Mentorías

Aplicando Herencia y Polimorfismo de Clases

Ver script `lista-productos-autos.py` de Clase 03

Curso de Python

Clase 04

¡Bienvenidxs!

Contenido

- Errores y excepciones
  - Parámetros en la línea de comandos
- El módulo Click
  - Archivos dinámicos en HTML
- El módulo JSON
  - Archivos en JSON
- El módulo Bottle
  - Creando una micro API JSON

Pre-work

Platzi: Curso de Python

3. **Uso de objetos y módulos**
    - Manejo de errores y jerarquía de errores en Python
    - Introducción a Click

Referencias externas:

1. Sitio principal del módulo Click
2. Módulo JSON
3. Micro framework Bottle

CARPETAS DE TRABAJO SUGERIDA

Antes de iniciar a realizar programas, se recomienda crear una estructura de carpetas para esta clase de la siguiente manera:

```
Curso-de-Python
    └── Clase-04
        └── mi-programa.py
```

Documentación oficial de Python

Palabras reservadas y la librería estándar

Documentación oficial:

https://docs.python.org/3

Librería estándar:

https://docs.python.org/3/library/

Funciones interconstruidas:

https://docs.python.org/3/library/functions.html

Errores y excepciones

Excepciones

Cuando ocurre un error en Python, el flujo normal de ejecución se detiene y se activa o se “lanza” una excepción indicando la causa del error.

Para controlar los errores y evitar que el script se detenga abruptamente se hace uso de las instrucciones **try-except**.

MANEJO DE ERRORES / EXCEPCIONES

```
# Cuando un error ocurre, Python lanza una excepción
In [1]: int(“10.5”)
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-2-54dd49a25c21> in <module>()
----> 1 int("10.5")

ValueError: invalid literal for int() with base 10: '10.5'

# El mismo error pero haciendo uso de try-except
In [2]: try:
   ...:     int("10.5")
   ...: except ValueError:
   ...:     print("10.5 no es un entero")
   ...:
10.5 no es un entero
```

Parámetros en la línea de comandos

Ejemplo 01

Son los diferentes argumentos (valores) u opciones que el usuario puede proporcionar o no, cuando ejecuta un comando o un script en una terminal o consola de comandos, permitiendo modificar el resultado del script desde su inicio, de tal forma que el script no se interrumpe hasta que termina o hasta que ocurre algún error.

EJEMPLO-01: PARÁMETROS EN LA LÍNEA DE COMANDOS

Nombre del script: `parametros.py <parámetros...>`

Descripción:
El script deberá de leer uno o más parámetros e imprimirlos en la salida estándar haciendo uso del módulo sys.

Ejemplo de ejecución:

```
Clase-04 $ python parametros.py
['parametros.py']

Clase-04 $ python parametros.py uno dos.txt 3 4 Come frutas
['parametros.py', 'uno', 'dos.txt', '3', '4', 'Come', 'frutas']

Clase-04 $ python parametros.py uno dos.txt 3 4 "Come frutas"
['parametros.py', 'uno', 'dos.txt', '3', '4', 'Come frutas']

# Usando main(sys.argv[1:])
Clase-04 $ python parametros.py uno dos.txt 3 4 "Come frutas"
['uno', 'dos.txt', '3', '4', 'Come frutas']
```

EJEMPLO-01: PARÁMETROS EN LA LÍNEA DE COMANDOS

Nombre del script: `agregar-persona.py NOMBRE EDAD`

Descripción:
El script deberá agregar una persona al archivo personas.csv en formato CSV haciendo uso del manejo de excepciones.

Ejemplo de ejecución:

```
Clase-04 $ python agregar-persona.py HUGO 10
La persona HUGO con edad 10 años ha sido agregada

Clase-04 $ python agregar-persona.py PACO 15
La persona PACO con edad 15 años ha sido agregada

Clase-04 $ cat personas.csv  # type personas.csv
HUGO,10
PACO,15

Clase-04 $
```

ARCHIVOS DE CUALQUIER CARPETA

Nombre del script: `lista-de-archivos.py <DIRECTORIO>`

Descripción:
Imprime la lista de archivos del DIRECTORIO proporcionado, si DIRECTORIO no se indica se muestran los archivos del directorio actual. Si el directorio indicado no existe se deberá mostrar un mensaje, pero evitar en todo momento que el script termine en error lanzando alguna excepción.

