

CÓNQUER BLOCKS

PYTHON

EXCEPCIONES

OBJETIVO

- Manejar errores para que nuestros programas no tengan un crash cuando se encuentren con situaciones inesperadas.

Objetos de tipo *excepción*

- Manejar excepciones en distintos casos de uso:

Errores aritméticos
Ausencia de archivos



Añadir solidez y estabilidad a nuestros códigos

ZeroDivisionError

Un numero no puede dividirse por 0

Objeto de excepción

Se crea cuando python no puede realizar aquello que se le pide

```
print(5/0)
```

⊗ 0.1s

ZeroDivisionError

Cell In[1], line 1

----> 1 print(5/0)

ZeroDivisionError: division by zero

ZeroDivisionError

```
print(5/0)
```

⊗ 0.1s

ZeroDivisionError

Cell In[1], line 1

----> 1 print(5/0)

ZeroDivisionError: division by zero

Nuestro objetivo será indicarle a python que hacer cuando ocurra una excepción de este tipo

Uso de bloques try-except

```
try:  
    print(5/0)  
except ZeroDivisionError:  
    print("¡No puedes dividir por cero!")
```

✓ 0.0s

¡No puedes dividir por cero!

Uso de bloques try-except

Si funciona, python ignorará el bloque en el except.

```
try:  
    print(5/0)  
except ZeroDivisionError:  
    print("¡No puedes dividir por cero!")
```

✓ 0.0s

¡No puedes dividir por cero!

Uso de bloques try-except

```
try:  
    print(5/0)  
except ZeroDivisionError:  
    print("¡No puedes dividir por cero!")
```

✓ 0.0s

¡No puedes dividir por cero!

Si no funciona, python buscará un bloque *except* que se corresponda con el error que ha producido

Uso de bloques try-except

```
try:  
    print(5/0)  
except ZeroDivisionError:  
    print("¡No puedes dividir por cero!")
```

✓ 0.0s

¡No puedes dividir por cero!

Si no funciona, python buscará un bloque *except* que se corresponda con el error que ha producido



ZeroDivisionError

Uso de bloques try-except

ZeroDivisionError

Mensaje de error personalizado

+

El código seguirá ejecutándose (no habrá crash) porque le hemos indicado a python como manejar el error.

```
try:
    print(5/0)
except ZeroDivisionError:
    print("¡No puedes dividir por cero!")
```

✓ 0.0s

¡No puedes dividir por cero!

Evitar crashes

```
print("Dame dos numeros y para dividir.")
print("Introduce 's' para salir.")

while True:
    numero1 = input("\nPrimer numero: ")
    if numero1 == 's':
        break
    numero2 = input("Segundo numero: ")
    if numero2 == 's':
        break
    resultado = int(numero1) / int(numero2)
    print(resultado)
```

Evitar crashes

```
print("Dame dos numeros y para dividir.  
print("Introduce 's' para salir.")
```

```
while True:  
    numero1 = input("\nPrimer numero: ")  
    if numero1 == 's':  
        break  
    numero2 = input("Segundo numero: ")  
    if numero2 == 's':  
        break  
    resultado = int(numero1) / int(numero2)  
    print(resultado)
```

```
-----  
ZeroDivisionError                                Traceback (most recent call last)  
Cell In[8], line 11  
  
     10     break  
--> 11 resultado = int(numero1) / int(numero2)  
     12 print(resultado)  
  
ZeroDivisionError: division by zero
```

Evitar crashes

```
print("Dame dos numeros y para dividir.")
print("Introduce 's' para salir.")

while True:
    numero1 = input("\nPrimer numero: ")
    if numero1 == 's':
        break
    numero2 = input("Segundo numero: ")
    if numero2 == 's':
        break
    try:
        resultado = int(numero1) / int(numero2)

    except ZeroDivisionError:
        print("You can't divide by 0!")

    else:
        print(resultado)
```

```
Dame dos numeros y para dividir.
Introduce 's' para salir.
```

```
Primer numero: 3
Segundo numero: 4
0.75
```

```
Primer numero: 5
Segundo numero: 0
You can't divide by 0!
```

```
Primer numero: 6
Segundo numero: 7
0.8571428571428571
```

Evitar crashes

```
print("Dame dos numeros y para dividir.")
print("Introduce 's' para salir.")

while True:
    numero1 = input("\nPrimer numero: ")
    if numero1 == 's':
        break
    numero2 = input("Segundo numero: ")
    if numero2 == 's':
        break
    try:
        resultado = int(numero1) / int(numero2)

    except ZeroDivisionError:
        print("You can't divide by 0!")

    else:
        print(resultado)
```

Dame dos numeros y para dividir.
Introduce 's' para salir.

Primer numero: 3
Segundo numero: 4
0.75

Primer numero: 5
Segundo numero: 0
You can't divide by 0!

