# Semana 0

## PEP 8

- snake_case para variables, atributos, funciones y métodos
- CamelCase para clases
- Nombres descriptivos de variables
- Imports al principio
- Espacios despues de ,
- Espacios a lados de operadores
- Uso de 4 espacios como indent
- Máximo de 100 caracteres por linea incluyendo espacios (Ejemplo de )

```
# forma incorrecta
if valor_1 + valor_2 >= 100 and (valor_1 - valor_2 >= 100 or valor_2 - valor_1 >= 100)
:
  pass
# forma correcta
if valor_1 + valor_2 >= 100 and \
(valor_1 - valor_2 >= 100 or valor_2 - valor_1 >= 100):
  pass
```

## Modulos

### Importación

- demo rapida

```
import modulo
modulo.funcion()

import modulo as md
md.funcion()

##parcial

from modulo import funcion, variable, Clase
funcion()

## no recomendado
from modulo import *
```

### Uso de *_name_*

- Sirve para ver el nombre real de un modulo, independiente del alias.
- Siempre es el nombre del archivo, excepto si es un archivo que esta siendo ejecutado como el main de un programa.
- Puede ser usado desde dentro de un modulo o como parte de una propiedad . del modulo.

```
## archivo main
import ejemplo2 as ej
ej.__name__ # ejemplo2
ej.mi_nombre # ejemplo2

if __name__ == "__main__":## ES VERDAD, SI SE EMPIEZA LA EJECUCION DESDE AQUI
    print("Soy el módulo principal")
## ejemplo2.py

mi_nombre = __name__ #ejemplo2

if __name__ == "__main__":## ESTO NO PASA SI EJEMPLO2 SE IMPORTA Y SE USA DESDE OTRO A
RCHIVO
    print("Soy el módulo principal")
```

### Uso real

- Se puede usar para separar funcionalidades distintas e independientes. Por ejemplo, contactos.py, pdf.py, emails.py
- Vuelve mas reutilizable el código

### Abstracción de componentes

- Cada módulo se vuelve una caja negra que provee funcionalidades. Ejemplo.. radio.. No importan su implementación, pero si poder usar sus funciones.

```
import random

print(random.randint(1, 10))
print(random.choice(["a", "b", "c"]))
```

## Listas

### Resumen

- Nos centramos en list
- list es ordenada, mutable. usa índices.

```
mi_lista = list() #o []

mi_lista.append(1)
```

```
## añadir varios elementos en una llamada
mi_lista.extend([1,5,7])

## Insertar en posición especifia
mi_lista.insert(3, 'ola')
# esto inserta y mueve el elemento ahi a la derecha
```

## List Comprehension

- Las listas por comprensión son listas formadas por un conjunto de objetos que cumplen con un concepto o condición particular.
- Forma:
  `nueva_lista = [expresión for elemento in lista]`
  `nueva_lista = [expresión for elemento in lista if condición]`
- Ejemplo:

```
importante = ["banana", "mono", "frutos", "uwu"]

largo_palabras_importantes = [len(palabra) for palabra in importante]

largo_sin_uwu = [len(word) for word in importante if word != "uwu"]

##equivalente sin comprehension
largo_sin_uwu = []
for word in importante:
  if word != "uwu":
    largo_sin_uwu.append(word)
```

## *Slicing*

- Permite sacar partes de una lista:
- Forma:
  `secuencia[inicio:término:pasos]`
- por terminar

## Sort

- `lista.sort(reverse=False)` ordena una lista

## Aparte: Tuplas

- Las tuplas son como listas pero inmutables

```
tuple_constructor = tuple(("dsa", "developement", "deep learning"))
mytuple = ("Geeks",)#cn la coma
```

```
var = ("Geeks", "for", "Geeks")
## ejemplos de tuplas validas
```