Ejemplo de ejecución:

```
Clase-04 $ python lista-de-archivos.py
soluciones                                4096 Feb 19 2019 18:24
README.md                                  249 Feb 19 2019 18:25

Clase-04 $ python lista-de-archivos.py ...
Error: el directorio "..." no existe!

Clase-04 $ python lista-de-archivos.py ..
Clase-02                                  4096 Feb 06 2019 09:17
README.md                                  465 Feb 06 2019 09:17

Clase-04 $ python lista-de-archivos.py /
initrd.img                            43733664 Jul 25 2018 08:49
vmlinuz                                7407392 Nov 24 2017 07:24
lost+found                               16384 May 07 2018 03:49
```

ACTIVIDAD

Crea el script `agrega-producto.py` para que agregue un producto desde la línea de comandos con los atributos nombre, cantidad y precio.

TIP: Usar el script `agrega-persona.py` como base

Ejemplo de ejecución:

```
Clase-04/Reto-01 $ python agrega-producto.py "Mesa chica" 3
100.00
Se ha agregado el producto ['Mesa chica', '3', '100.00'] al
archivo productos.csv

Clase-04/Reto-01 $ python agrega-producto.py "Mesa mediana" 2
150.00
Se ha agregado el producto ['Mesa mediana', '2', '150.00'] al
archivo productos.csv

Clase-04/Reto-01 $ python agrega-producto.py "Mesa grande" 1
300.00
Se ha agregado el producto ['Mesa grande', '1', '300.00'] al
archivo productos.csv

Clase-04/Reto-01 $ cat productos.csv
Mesa chica,3,100.00
Mesa mediana,2,150.00
Mesa grande,1,300.00

Clase-04/Reto-01 $
```

Python Package Index (PyPi)

Sitio oficial:

https://pypi.org

El módulo click

Click (Command Line Interface Creation Kit) es un paquete en Python para crear hermosas interfaces para la línea de comandos en una forma simple, rápida, divertida y con el menor código posible. Es altamente personalizable pero ya incluye valores por omisión con sólo importar el módulo.

Sitio oficial:

https://click.palletsprojects.com

El módulo csv

Python ya cuenta con un módulo para tratar con archivos en formato CSV y su documentación se puede consultar en la librería estándar:

https://docs.python.org/3/library/csv.html

INSTALACIÓN DEL MÓDULO CLICK

Cuando se requiere hacer uso de un módulo que no pertenece a la librería estándar se debe contar con el archivo o carpeta del módulo en nuestra carpeta de trabajo, en otro caso será necesario instalarlo haciendo uso del comando **pip** que es el administrador de paquetes de Python.

Como el módulo click no es parte de la librería estándar, por lo que procedemos a su instalación:

```
Clase-04 $ pip install click
Collecting click
  Using cached
https://files.pythonhosted.org/packages/fa/37/45185cb5abbc30d7257104c434fe0b07e5a
195a6847506c074527aa599ec/Click-7.0-py2.py3-none-any.whl
Installing collected packages: click
Successfully installed click-7.0

Clase-04 $
```

Ejemplo-02

Generando archivos CSV

Aplicar los módulos click y csv al script `agregar-persona.py`.

HOLA CLICK

Nombre del script: `hola-click.py [OPCIONES] NOMBRE N`

Descripción:
El script deberá de imprimir N veces el mensaje “Hola NOMBRE!” en la salida estándar haciendo uso del módulo click. Implementar la opción `--help` para que imprima la ayuda de uso del script.

Ejemplo de ejecución:

```
Clase-04 $ python hola-click.py
Usage: hola-click.py [OPTIONS] NOMBRE
Try "14-hola-click.py --help" for help.

Error: Missing argument "NOMBRE".

Clase-04 $ python hola-click.py Click 1.5
. . .
Error: Invalid value for "[N]": 3.5 is not a valid
integer

Clase-04 $ python hola-click.py Click 3
Hola Click!
Hola Click!
Hola Click!
```

```
Clase-04 $ python hola-click.py --help
Usage: 14-hola-click.py [OPTIONS] NOMBRE [N]

  El script deberá de imprimir N veces el mensaje
“Hola NOMBRE!” en la
  salida estándar haciendo uso del módulo click.

Options:
  --help  Show this message and exit.

Clase-04 $
```

ARCHIVOS DE CUALQUIER CARPETA CON CLICK

Nombre del script: `agrega-persona.py [OPCIONES] PERSONA EDAD`

Descripción:
El script deberá agregar una persona al archivo personas.csv. El archivo deberá estar en formato CSV, se deberá de hacer un manejo adecuado de las excepciones y hacer uso de los módulos click y csv.

Ejemplo de ejecución:

```
Clase-04/Ejemplo-02 $ python agrega-persona.py HUGO 10
Se ha agregado la persona (HUGO, 10) al archivo personas.csv

Clase-04/Ejemplo-02 $ python agrega-persona.py PACO 15
Se ha agregado la persona (PACO, 15) al archivo personas.csv

Clase-04/Ejemplo-02 $ cat personas.csv
HUGO,10
PACO,15

Clase-04/Ejemplo-02 $
```

Reto-02

Modifica el script `agrega-producto.py` del Reto-01 para que realice lo mismo pero validando que cantidad sea un entero (int), precio un decimal (float) y que se haga uso de los módulos click y csv.

Ejemplo de ejecución:

```
Clase-04/Reto-02 $ python agrega-producto.py "Mesa chica" 3 100
Se ha agregado el producto (Mesa chica, 3, 100.0) al archivo
productos.csv

Clase-04/Reto-02 $ python agrega-producto.py "Mesa mediana" 2
150
Se ha agregado el producto (Mesa mediana, 2, 150.0) al archivo
productos.csv

Clase-04/Reto-02 $ python agrega-producto.py "Mesa grande" 1
300
Se ha agregado el producto (Mesa grande, 1, 300.0) al archivo
productos.csv

Clase-04/Reto-02 $ cat productos.csv
Mesa chica,3,100.0
Mesa mediana,2,150.0
Mesa grande,1,300.0

Clase-04/Reto-02 $
```

Ejemplo-03

Generando archivos HTML

```
<html>
<head>
    <title>
     Lista de personas
   </title>
</head>
<body>
  <!-- Lista de
personas creada
       desde Python -->
</body>
</html>
```

EJEMPLO-03: DE CSV A HTML

Nombre del script: `csv2html.py [OPCIONES] ARCHIVO`

Descripción:
El script deberá crear un archivo HTML a partir del archivo en formato CSV proporcionado

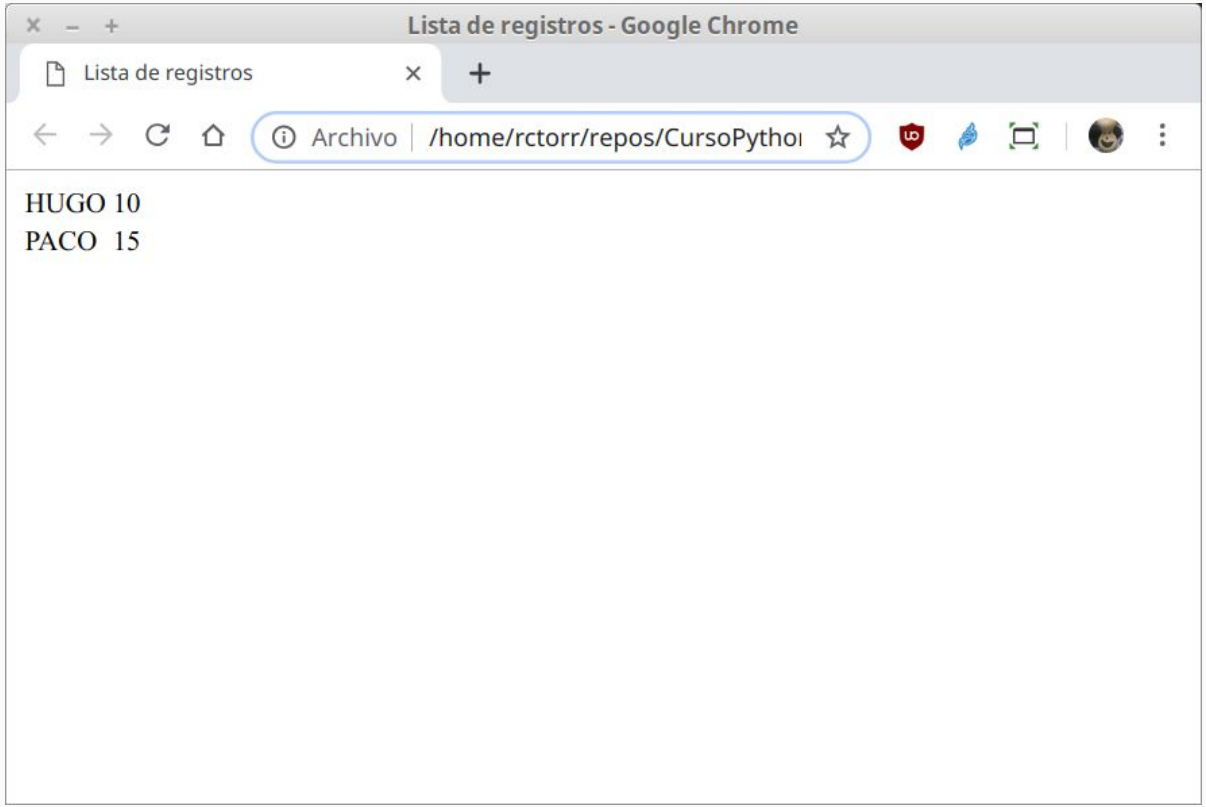
Ejemplo de ejecución:

```
Clase-04/Ejemplo-03 $ python csv2html.py personas.csv

Clase-04/Ejemplo-03 $ cat personas.html
<!DOCTYPE html>
<html>
[...]
    <table>
    <tr><td>HUGO</td><td>10</td></tr>
<tr><td>PACO</td><td>15</td></tr>
    </table>
</body>
</html>
```

EJEMPLO-03: DE CSV A HTML

Resultado en el navegador:



Reto-03

Modifica el script `csv2html.py` del Ejemplo-02 para que realice lo mismo pero con el archivo productos.csv creado en el Reto-02.

Ejemplo de ejecución:

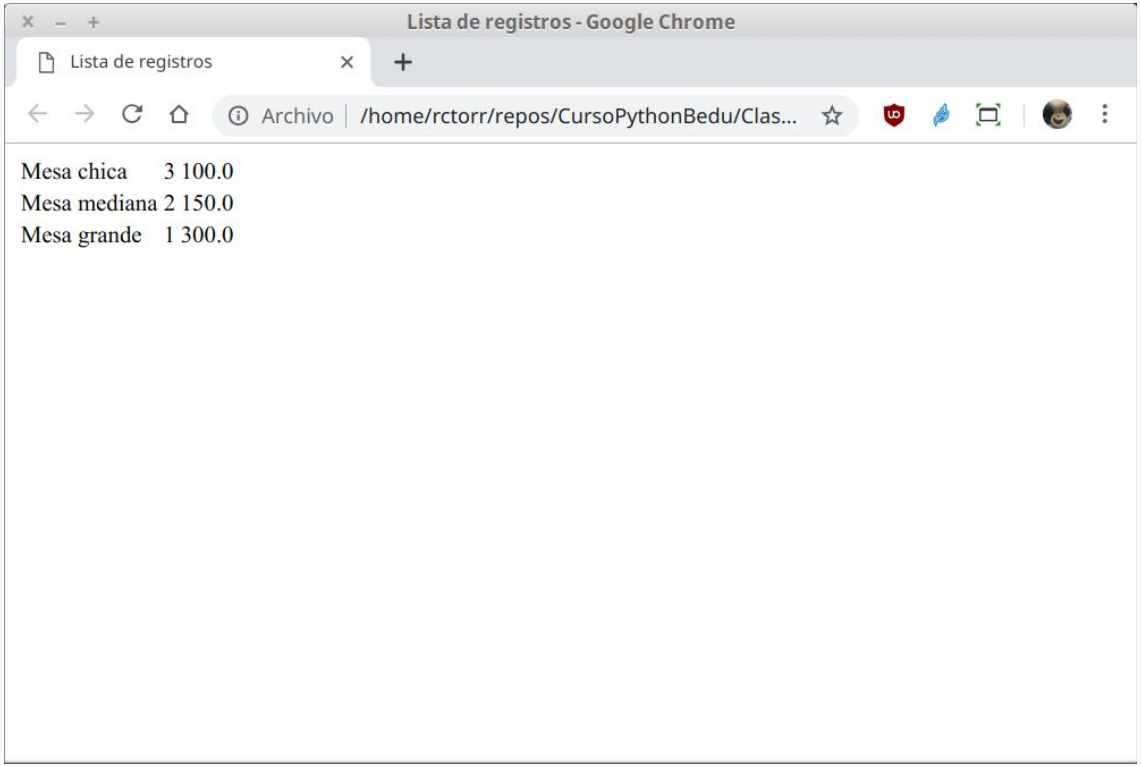
```
Clase-04/Reto-03 $ python csv2html.py productos.csv

Clase-04/Reto-03 $ cat productos.html
<!DOCTYPE html>
<html>
<head>
    <title>Lista de registros</title>
</head>
<body>
    <table>
    <tr><td>Mesa chica</td><td>3</td><td>100.0</td></tr>
<tr><td>Mesa mediana</td><td>2</td><td>150.0</td></tr>
<tr><td>Mesa grande</td><td>1</td><td>300.0</td></tr>
    </table>
</body>
</html>

Clase-04/Reto-03 $
```

