

Curso de Python Intermedio



Facundo García Martoni
 @facmartoni





Curso Básico de Python

★★★★★ 5125 Opiniones

BÁSICO

Inicia en el mundo de la programación con el lenguaje de mayor crecimiento en el planeta: Python. Descubre qué es un algoritmo, y cómo se construye uno. Domina las variables, funciones, estructuras de datos, los condicionales y ciclos.

- Hacer estructuras de datos
- Crear bucles
- Conocer herramientas para programar
- Aprender conceptos básicos de Python

CONTINUAR APRENDIENDO

+ AGREGAR A MI RUTA



Facundo García Martoni 
Technical Mentor en Platzi



Curso Profesional de Git y GitHub

★★★★★ 9589 Opiniones

BÁSICO

Deja de versionar tus proyectos usando tu propio sistema de control de versiones. Mejor usa Git, el sistema de control de versiones por excelencia que utiliza la industria tecnológica. Aprende a trabajar con git, conceptos básicos, clonar un repositorio y gestionar tus proyectos alojándolos en tu repositorio local y en GitHub.

- Llevar un Control de Versiones en tus Proyectos con Git
- Trabajar en Equipos de Forma Colaborativa
- Utilizar Dominios Personalizados con GitHub Pages
- Instalar Git en tu sistema operativo

CONTINUAR APRENDIENDO

+ AGREGAR A MI RUTA



Freddy Vega 
CEO en Platzi

El Zen de Python

¿Qué es?



Bello es mejor que feo.

Explícito es mejor que implícito.


Simple es mejor que complejo.

Complejo es mejor que complicado.

Plano es mejor que anidado.

Espaciado es mejor que denso.

La legibilidad es importante.



Los casos especiales no son lo suficientemente especiales como para romper las reglas.

Sin embargo la practicidad le gana a la pureza.

Los errores nunca deberían pasar silenciosamente.

A menos que se silencien explícitamente.

Frente a la ambigüedad, evitar la tentación de adivinar.



Debería haber una, y preferiblemente solo una, manera obvia de hacerlo.

A pesar de que esa manera no sea obvia a menos que seas holandés.

Ahora es mejor que nunca.

A pesar de que nunca es muchas veces mejor que *ahora* mismo.



Si la implementación es difícil de explicar, es una mala idea.

Si la implementación es fácil de explicar, puede que sea una buena idea.

Los espacios de nombres son una gran idea, ¡tenemos más de esos!

¿Qué es la documentación?



Y por qué es tan importante

¿Qué es un entorno virtual?

Controla tus módulos

Mi computadora

Python

Módulo 1

Módulo 2

Módulo 3

Proyecto 1

Proyecto 2

Proyecto 3

Proyecto 4

Mi computadora

Python

Módulo 1

Módulo 2 v2

Módulo 3

Proyecto 1

Proyecto 2

Proyecto 3

Proyecto 4

Mi computadora

Python

Proyecto 1

Módulo 1

Módulo 2

Módulo 3

Python

Proyecto 2

Módulo 1

Módulo 2

Módulo 3

Python

Proyecto 3

Módulo 1

Módulo 2

Módulo 3

Python

Proyecto 4

Módulo 1

Módulo 2

Módulo 3

Mi computadora

Python

Proyecto 1

Módulo 1

Módulo 2 v2

Módulo 3

Python

Proyecto 2

Módulo 1

Módulo 2 v2

Módulo 3

Python

Proyecto 3

Módulo 1

Módulo 2 v2

Módulo 3

Python

Proyecto 4

Módulo 1

Módulo 2

Módulo 3

Python

Proyecto 1

Módulo 1

Módulo 2 v2

Módulo 3

El primer paso profesional: creación de un entorno virtual

Controla tus módulos

Instalación de dependencias con PIP



Package **I**nstaller for **P**ython



Módulos populares

- Requests
- BeautifulSoup4
- Pandas
- Numpy
- Pytest

PIP

PYENV



PIP

PYENV



Una alternativa: Anaconda

Una distribución especial de Python



ANACONDA[®]

Listas y diccionarios anidados



List comprehensions

Genera listas sin ciclos

```
[element for element in iterable if condition]
```

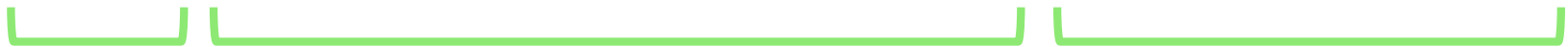
```
[element for element in iterable if condition]
```

