EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: DEFINIENDO EXCEPCIONES PERSONALIZADAS
- EXERCISE 2: CREANDO EXCEPCIONES ENCADENADAS PERSONALIZADAS

EXERCISE 1: DEFINIENDO EXCEPCIONES PERSONALIZADAS

Una clase de excepción en Python personalizada se realiza de la misma manera que una normal. Lo que se debe tener en cuenta es incluir la clase base de Python **Exception**, con la finalidad de notificar al compilador que la clase que estás creando es una de excepción.

Probemos este método para crear una clase de excepción llamada **NuestraException,** y usemos la palabra clave de flujo de control de marcador de posición `pass` dentro como marcador de posición.

Excepciones/excepciones-personalizadas.py

```
class NuestraExcepcion(Exception):
    pass

raise NuestraExcepcion
```

Obtenemos la siguiente salida:

Terminal

```
$ python excepciones-personalizadas.py
Traceback (most recent call last):
  File "excepciones-personalizadas.py", line 4, in <module>
    raise NuestraExcepcion
__main__.NuestraExcepcion
```

Para declarar un mensaje de excepción personalizado para **NuestraExcepcion**, se debe anular el método `__init__()` de la clase de excepción, y debe incluir el mensaje que se enviará para la excepción en los parámetros, junto con el parámetro `self`.

Excepciones/excepciones-personalizadas.py

```
class NuestraExcepcion(Exception):
    def __init__(self, message):
        super().__init__(message)

message = "Este es el mensaje de Nuestra Excepción."
raise NuestraExcepcion(message)
```

Obtenemos la siguiente salida:

Terminal

```
$ python excepciones-personalizadas.py
Traceback (most recent call last):
  File "excepciones-personalizadas.py", line 6, in <module>
    raise NustraExcepcion(message)
__main__.NuestraExcepcion: Este es el mensaje de Nuestra Excepción.
```

Ya hemos creado y activado satisfactoriamente una clase de excepción con un mensaje de error personalizado.

Procedemos a utilizar la excepción personalidad en alguna determinada función. Por ejemplo, en la división entera.

Excepciones/excepciones-personalizadas.py

```
1  class NuestraExcepcion(Exception):
2      def __init__(self, message):
3          super().__init__(message)
4
5  message = "Este es el mensaje de Nuestra Excepción."
6
7  Def division_entera(x,y):
8      if y == 0:
9          raise NuestraExcepcion(message)
10     else:
11         dividendo = int(x)
12         divisor = int(y)
13         return dividendo/divisor
14
15 resultado = division_entera(1, 0)
16 print(resultado)
```

Obtenemos la siguiente salida:

Terminal

```
$ 2ivisi excepciones-personalizadas.py
Traceback (most recent call last):
  File "excepciones-personalizadas.py", line 15, in <module>
    resultado = 2ivisión_entera(1, 0)
  File "excepciones-personalizadas.py", line 9, in 2ivisión_entera
    raise NuestraExcepcion(message)
__main__.NuestraExcepcion: Este es el mensaje de Nuestra Excepción.
```

Para situaciones reales que pueden desencadenar una excepción, se resuelven implementando el manejo de excepciones usando el bloque ***try...except***.

Excepciones/excepciones-personalizadas.py

```
class NuestraExcepcion(Exception):
    def __init__(self, message):
        super().__init__(message)

message = "Este es el mensaje de Nuestra Excepción."

def division_entera(x,y):
    if y == 0:
        raise NuestraExcepcion(message)
    else:
        dividendo = int(x)
        divisor = int(y)
        return dividendo/divisor

try:
    resultado = division_entera(1, 0)
    print(resultado)
except NuestraExcepcion:
    print('Error en división por 0')
```

Obtenemos la siguiente salida:

Terminal

```
python excepciones-personalizadas.py
Error en división por 0
```

EXERCISE 2: CREANDO EXCEPCIONES ENCADENADAS PERSONALIZADAS

Vamos a crear otra excepción. Llamemos a la nueva excepción **NumeroFormatoException**, que se activa si la entrada dada no es un número. Para esta clase de excepción, declaremos el mensaje dentro de la clase.

Excepciones/excepciones-personalizadas.py

```
class NuestraExcepcion(Exception):
    def __init__(self, message):
        super().__init__(message)

class NumeroFormatoExcepcion(Exception):
    def __init__(self, message, value):
        message = f'{value} no es un número'
        super().__init__(message)

message = "Se ha generado una excepción."

```

```
12
13 def division_entera(x,y):
14     if type(x) != int:
15         raise NumeroFormatoExcepcion(message, x)
16     elif type(y) != int:
17         raise NumeroFormatoExcepcion(message, y)
18     elif y == 0:
19         raise NuestraExcepcion(message)
20     else:
21         dividendo = int(x)
22         divisor = int(y)
23         return dividendo/divisor
24
25 try:
26     resultado = division_entera(11, 'q')
27     print(resultado)
28 except NuestraExcepcion:
29     print('Error en división por 0')
```

Obtenemos la siguiente salida:

Terminal

```
$ python excepciones-personalizadas.py
Traceback (most recent call last):
  File "excepciones-personalizadas.py", line 26, in <module>
    resultado = division_entera(11, 'q')
  File "excepciones-personalizadas.py", line 17, in division_entera
    raise NumeroFormatoExcepcion(message, y)
__main__.NumeroFormatoExcepcion: q no es un numero
```