# Paths

- Módulo os: Se usa os.path para manejo de archivos, importar os.
- Path absoluto y relativo: trivial
- Por lo general se usan paths relativos, ya que no dependen de los directorios exactos del computador dónde se instala el programa.
- Un path es la ruta a un archivo o directorio.
- Se puede dividir en dirname y filename o basename.
- El dirname es la carpeta donde esta el archivo o directorio objetivo.
- El filename es el nombre del archivo con su extension, o directorio.
- Las funciones os.path.dirname() y os.path.basename() permiten extraerlos de un path como string.
- Ejemplo:

```
path1 = "/folder1/folder2/cat.jpg"

dirname1 = os.path.dirname(path1) # /folder1/folder2
basename1 = os.path.basename(path1) # cat.jpg

path2 = "/f1/f2/f3"

dirname2 = os.path.dirname(path1) # /f1/f2
basename2 = os.path.basename(path2) # f3
```

## File extensions

- Para extraer la extension de un path o basename, usamos `os.path.splitext()`.
- Retorna una lista o una tupla? pero es [0] todo lo demas, y [1] la extension

```
path1 = "/folder1/folder2/cat.jpg"

basename1 = os.path.basename(path1)# cat.jpg
path_without_ext, ext = os.path.splitext(path1) # ["/folder1/folder2/cat",".jpg"]
```

## Portabilidad de Paths (Entre OSes)

- Existen problemas de portabilidad, ya que por ejemplo "Users/Pedro/Libros/python.pdf" sirve en Unix, pero en Windows sería "Users\Pedro\Libros\python.pdf"
- Podemos usar os.path.join() para unirlo según el OS que ejecuta el programa.
- Ej:

```
path = os.path.join("Users", "endless", "monke", "banana.png")
path # "Users/endless/monke/banana.png" from Unix
```

## Navegación entre directorios

- `os.listdir(path)` permite listar los archivos, es como el comando dir.
- `os.walk("dir", topdown=True)` comando para obtener rutas de directorio, subdirectorio y archivos recursivamente.
- Con topdown=True, parte del raiz, y luego se va metiendo a los otros.
- Con topdown=False, parte listando desde los directorios mas interiores.
- Cada ciclo del for es un directorio como raiz, y muestra la raiz actual, los directorios dentro, y los archivos directamente dentro (de la raíz, no adentro de los demas).
- Ejemplo (pendiente crear ej propio)

```
for raiz, directorios, archivos in os.walk("data", topdown=True):
    print("Raíz:", raiz)
    print()
    print("Archivos:")
    for archivo in archivos:
        print(os.path.join(raiz, archivo))
    print()
    print("Directorios:")
    for directorio in directorios:
        print(os.path.join(raiz, directorio))
    print("-" * 30)
```

- Salida:

```
Raíz: data

Archivos:
data/screenshot_3.png
data/example.png
data/screenshot_6.png
data/archivo.txt
data/indices_secuencia.png
data/indices_slicing.png
data/files.png
```

```
data/screenshot_4.png
data/archivo_de_texto.jpg
data/screenshot_2.png
data/screenshot_1.png
data/screenshot_5.png

Directorios:
data/gato
------------------------------
Raíz: data/gato

Archivos:
data/gato/juego_1.txt
data/gato/juego_2.txt

Directorios:
------------------------------
```

### Ejemplo de lectura escritura básico

- No hay mhco que decir

```
ruta_juego_1 = os.path.join("data", "gato", "juego_1.txt")
archivo = open(ruta_juego_1, "rt")
archivo.readlines()
archivo.close()
```

- Usando with (permite no usar .close()):

```
ruta_juego_1 = os.path.join("data", "gato", "juego_1.txt")
with open(ruta_juego_1, "rt") as archivo:
    lineas = archivo.readlines()
```

## Strings

pendiente, no muy importante

## Annotations

- Permite añadir como comentario los tipos de datos!

### Anotaciones simples

- ej:

```
variable_num = 32 #normal
variable_num: int = 32 #anotado

varrr: str = "hola"
```

- En funciones:

```
def llamar(nombre: str, numero: int):
  pass
```

- En funciones especificando retorno:

```
def contar_vocales(texto: str) -> int:
  return 400
```

## Módulo typing

pendiente

## otros

pendiente