librería Matplotlib: <https://matplotlib.org/>
* Pandas: <https://pandas.pydata.org/>
* <https://laravel.com/>

Para linux con instalar python que ya viene por defecto y VSC es suficiente, aunque es importante que hay que añadir python .exe al path, de esta manera puedes encontrar el ejecutable al path , si es en windows habría que “saltarse los 250 caracteres”

En terminal al poner python en terminal te lleva al path y el prompt es >>> al final, para llegar aquí es mejor tener un poco más de nivel, es mejor ir al entorno de desarrollo de Anaconda que te facilita la vida.

Para empezar hay que crear el proyecto en Spyder

Ruta windows : C:\Users\rportatil112\PYTHON\_BÁSICO

sincronizado con el drive local / MEGA

Para anular una comilla dentro de un string se pone \

# PARTE I

# Listas

Son datos ordenados

Las posiciones si quiero modificar por la derecha utilizo un valor negativo si quiero modificar por la izquierda utilizo un valor positivo

*#importar un alias como librería*

*import copy as cp*

*#crear variable y hacer copia de la variable*

*lista = [ 1, 'Bilbao', 'Ana', 'Orestes', 2.5]*

*copia\_lista = lista*

*#modificar la lista por la izquierda y por la derecha*

*lista[2] = 'Sevilla'*

*lista [-2] = 'Perro'*

Visualización: [Modificar valores en lista](https://pythontutor.com/render.html#code=import%20copy%20as%20cp%0A%0Alista%20%3D%20%5B%201,%20'Bilbao',%20'Ana',%20'Orestes',%202.5%5D%0Acopia_lista%20%3D%20lista%0A%0Alista%5B2%5D%20%3D%20'Sevilla'%0Alista%20%5B-2%5D%20%3D%20'Perro'&cumulative=false&heapPrimitives=nevernest&mode=edit&origin=opt-frontend.js&py=3&rawInputLstJSON=%5B%5D&textReferences=false)

# Tuplas

Se distinguen por el paréntesis y son inmutables, una vez definida la lista no se puede modificar el tamaño del contenedor y tampoco el contenido de manera directa, sin embargo puedes modificarlos en caso de que hagas un lista de la tupla (ya que sus valores se convierten en lista) y es como si tuvieses un pequeño backup con la tupla con valores que puedes restaurar o copiar si necesitas (un txiki-GIT)

*lista = ['elemento1','elemento2']*

*alias = lista*

*tupla = (1, 2, 'Python', True, 3.14,lista)*

*lista =’Asigno\_otroValor!' #en este ejemplo reasignamos el valor de lista a Reasignada para demostrar que una lista no se cambia sino que se reasigna, es decir el valor no se sustituye*

*tupla[5].append('elemento 3')* #recordemos que la posición 1 es igual a 0 y aquí vemos como podemos modificar desde la ‘clase’ dupla dentro del ‘contenedor’ = 5 asignar valores a la lista

*alias.append('elemento 4')*

Visualización: [Tupla y anidación con lista](https://pythontutor.com/render.html#code=lista%20%3D%20%5B'elemento%201','elemento2'%5D%0Aalias%20%3D%20lista%0A%0Atupla%20%3D%20%281,%202,%20'Python',%20True,%203.14,lista%29%0A%0Alista%20%3D'Reasignada!'%0A%0Atupla%5B5%5D.append%28'elemento%203'%29%0Aalias.append%28'elemento%204'%29&cumulative=false&curInstr=5&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=3&rawInputLstJSON=%5B%5D&textReferences=false)

# Diccionarios

Son datos no ordenados que contienen los argumentos tipo clave:valor

# Conjuntos

Igual que el diccionario pero no son argumentos clave:valor, se CARGA LOS ARGUMENTOS DUPLICADOS por eso se suelen utilizar para eliminar duplicados.

El error "unhashable type" en Python generalmente ocurre cuando intentas usar un tipo de dato que no es "*hasheable*" como un elemento en un conjunto (set) o como clave en un diccionario (dict). Los tipos de datos que son inmutables, como las cadenas, los números y las tuplas (si contienen solo elementos inmutables), son *hasheables*. En cambio, los tipos mutables, como las listas y los diccionarios, no lo son.

# Condiciones

if…else

El if no siempre debe estar acompañado de else

Condición elif … es un punto intermedio entre if y else, else siempre va a dar una salida final,

*numero\_secreto = 7*

*adivinanza = None*

*while adivinanza != numero\_secreto:*

*adivinanza = int(input("¿qué número es?:"))*

*if adivinanza < numero\_secreto:*

*print("muy bajo")*

*elif adivinanza > numero\_secreto:*

*print("muy alto")*

*print("¡Zorionak!")*

i = iterar por índice

j = 2ª iteración sin perder el primero

k =

i++ = detiene el bucle

i+= = incrementa en 1

# Patrones de diseño

Try … except

Funciona como bucles y condiciones, andiables, con variables adjuntas a las funciones, etc,...