Representa
a cada uno
de los
elementos
a poner en
la nueva
lista

Ciclo a partir del cual se extraerán
elementos de otra lista o cualquier
iterable

Condición
opcional para
filtrar los
elementos del
ciclo

```
[i**2 for i in range(1, 101) if i % 3 != 0]
```



Representa
a cada uno
de los
elementos
a poner en
la nueva
lista

Ciclo a partir del cual se extraerán
elementos de otra lista o cualquier
iterable

Condición
opcional para
filtrar los
elementos del
ciclo

Reto

- Crear, con un list comprehension, una lista de todos los múltiplos de 4 que a la vez también son múltiplos de 6 y de 9, hasta 5 dígitos.



Reto

[36, 72, 108, 144, 180, 216...]

Dictionary comprehensions

Genera diccionarios sin ciclos

```
{key:value for value in iterable if condition}
```



```
{key:value for value in iterable if condition}
```

Representa a
cada una de
las llaves y
valores a
poner en el
nuevo
diccionario

Ciclo a partir del cual se
extraerán elementos de
cualquier iterable

Condición
opcional para
filtrar los
elementos del
ciclo

```
{i: i**3 for i in range(1, 101) if i % 3 != 0}
```

Representa
a cada una
de las llaves
y valores a
poner en el
nuevo
diccionario

Ciclo a partir del cual se extraerán
elementos de cualquier iterable

Condición
opcional para
filtrar los
elementos del
ciclo

Reto

- Crear, con un dictionary comprehension, un diccionario cuyas llaves sean los 1000 primeros números naturales con sus raíces cuadradas como valores.



Reto

$\{1: 1.0, 2: 1.4142135623730951...\}$

Funciones anónimas



Y qué significa “lambda”



```
lambda argumentos: expresión
```



```
palindrome = lambda string: string == string[::-1]  
print(palindrome('ana'))
```



True


identificador

argumento

expresión

```
palindrome = lambda string: string == string[::-1]
```

```
print(palindrome('ana'))
```



True



retorna un objeto de tipo función

```
palindrome = lambda string: string == string[::-1]
```

```
print(palindrome('ana'))
```



True



```
# palindrome = lambda string: string == string[::-1]

def palindrome(string):
    return string == string[::-1]

print(palindrome('ana'))
```



True

High order functions



Filter, map y reduce



Función de orden superior

Es una función que recibe como parámetro a otra función.



```
def saludo(func):  
    func()  
  
def hola():  
    print("Hola!!!")  
  
def adios():  
    print("Adios!!!")  
  
saludo(hola)  
saludo(adios)
```



```
Hola!!!  
Adios!!!
```

filter

[1, 4, 5, 6, 9, 13, 19, 21]



[1, 5, 9, 13, 19, 21]



```
my_list = [1, 4, 5, 6, 9, 13, 19, 21]  
odd = [i for i in my_list if i % 2 != 0]  
print(odd)
```



```
[1, 5, 9, 13, 19, 21]
```




```
# Uso con filter
```

```
my_list = [1,4,5,6,9,13,19,21]
```

```
odd = list(filter(lambda x: x%2 != 0, my_list))
```

```
print(odd)
```



```
[1, 5, 9, 13, 19, 21]
```

—

map

[1, 2, 3, 4, 5]



[1, 4, 9, 16, 25]



```
my_list = [1, 2, 3, 4, 5]  
squares = [i**2 for i in my_list]  
print(squares)
```



```
[1, 4, 9, 16, 25]
```



```
# Uso con map

my_list = [1, 2, 3, 4, 5]

squares = list(map(lambda x: x**2, my_list))

print(squares)
```



```
[1, 4, 9, 16, 25]
```

—

reduce

[2, 2, 2, 2, 2]



32



```
my_list = [2, 2, 2, 2, 2]

all_multiplied = 1

for i in my_list:
    all_multiplied = all_multiplied * i

print(all_multiplied)
```



32



```
# Use con reduce
```

```
from functools import reduce
```

```
my_list = [2, 2, 2, 2, 2]
```

```
all_multiplied = reduce(lambda a, b: a * b, my_list)
```

```
print(all_multiplied)
```



```
32
```

