

## KATA-MODULO 10

### Tracebacks

```
open("/path/to/mars.jpg")
0.9s Python

-----
FileNotFoundError                                Traceback (most recent call last)
c:\Users\WINDOWS\Desktop\Kata-Modulo 3 - copia.ipynb Cell 1' in <module>
----> 1 open("/path/to/mars.jpg")

FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL** JUPYTER

```
open.py x
1 def main():
2     open("/path/to/mars.jpg")
3
4 if __name__ == '__main__':
5     main()
6
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN **TERMINAL** JUPYTER

(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\WINDOWS>cd Desktop

C:\Users\WINDOWS\Desktop>open("/path/to/mars.jpg")  
"open" no se reconoce como un comando interno o externo,  
programa o archivo por lotes ejecutable.

C:\Users\WINDOWS\Desktop>python open.py  
Traceback (most recent call last):  
 File "C:\Users\WINDOWS\Desktop\open.py", line 5, in <module>  
 main()  
 File "C:\Users\WINDOWS\Desktop\open.py", line 2, in main  
 open("/path/to/mars.jpg")  
FileNotFoundError: [Errno 2] No such file or directory: '/path/to/mars.jpg'

cmd  
cmd

## Controlando las excepciones

### Try y Except de los bloques

```

try:
    open('config.txt')
except FileNotFoundError:
    print("Couldn't find the config.txt file!")

```

... Couldn't find the config.txt file!

```

def main():
    try:
        configuration = open('config.txt')
    except FileNotFoundError:
        print("Couldn't find the config.txt file!")
    except IsADirectoryError:
        print("Found config.txt but it is a directory, couldn't read it")

if __name__ == '__main__':
    main()

```

... Couldn't find the config.txt file!

```

def main():
    try:
        configuration = open('config.txt')
    except FileNotFoundError:
        print("Couldn't find the config.txt file!")

    except Exception:
        print("Couldn't find the config.txt file!")

if __name__ == '__main__':
    main()

```

... Couldn't find the config.txt file!

```

try:
    open("mars.jpg")
except FileNotFoundError as err:
    print("got a problem trying to read the file:", err)

```

... got a problem trying to read the file: [Errno 2] No such file or directory: 'mars.jpg'

```

try:
    open("config.txt")
except OSError as err:
    if err.errno == 2:
        print("Couldn't find the config.txt file!")
    elif err.errno == 13:
        print("Found config.txt but couldn't read it!")

```

... Couldn't find the config.txt file!

## Generación de excepciones

```

def water_left(astronauts, water_left, days_left):
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    return f"Total water left after {days_left} days is: {total_water_left} liters"

```

[16] ✓ 0.6s Python

```

water_left(5, 100, 2)

```

[17] ✓ 0.7s Python

... 'Total water left after 2 days is: -10 liters'

```

def water_left(astronauts, water_left, days_left):
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    if total_water_left < 0:
        raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
    return f"Total water left after {days_left} days is: {total_water_left} liters"

```

[18] ✓ 0.5s Python

```

water_left(5, 100, 2)

```

[19] ✗ 0.8s Python

...

```

-----
RuntimeError                                Traceback (most recent call last)
Input In [19], in <module>
----> 1 water_left(5, 100, 2)

Input In [18], in water_left(astronauts, water_left, days_left)
      4 total_water_left = water_left - total_usage
      5 if total_water_left < 0:
----> 6     raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
      7 return f"Total water left after {days_left} days is: {total_water_left} liters"

RuntimeError: There is not enough water for 5 astronauts after 2 days!

```

```

water_left("3", "200", None)

```

[20] ✗ 0.1s Python

...

```

-----
TypeError                                Traceback (most recent call last)
Input In [20], in <module>
----> 1 water_left("3", "200", None)

Input In [18], in water_left(astronauts, water_left, days_left)
      1 def water_left(astronauts, water_left, days_left):
      2     daily_usage = astronauts * 11
----> 3     total_usage = daily_usage * days_left
      4     total_water_left = water_left - total_usage
      5     if total_water_left < 0:

TypeError: can't multiply sequence by non-int of type 'NoneType'

```

```

def water_left(astronauts, water_left, days_left):
    for argument in [astronauts, water_left, days_left]:
        try:
            # If argument is an int, the following operation will work
            argument / 10
        except TypeError:
            # TypeError will be raised only if it isn't the right type
            # Raise the same exception but with a better error message
            raise TypeError(f"All arguments must be of type int, but received: '{argument}'")
    daily_usage = astronauts * 11
    total_usage = daily_usage * days_left
    total_water_left = water_left - total_usage
    if total_water_left < 0:
        raise RuntimeError(f"There is not enough water for {astronauts} astronauts after {days_left} days!")
    return f"Total water left after {days_left} days is: {total_water_left} liters"

```

[23] ✓ 0.7s Python

```

water_left("3", "200", None)

```

[24] ✗ 0.1s Python

...

```

-----
TypeError                                Traceback (most recent call last)
Input In [23], in water_left(astronauts, water_left, days_left)
      3 try:
      4     # If argument is an int, the following operation will work
----> 5     argument / 10
      6 except TypeError:
      7     # TypeError will be raised only if it isn't the right type
      8     # Raise the same exception but with a better error message

TypeError: unsupported operand type(s) for /: 'str' and 'int'

During handling of the above exception, another exception occurred:

TypeError                                Traceback (most recent call last)
Input In [24], in <module>
----> 1 water_left("3", "200", None)

Input In [23], in water_left(astronauts, water_left, days_left)
      5     argument / 10
      6 except TypeError:
      7     # TypeError will be raised only if it isn't the right type
      8     # Raise the same exception but with a better error message
----> 9     raise TypeError(f"All arguments must be of type int, but received: '{argument}'")
     10 daily_usage = astronauts * 11
     11 total_usage = daily_usage * days_left

TypeError: All arguments must be of type int, but received: '3'

```