*try:*

*num = int(input("Introduce un número: "))*

*resultado = 10 / num*

*print("Resultado: " + str(resultado))*

*except ZeroDivisionError as e: # Manejar la excepción de división por cero*

*print("Error: No se puede dividir por cero.")*

*except ValueError as e: # Manejar la excepción si la entrada no es un número*

*print("Error: Debes introducir un número entero.")*

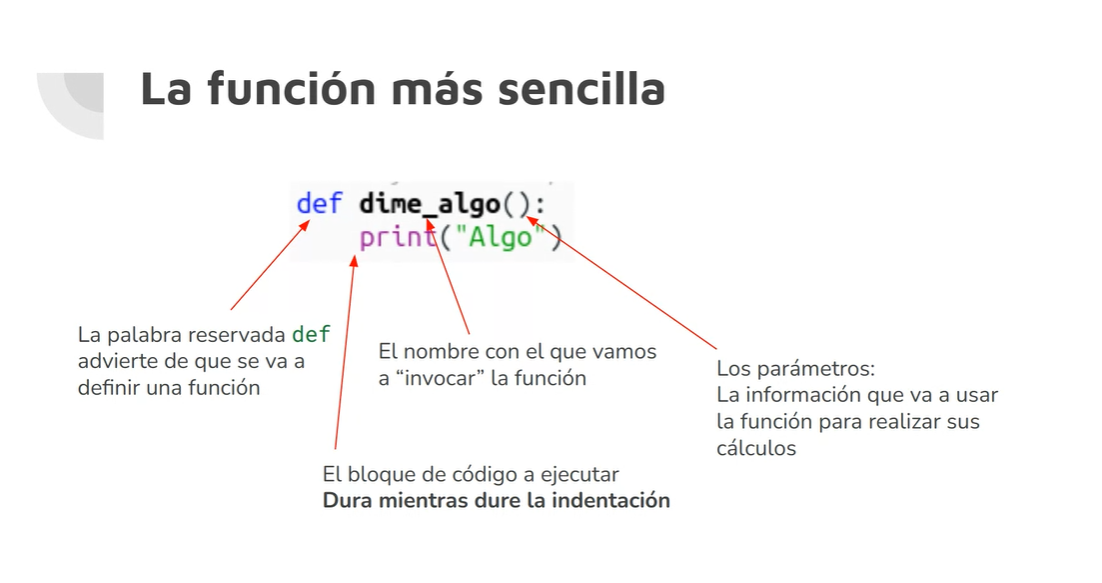
*finally:*

*print("Me ejecuto sí o sí.")*

Visualización: [Ejemplo para ZeroDivisionError](https://pythontutor.com/render.html#code=try%3A%0A%20%20%20%20num%20%3D%20int%28input%28%22Introduce%20un%20n%C3%BAmero%3A%20%22%29%29%0A%20%20%20%20%0A%20%20%20%20resultado%20%3D%2010%20/%20num%0A%20%20%20%20print%28%22Resultado%3A%20%22%20%2B%20str%28resultado%29%29%0A%0Aexcept%20ZeroDivisionError%20as%20e%3A%20%20%23%20Manejar%20la%20excepci%C3%B3n%20de%20divisi%C3%B3n%20por%20cero%0A%20%20%20%20print%28%22Error%3A%20No%20se%20puede%20dividir%20por%20cero.%22%29%0Aexcept%20ValueError%20as%20e%3A%20%20%23%20Manejar%20la%20excepci%C3%B3n%20si%20la%20entrada%20no%20es%20un%20n%C3%BAmero%0A%20%20%20%20print%28%22Error%3A%20Debes%20introducir%20un%20n%C3%BAmero%20entero.%22%29%0A%20%20%20%20%0Afinally%3A%0A%20%20%20%20print%28%22Me%20ejecuto%20s%C3%AD%20o%20s%C3%AD.%22%29%0A&cumulative=false&curInstr=7&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=311&rawInputLstJSON=%5B%220%22%5D&textReferences=false)

# Funciones I

Siempre detrás del nombre de la función siempre va a haber unos paréntesis que pueden estar vacíos o no, el contenido serán los argumentos que pueden ser indefinidos el contenido de la función se considera o se trata como una TUPLA.



La **indentación** es una tabulación de 4 espacios, a diferencia de otros lenguajes, en python el bloque de código a ejecutar no va entre corchetes, por eso es importante la indentación en python porque si no no entiende el código.

