Kata 10 – Manejo de errores

Uso de tracebacks para buscar errores

Intento de abrir un archivo inexistente desde el intérprete:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ python
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> open("/path/to/mars.jpg")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
>>>
```

Creación del archivo **open.py** que contiene a la **función main()**, que a su vez incluye la instrucción de abrir el archivo con la ruta especificada. Al querer ejecutar el archivo, nos marca una excepción del tipo **FileNotFoundError**:

```
open.py
open.py > ...
1  def main():
2      open("/path/to/mars.jpg")
3
4  if __name__ == '__main__':
5      main()

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ touch open.py
ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ python open.py
Traceback (most recent call last):
  File "open.py", line 5, in <module>
    main()
  File "open.py", line 2, in main
    open("/path/to/mars.jpg")
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$
```

Controlando las excepciones

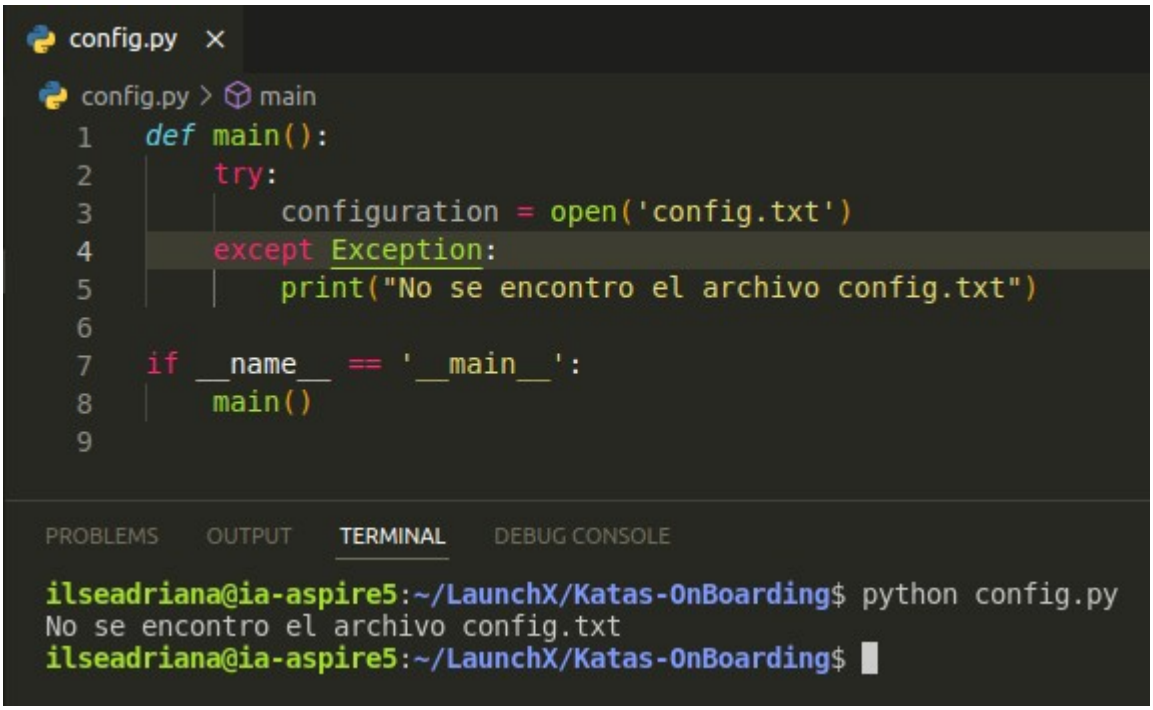
Prueba de un bloque **try-except** desde el intérprete:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ python
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> try:
...     open("config.txt")
... except FileNotFoundError:
...     print("No se encontro el archivo config.txt")
...
No se encontro el archivo config.txt
>>>
```

Creación del archivo **config.py** que contiene nuevamente la **función main()**, la que en esta ocasión incluye un bloque **try-except** a manejar la excepción del tipo **FileNotFoundError**. Al querer ejecutar el archivo, nos marca una excepción del tipo **IsADirectoryError:**