Proyecto: filtrando datos

¡Pongamos en práctica lo aprendido!

```
DATA = [  
    {  
        'name': 'Facundo',  
        'age': 72,  
        'organization': 'Platzi',  
        'position': 'Technical Coach',  
        'language': 'python',  
    },  
    {  
        'name': 'Luisana',  
        'age': 33,  
        'organization': 'Globant',  
        'position': 'UX Designer',  
        'language': 'javascript',  
    }  
]
```

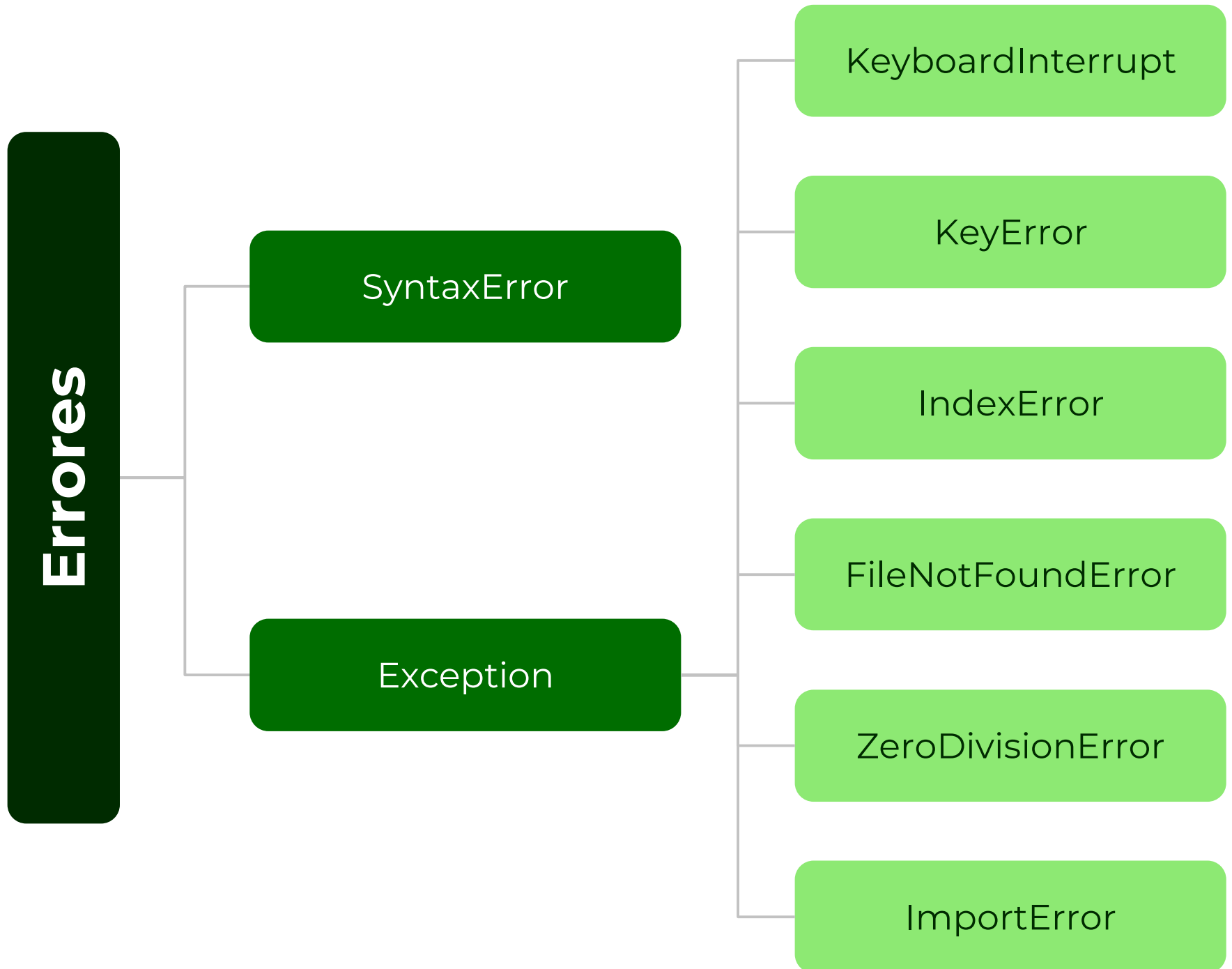
Reto

- Crear las listas `all_python_devs` y `all_Platzi_workers` usando una combinación de `filter` y `map`.
- Crear la lista `old_people` y `adults` con `list comprehensions`.