# Argumentos

Son los datos que se incluyen dentro del paréntesis de la función

Ejemplos de como hacer lo mismo de dos formas distintas:

*lista\_numeros = [1,2,3,4,5]*

*lista\_paridad = []*

*for numero in lista\_numeros:*

*lista\_paridad.append(es\_par(numero))*

*—----------------------------------------------------------*

*def es\_par\_lista(lista\_num)*

*lista\_paridad = []*

*for numero in lista\_num:*

*lista\_paridad.append(numero%2 == 0)*

*return lista\_paridad*

*es\_par\_lista(lista\_numero)*

*lista\_numeros = es\_par\_lista(lista\_numeros)*

Importante: el print no devuelve resultados, solo te lo saca por la pantalla, por lo tanto no ejecuta, por eso hay que tener cuidado que muchas veces puede haber una incongruencia entre lo que se ve por pantalla

Hay que evitar que las ejecuciones queden paradas, ya que te puede tirar el servidor, siempre es bueno hacer comprobaciones

Por otro lado también hay que tener cuidado con los booleanos y sus salidas ya que se convierten en enteros con facilidad

FUNCIONES II

Dentro de la tupla de la función podemos indicar que los argumentos se busquen por posición o por clave : valor.

Esto quiere decir que si se busca por posición python buscará en ese órden, pero si asigno valores (y son argumentos que no recuerdo su posición) es cuando le asignamos una clave valor

# Manejo de errores

El manejo de errores en python puede ser por “defecto” , es decir son errores ya predefinidos, sin embargo no solo podremos declarar o llamar a estos errores con la lógica correspondiente sino que también podemos generar nuestros propios mensajes de error personalizados, lo cual viene bien sobre todo para el tema de la localización, ya que los errores de Python estarán en inglés normalmente

TIP: la instrucción PASS digamos que se salta la ejecución de cualquier función o método, esto no sirve sobretodo para hacer pruebas

El manejo de errores es importante para poder predecir errores de uso en la aplicación y poder informar a la persona usuaria

# Archivos

+info: ppt 8 - Archivos

Para leer un .CSV se utiliza la librería CSV por lo que para importar seria:

**import csv**

*with open (“nombre\_archivo.csv”, modo) as nombre\_variable*:

variable\_lectura = csv.reader(nombre\_variable)

*with open (“nombre\_archivo.csv”, modo) as nombre\_variable*:

variable\_escritura = csv.writer(nombre\_variable)

\*modo: lectura, escritura, binario, … los permisos y alguna cosita más



OS y SYS son otras librerías (de sistema operativo (OS) y de sistema (SyS) para importar también, bueno como cualquier librería)

Open abre el archivo y lo lee pero no lo cierra, sin lo le pasas el método (write, read, etc,...) hay que cerrarlo “*manualmente*” (close open) si no te lo corrompe, por solo recomendable es la estructura explicada.

os.getcwd() = GETCurrentWorkingDirectory = método de OS

## Gestionar los permisos de los archivos

En Python, no hay un sistema de permisos integrado como en sistemas operativos como Linux o Windows. Sin embargo, hay algunas formas de gestionar permisos y acceso a recursos en Python:

**Modos de archivo**: El módulo os proporciona constantes para los modos de archivo, como os.R\_OK (lectura), os.W\_OK (escritura), os.X\_OK (ejecución). Puedes utilizar estas constantes para verificar o establecer los permisos de un archivo.

**chmod**: El módulo os también proporciona una función chmod que permite cambiar los permisos de un archivo o directorio. Por ejemplo, os.chmod('ruta/al/archivo', 0o644) establece los permisos de lectura y escritura para el usuario y grupo, y solo lectura para otros usuarios.

**Sistema de archivos virtual**: Si estás desarrollando una aplicación que requiere un sistema de archivos virtual, puedes utilizar una biblioteca como pathlib o pyfilesystem2 para crear un sistema de archivos en memoria. En este caso, puedes implementar tus propios mecanismos de permisos y acceso a recursos.

**Autenticación y autorización**: Si necesitas autenticar y autorizar usuarios para acceder a recursos, puedes utilizar bibliotecas como auth o oauthlib. Estas bibliotecas te permiten verificar la identidad de los usuarios y controlar el acceso a recursos según sus permisos y roles.

Ejemplos de código:

Verificar permisos de lectura en un archivo:

import os

*if os.access('ruta/al/archivo', os.R\_OK):*

*print("El archivo es readable")*

Establecer permisos de escritura en un archivo:

import os

*os.chmod('ruta/al/archivo', 0o666*) # establece permisos de escritura para todos

Crear un sistema de archivos virtual con permisos:

import pathlib

vfs = pathlib.VFS() # crea un sistema de archivos virtual

vfs.mkdir('ruta/al/directorio', 0o755) # crea un directorio con permisos 755

vfs.touch('ruta/al/archivo', 0o644) # crea un archivo con permisos 644

# PyQT (Visualización)

requirements es un librería/clase/funcion que te resume las versiones instaladas en un proyecto (dependencias) de cara a compartir con un equipo u otras personas

# Pandas

Esta librería es la estandard en el mundo profesional de la ciencia de datos, también hay otra llamada Polars pero es menos utilizada, quizás a futuro pueda aportar más valor

Polars (alternativa a PANDAS)