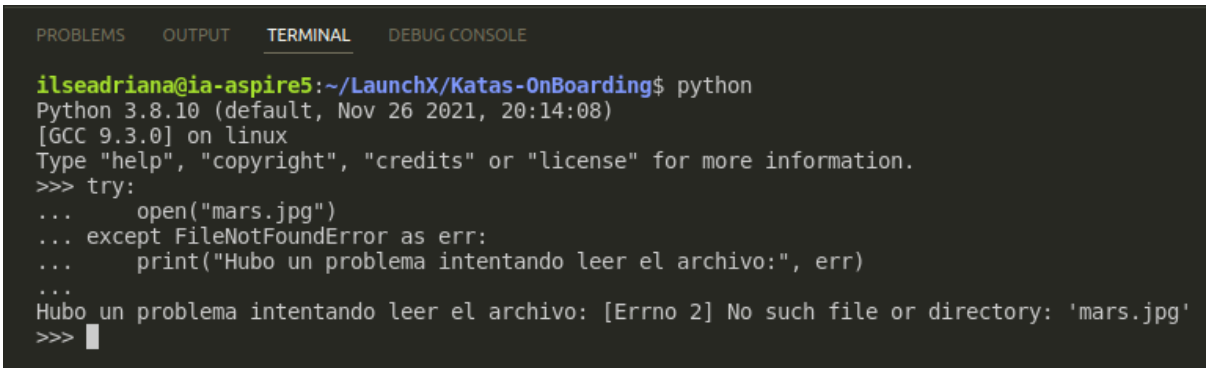
```
config.py
config.py > ...
1   def main():
2       try:
3           configuration = open('config.txt')
4       except FileNotFoundError:
5           print("No se encontro el archivo config.txt")
6
7   if __name__ == '__main__':
8       main()
9

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ touch config.py
ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ mkdir config.txt
ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ python config.py
Traceback (most recent call last):
  File "config.py", line 8, in <module>
    main()
  File "config.py", line 3, in main
    configuration = open('config.txt')
IsADirectoryError: [Errno 21] Is a directory: 'config.txt'
ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$
```

Actualización de la **función main()**, para que en esta ocasión se maneje la **excepción general** (**Exception)**. Al querer ejecutar el archivo, nos imprime el texto indicado en la cláusula except:

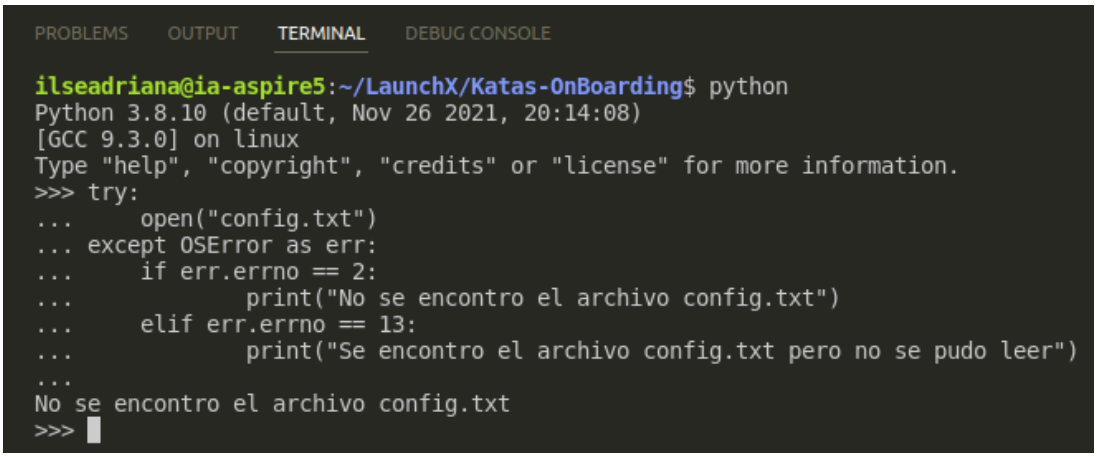


Actualización de la **función main()**, para que en esta ocasión se manejen las excepciones de tipo **FileNotFoundError** e **IsADirectoryError** Al querer ejecutar el archivo nos imprime el texto indicado en la cláusula except:

Prueba de bloque **try-except** incluyendo una variable asignada a través de la palabra reservada **as**, desde el intérprete:



Prueba de bloque **try-except** incluyendo una variable asignada a través de la palabra reservada **as**, dicha variable permitirá acceder directamente a los atributos del error por medio de excepciones de tipos específicos.