Los errores en el código



Y cómo manejarlos



```
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ZeroDivisionError: division by zero
```

Debugging




O también, depuración


Manejo de excepciones

raise, try, except y finally

try, except



```
def palindrome(string):  
    return string == string[::-1]  
  
print(palindrome(1))
```



```
Traceback (most recent call last):  
  File "main.py", line 4, in <module>  
    print(palindrome(1))  
  File "main.py", line 2, in palindrome  
    return string == string[::-1]  
TypeError: 'int' object is not subscriptable
```



```
def palindrome(string):  
    return string == string[::-1]  
  
try:  
    print(palindrome(1))  
except TypeError:  
    print("Solo se pueden ingresar strings")
```



Solo se pueden ingresar strings


raise



```
def palindrome(string):  
    return string == string[::-1]  
  
try:  
    print(palindrome(""))  
except TypeError:  
    print("Solo se pueden ingresar strings")
```



True



```
def palindrome(string):  
    try:  
        if len(string) == 0:  
            raise ValueError("No se puede ingresar una cadena vacía")  
        return string == string[::-1]  
    except ValueError as ve:  
        print(ve)  
        return False  
  
try:  
    print(palindrome(""))  
except TypeError:  
    print("Solo se pueden ingresar strings")
```



```
No se puede ingresar una cadena vacía  
False
```

—

finally



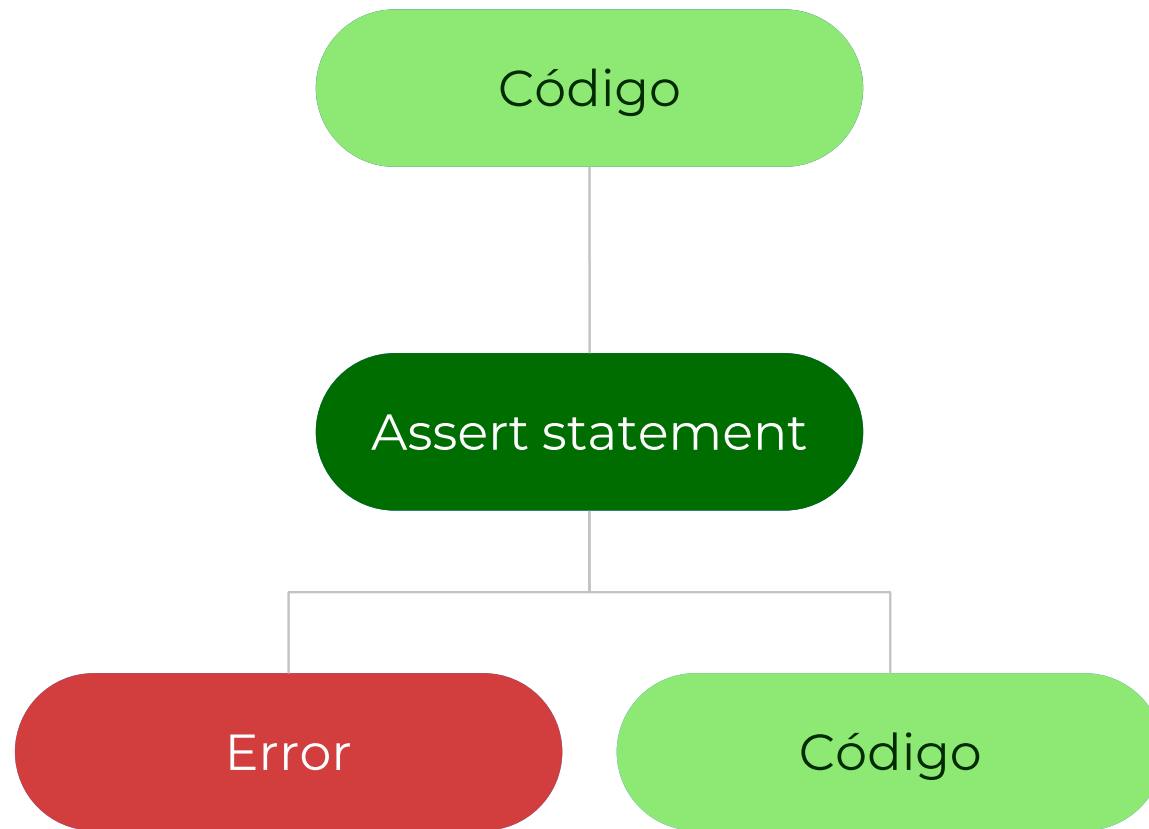
```
try:
    f = open("archivo.txt")
    # hacer cualquier cosa con nuestro archivo
finally:
    f.close()
```

Reto

- Utiliza las palabras clave `try`, `except` y `raise` para elevar un error si el usuario ingresa un número negativo en nuestro programa de divisores.

Assert Statements

Afirmaciones en Python





```
assert condición, mensaje de error
```



```
def palindrome(string):  
    return string == string[::-1]  
  
print(palindrome(""))
```



True



```
def palindrome(string):  
    assert len(string) > 0, "No se puede ingresar una cadena vacía"  
    return string == string[::-1]  
  
print(palindrome(""))
```



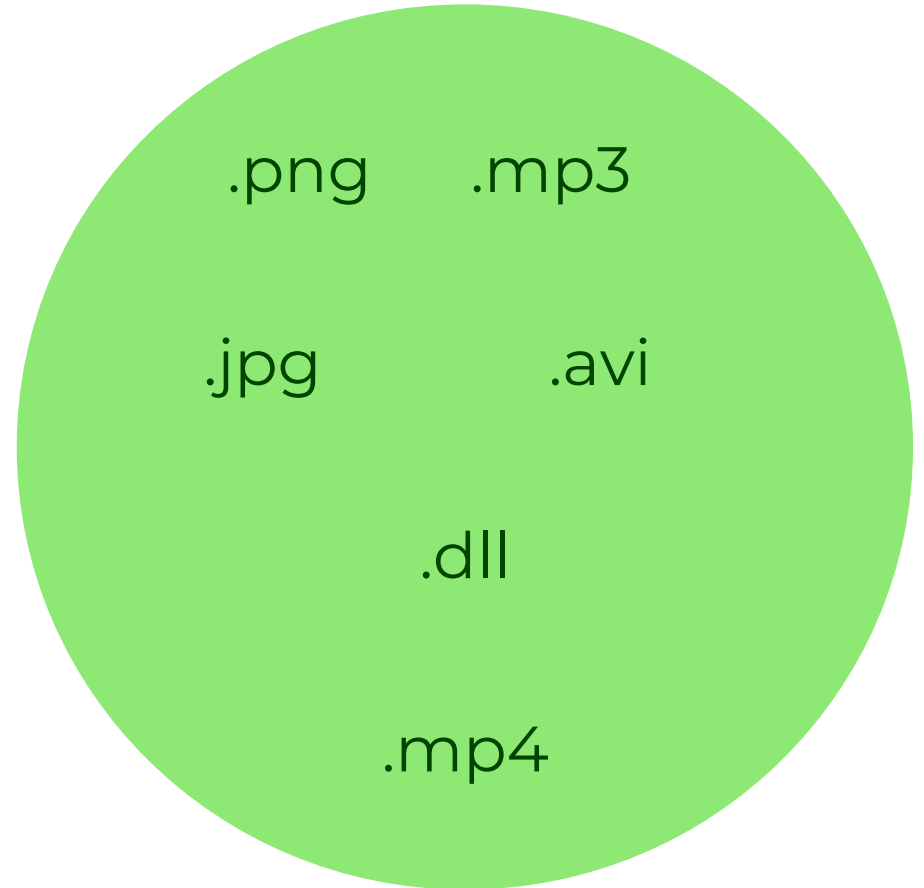
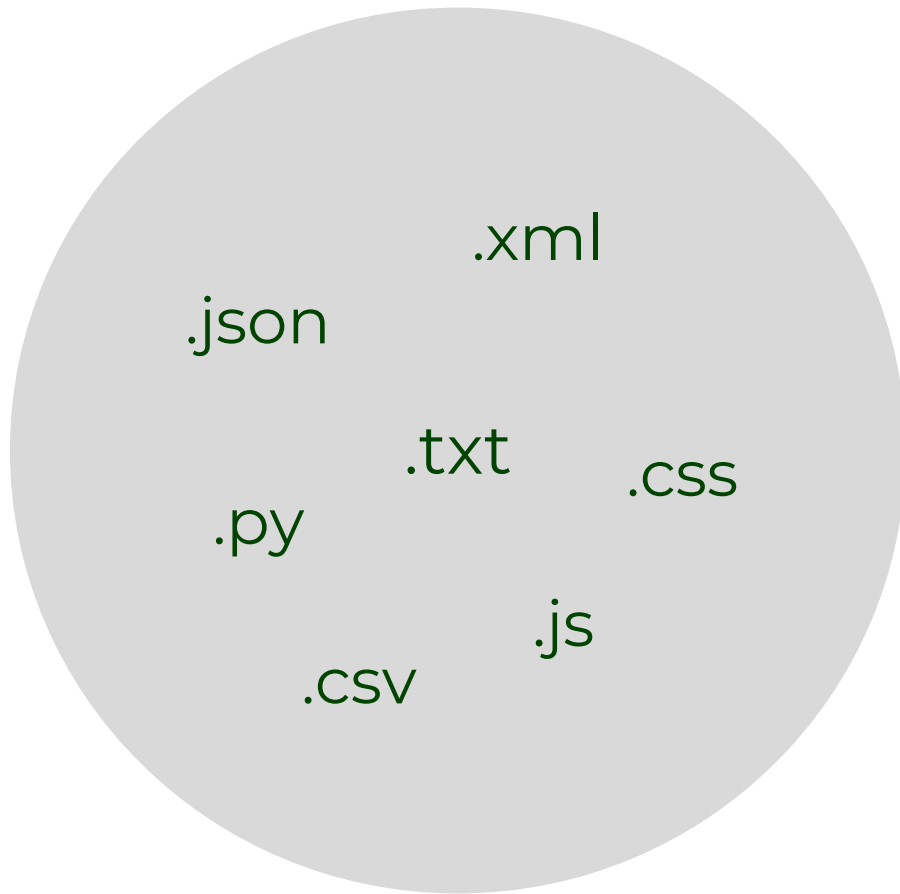
AssertionError: No se puede ingresar una cadena vacía

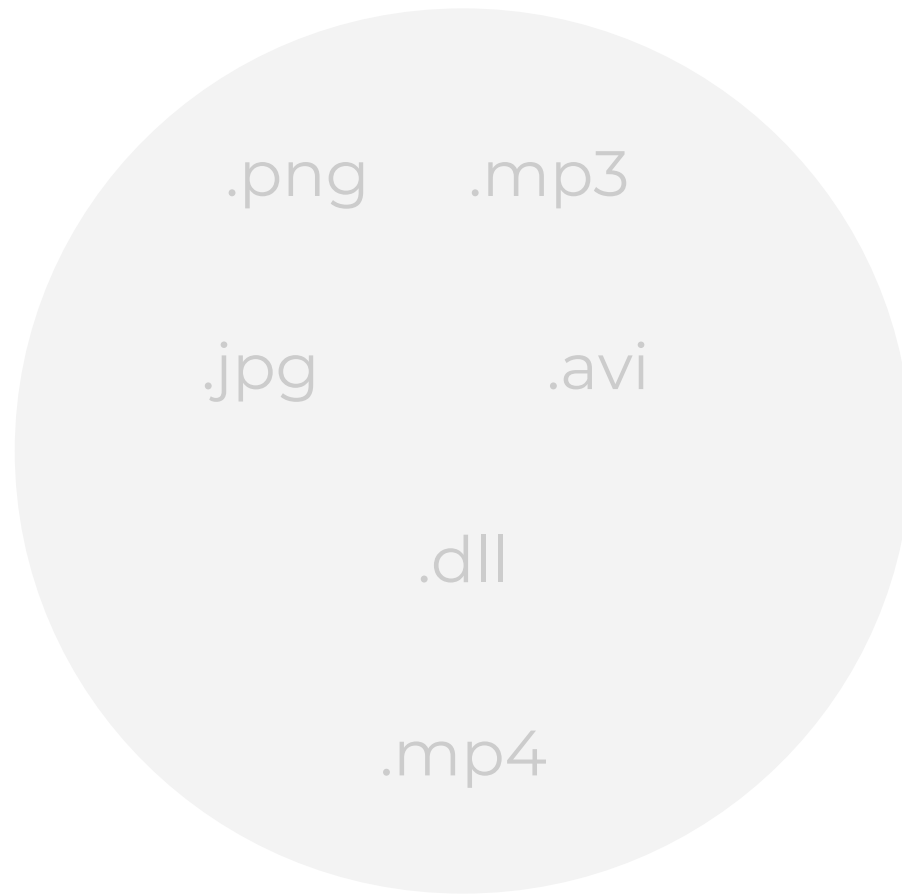
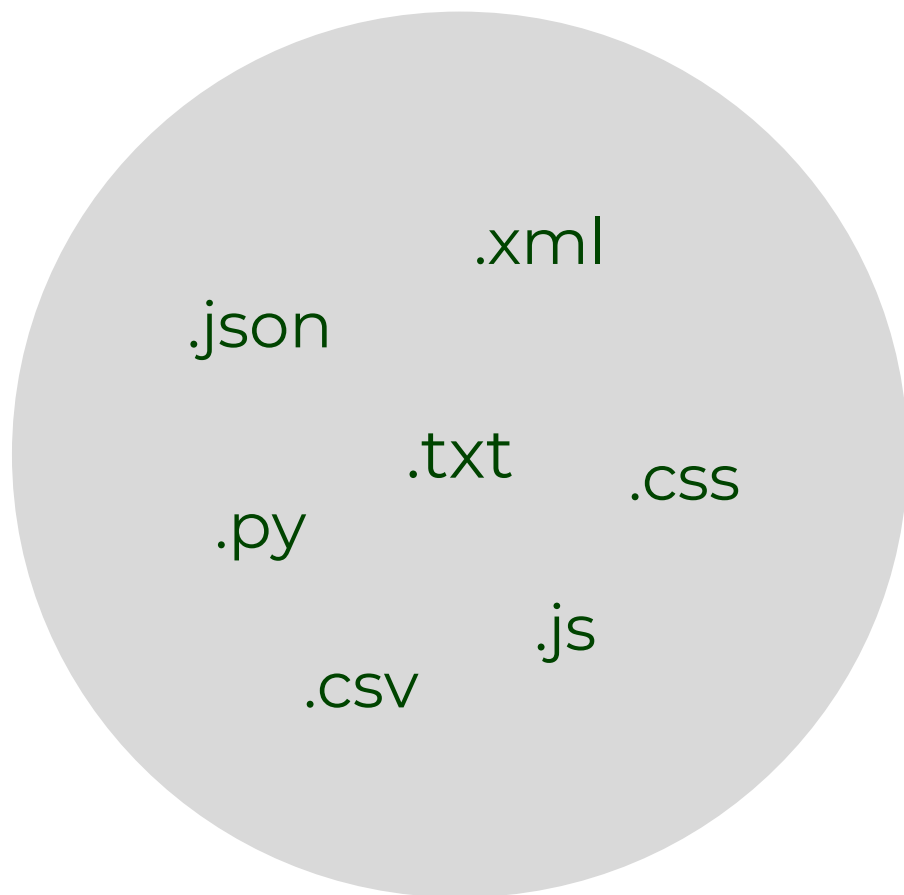
Reto

- Utiliza `assert statements` para evitar que el usuario ingrese un número negativo en nuestro programa de divisores.

¿Cómo trabajar con archivos?

Lectura y escritura





Modos de apertura

- **R** -> Lectura
- **W** -> Escritura (sobrescribir)
- **A** -> Escritura (agregar al final)



```
with open("./ruta/del/archivo.txt", "r") as f:
```

Reto final: Juego del Ahorcado o Hangman Game



Tu última prueba

Reglas

- Incorpora comprehensions, manejo de errores y manejo de archivos.
- Utiliza el archivo `data.txt` y léelo para obtener las palabras.

Ayudas y pistas

- Investiga la función `enumerate`.
- El método `get` de los diccionarios puede servirte.
- La sentencia
 `os.system("cls")` -> Windows
 `os.system("clear")` -> Unix
te servirá para limpiar pantalla.

Mejora el juego

- Añade un sistema de puntos.
- Dibuja al “ahorcado” en cada jugada con código ASCII.
- Mejora la interfaz.

¡Hasta pronto!



Llegaste al final, pero aún no termina

¿Quieres preguntarme algo?



@facmartoni

facundonicolas.com