LISTA DE PRODUCTOS CON CLICK EN HTML

RESULTADO EN NAVEGADOR:



GALERÍA DESDE UNA CARPETA

RETO:

Crear el script que se indica haciendo uso de los módulos os, click

Abrir el archivo resultante en un navegador.

TIP: Se sugiere basarse en la última versión del script `la.py`

Nombre del script: `galeria.py [OPCIONES] DIRECTORIO`

Descripción:
Imprime en la salida estándar en formato texto la lista de todos los archivos con extensión jpg y png contenidos en DIRECTORIO. Sí DIRECTORIO no usa el directorio actual.

Opciones:

`--help`	Muestra esta ayuda del script
`--html`	Imprime la lista en formato HTML a tres columnas.
`--out ARCHIVO`	Guarda la lista en ARCHIVO en lugar de la salida estándar

EJEMPLO DE VISTA EN NAVEGADOR DEL ARCHIVO HTML CREADO.

El módulo json

JSON (JavaScript Object Notation) es una sintaxis para almacenar e intercambiar datos en formato de texto usando la notación de objetos de Javascript.

El módulo json proporciona una API para trabajar con datos en formato JSON.

ENTENDIENDO EL MÓDULO JSON

```
# Importando el módulo json
In [1]: import json

# De objetos de python -> objetos json
In [2]: oj = json.dumps({“nombre”:arch1.txt, “tamanio”: 1234, “cpadre”: None,
“es_carpeta”: False})
In [3]: oj
Out[3]: '{"nombre": "arch1.txt", "tamanio": 1234, "cpadre": null, "es_carpeta":
false}'

# De objetos de json -> objetos python
In [4]: op = json.loads('{"nombre": "arch1.txt", "tamanio": 1234, "cpadre":
null, "es_carpeta": false}')
In [5]: op
Out[5]: {'nombre': 'arch1.txt', 'tamanio': 1234, 'cpadre': None, 'es_carpeta':
False}
```

ENTENDIENDO EL MÓDULO JSON

```
# De objetos de python -> archivo json
In [6]: with open("ejemplo.json", "w") as fjson:
   ...:     json.dump(op, fjson)
In [7]: !cat ejemplo.json  # usar !type en windows
{"nombre": "arch1.txt", "tamanio": 1234, "cpadre": null, "es_carpeta": false}

# De archivo json -> objetos python
In [8]: with open("ejemplo.json") as fjson:
   ...:     opej = json.load(fjson)
In [9]: opej
Out[9]: {'nombre': 'arch1.txt', 'tamanio': 1234, 'cpadre': None, 'es_carpeta': False}

# Creando archivos json para humanos
In [6]: with open("ejemplo.json", "w") as fjson:
   ...:     json.dump(op, fjson, indent=4, sort_keys=True)
In [7]: !cat ejemplo.json  # usar !type en windows
{
    "cpadre": null,
    "es_carpeta": false,
    "nombre": "arch1.txt",
    "tamanio": 1234
}
```

CORRESPONDENCIA DE TIPOS

Objetos Python serializables	JSON
dict	object
list, tuple	array
str	string
int, float, int- & float-derived Enums	number
True	true
False	false
None	null

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

Reto-04

Crear script `csv2json.py` a partir del script csv2html.py para que convertir archivos csv en formato json.

Ejemplo de ejecución:

```
Clase-04/Reto-04 $ python csv2json.py productos.csv

Clase-04/Reto-04 $ cat productos.json
[
    {
        "nombre": "Caja chica",
        "cantidad": 3,
        "precio": 100.0,
        "subtotal": 300.0
    },
. . .
    {
        "nombre": "Caja grande",
        "cantidad": 1,
        "precio": 199.99,
        "subtotal": 199.99
    }
]

Clase-04/Reto-04 $
```