Generación de excepciones

A partir de esta sección, todo se realizará a través del intérprete de python desde la terminal.

Creación de la función **agua_restante()**, la cual calcula, con base en el número de astronautas, la cantidad de agua que quedará después de un día o más. Al ejecutar la función, vemos que se ejecuta correctamente; sin embargo, el resultado no es muy útil:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

ilseadriana@ia-aspire5:~/LaunchX/Katas-OnBoarding$ python
Python 3.8.10 (default, Nov 26 2021, 20:14:08)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> def agua_restante(astronautas, agua_restante, dias_restantes):
...     uso_diario = astronautas * 11
...     uso_total = uso_diario * dias_restantes
...     total_agua_restante = agua_restante - uso_total
...     return f"Total de agua restante despues de {dias_restantes} dias: {total_agua_restante} litros"
...
>>> agua_restante(5, 100, 2)
'Total de agua restante despues de 2 dias: -10 litros'
>>>
```

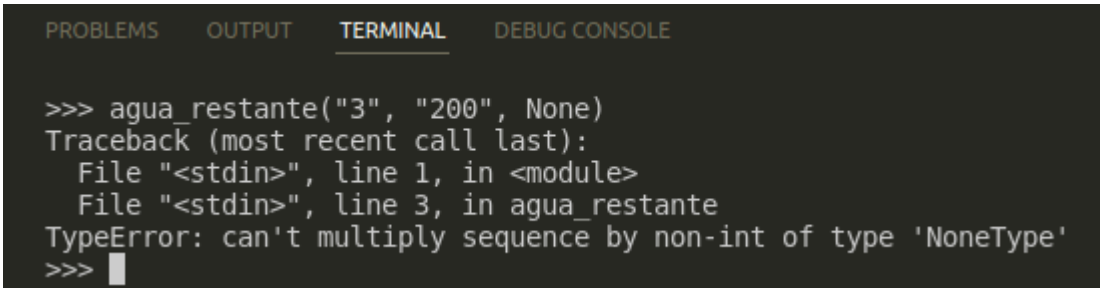
Actualización de la función **agua_restante()**, de tal modo que ahora genere una **excepción** para alertar de la condición de error:

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

>>> def agua_restante(astronautas, agua_restante, dias_restantes):
...     uso_diario = astronautas * 11
...     uso_total = uso_diario * dias_restantes
...     total_agua_restante = agua_restante - uso_total
...     if total_agua_restante < 0:
...         raise RuntimeError(f"No hay suficiente agua para {astronautas} astronautas despues de {dias_restantes} dias")
...     return f"Total de agua restante despues de {dias_restantes} dias: {total_agua_restante} litros"
...
>>> agua_restante(5, 100, 2)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 6, in agua_restante
RuntimeError: No hay suficiente agua para 5 astronautas despues de 2 dias
>>>
```

Creación de la función **alerta_sistema_navegación()**, la cual imprime el texto del error. Al ejecutar el la función dentro de un bloque try-except, obtenemos la alerta indicada para el RuntimeError:

```
>>> def alerta_sistema_navegacion(error):
...     print(error)
...
>>> try:
...     agua_restante(5, 100, 2)
... except RuntimeError as err:
...     alerta_sistema_navegacion(err)
...
No hay suficiente agua para 5 astronautas despues de 2 dias
>>>
```

Intento de pasar argumentos que no sean enteros para comprobar la salida de error:



Actualización de la función **agua_restante**, de tal modo que use la excepción de tipo TypeError, pero con un mensaje mejorado: