# Clase4

April 6, 2021

#### 0.0.1 Seminario de Lenguajes - Python

### 0.1 Cursada 2021

#### 0.1.1 Clase 4

## 1 Les dejé algo para investigar en el video del fin de semana....

### 1.1 ¿Cuándo un módulo se denomina ___main___?

## 2 Recordemos con este ejemplo cuál es la situación:

```
#módulo funciones
def uno():
    print("uno")
    print(f"El nombre de este módulo es {__name__}")
#uno()

#uso_funciones
import funciones
funciones.uno()
```

## 3 El módulo ___main___

- Las instrucciones ejecutadas en el nivel de llamadas superior del intérprete, ya sea desde un script o interactivamente, se consideran parte del módulo llamado **___main___**, por lo tanto tienen su propio espacio de nombres global.

```
#módulo funciones
def uno():
    print("uno")
    print(f"El nombre de este módulo es {__name__}")

if __name__ == "__main__":
    uno()
```

## 4 Veamos este otro ejemplo:

```
# modulo utiles
def vocales(cadena):
```

```
    print(list(filter(lambda l: l.lower() in "aeiou", cadena)))

# modulo uso_utiles
import utiles

utiles.vocales("Holaaaa!!!!!!")
```

- Primero: ¿qué hace?

#### 4.0.1 ¿Y si queremos invocar el módulo utiles (e invocar a la función vocales) desde la línea de comandos?¿Cómo les pasamos la cadena a analizar?

[ ]:
```
import sys
print(type(sys.argv))
```

- ¿De qué tipo es argv?
- ¿Qué valores contiene?

```
# modulo utiles
def vocales(cadena):
    print(list(filter(lambda l: l.lower() in "aeiou", cadena)))

if __name__ == "__main__":
    import sys
    vocales(sys.argv[1])
```

## 5 Set de juegos

- En el libro "Invent Your Own Computer Games with Python" se presentan varios juegos.
- Tomamos el tateti, el ahorcado y el reverse y los queremos integrar en una única aplicación: juegos.py.
- ¿Es tan sencillo como importar los módulos?
  - Pista: estos programas no fueron pensados para ser importados y ejecutados en forma explícita desde otro programa.
- El que se anime a hacerlo, pueden subirlo a su repositorio y lo comparte con @clauBanchoff
  - Recuerden mandarme mensaje para revisar.

## 6 Paquetes

- Veamos el ejemplo de la documentación oficial de paquetes

```
import sound.effects.echo
from sound.effects import echo
```

#### 6.0.1 ¿Qué contiene el archivo ___init___.py?

## 7 ¿Qué pasa si tenemos la siguiente sentencia?

```
from sound import *
```

- **___all___: es una variable que contiene una lista con los nombres de los módulos que deberían poder importarse cuando se encuentra la sentencia** from package import ***.

```
#Por ejemplo, en sound/effects/__init__.py

__all__ = ["echo", "surround", "reverse"]
```

- Si **___all___** no está definida, **from sound.effects import *** no importa los submódulos dentro del paquete **sound.effects** al espacio de nombres.
- Un artículo para leer luego.

## 8 Analicemos esta librería

- console-menu

### 8.1 Pueden armar el menú de juegos con esta librería.. :)

## 9 Pensemos en las siguientes situaciones

¿Qué estructura usamos si queremos:

- guardar los puntajes cada vez que jugamos a un juego determinado?,
- tener un banco de preguntas para que cada vez que juguemos al juego de repaso las pueda acotar por temas?,
- manipular los Python Plus de los estudiantes por turnos?.

**¿Qué tienen todas estas situaciones en común?**

#### 9.0.1 Necesitamos una estructura que permita que los datos puedan persistir cuando la ejecución del programa finalice.

## 10 Algunas consideraciones antes de empezar

- Lo básico: ¿qué es un **archivo**?
- ¿Cómo podemos manipular los archivos desde un programa Python?

## 11 Manejo de archivos

- Existen funciones predefinidas.
- Si las operaciones fallan, se levanta una **excepción**.
- Los archivos se manejan como objetos que se crean usando la función open.

- **Tarea para el hogar**. Investigar: ¿qué diferencias hay entre un archivo de texto y uno binario?

## 12 Veamos este ejemplo

[ ]:
```
archi1 = open('archivo.txt', 'w')
```

- ¿De qué modo se abre este archivo? ¿Qué significa?
- Luego de la instrucción, ¿dónde se encuentra archivo.txt?
- ¿Cuándo puede dar un error esta sentencia?

## 13 ¿Y este otro ejemplo?

[ ]:
```
archi2 = open('archivo.txt', 'x')
```

- Y en este caso, ¿de qué modo se abre este archivo?
- ¿Cuándo puede dar un error esta sentencia?

## 14 ¿Y en este caso?

[ ]:
```
archi3 = open('archivo.txt')
```

- En realidad la función open tiene más argumentos:

```
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True,
```

- **encoding**: sólo para modo texto. Por defecto, la codificación establecida en las configuraciones del sistema
- **errors**: sólo en modo texto. Es una cadena que dice qué hacer ante un error en la codificación/decodificación. ("strict", "ignore", ..)
- **newline**: sólo modo texto. Puede ser: None, '','\n', '\r', y '\r\n'.

```
archi = open("pp.xxx", "r+", encoding="UTF-8")
```

## 15 ¡Hacemos la actividad 1 por puntos!