Primer numero: 6
Segundo numero: 7
0.8571428571428571

FileNotFound

```
filename = "test.txt"

with open(filename) as f_obj:
    contenido = f_obj.read()
```

FileNotFound

```
with open(filename) as f_obj:  
FileNotFoundError: [Errno 2] No such file or directory: 'test.txt'
```

```
filename = "test.txt"
```

```
with open(filename) as f_obj:  
    contenido = f_obj.read()
```


FileNotFound

```
filename = "test.txt"
try:
    with open(filename) as f_obj:
        contents = f_obj.read()
except FileNotFoundError:
    msj = "Lo siento, el archivo " + filename + " no existe."
    print(msj)
```

FileNotFound

```
filename = "test.txt"
try:
    with open(filename) as f_obj:
        contents = f_obj.read()
except FileNotFoundError:
    msj = "Lo siento, el archivo " + filename + " no existe."
    print(msj)
```

```
(cblocks) MacBook-Pro-5:Clase 8 Elena$ python3 /Users/
vanzado/Clase 8/Ejemplo/FileNotFounds_test.py
Lo siento, el archivo test.txt no existe.
(cblocks) MacBook-Pro-5:Clase 8 Elena$
```

Ejemplo Practico

(<http://gutenberg.org/>)

ANALISIS DE TEXTOS



Contar el numero de palabras en un texto

```
filename = "test.txt"
try:
    with open(filename) as f_obj:
        contenido = f_obj.read()
except FileNotFoundError:
    msj = "Lo siento, el archivo " + filename + " no existe."
    print(msj)

else:
    # Count the approximate number of words in the file.
    palabras = contenido.split()
    num_palabras = len(palabras)
    print("El archivo " + filename + " tiene " + str(num_palabras) + " palabras.")
```

```
(chblocks) MacBook-Pro-5: Ejemplo_Elena$ python3 /Users/Elena/PycharmProjects/ConquerBlocks/Clase 8/Ejemplo/analisis_de_texto.py
El archivo test.txt tiene 125 palabras.
```

Ejemplo Practico

(<http://gutenberg.org/>)

ANALISIS DE TEXTOS



Trabajar con multiples archivos

```
def contar_palabras(filename):  
    """Count the approximate number of words in a file."""  
    try:  
        with open(filename) as f_obj:  
            contenido = f_obj.read()  
    except FileNotFoundError:  
        msj = "Lo siento, el archivo " + filename + " no existe."  
        print(msj)  
    else:  
        # Count approximate number of words in the file.  
        palabras = contenido.split()  
        num_palabras = len(palabras)  
        print("El archivo " + filename + " tiene " + str(num_palabras) +  
              " palabras.")  
  
filenames = ['alice.txt', 'siddhartha.txt', 'moby_dick.txt', 'metamorfosis.txt']  
  
for filename in filenames:  
    contar_palabras(filename)
```

```
vanzado/Clase 8/Ejemplo/analisis_de_texto_2.py"  
El archivo alice.txt tiene 125 palabras.  
El archivo siddhartha.txt tiene 267 palabras.  
Lo siento, el archivo moby_dick.txt no existe.  
El archivo metamorfosis.txt tiene 192 palabras.  
(ch1-block) Mes-Book-Pro-5-Ejemplo-Flora
```


Errores silenciosos

```
def contar_palabras(filename):  
    """Count the approximate number of words in a file."""  
    try:  
        with open(filename) as f_obj:  
            contenido = f_obj.read()  
    except FileNotFoundError:  
        pass  
    else:  
        # Count approximate number of words in the file.  
        palabras = contenido.split()  
        num_palabras = len(palabras)  
        print("El archivo " + filename + " tiene " + str(num_palabras) +  
              " palabras.")  
  
filenames = ['alice.txt', 'siddhartha.txt', 'moby_dick.txt', 'metamorfosis.txt']  
  
for filename in filenames:  
    contar_palabras(filename)
```

Errores silenciosos

```
def contar_palabras(filename):  
    """Count the approximate number of words in a file."""  
    try:  
        with open(filename) as f_obj:  
            contenido = f_obj.read()  
    except FileNotFoundError:  
        pass ←  
    else:  
        # Count approximate number of words in the file.  
        palabras = contenido.split()  
        num_palabras = len(palabras)  
        print("El archivo " + filename + " tiene " + str(num_palabras) +  
              " palabras.")  
  
filenames = ['alice.txt', 'siddhartha.txt', 'moby_dick.txt', 'metamorfosis.txt']  
  
for filename in filenames:  
    contar_palabras(filename)
```

Errores silenciosos

```
def contar_palabras(filename):  
    """Count the approximate number of words in a file."""  
    try:  
        with open(filename) as f_obj:  
            contenido = f_obj.read()  
    except FileNotFoundError:  
        pass  
    else:  
        # Count approximate number of words in the file.  
        palabras = contenido.split()  
        num_palabras = len(palabras)  
        print("El archivo " + filename + " tiene " + str(num_palabras) +  
              " palabras.")  
  
filenames = ['alice.txt', 'siddhartha.txt', 'moby_dick.txt', 'metamorfosis.txt']  
  
for filename in filenames:  
    contar_palabras(filename)
```

```
vanzado/Clase 8/Ejemplo/analisis_de_texto_2.py"  
El archivo alice.txt tiene 125 palabras.  
El archivo siddhartha.txt tiene 267 palabras.  
El archivo metamorfosis.txt tiene 192 palabras.
```


¿Qué errores debemos reportar?

En el caso anterior...

- **Si el usuario sabe que textos deben ser analizados apreciará saber por qué algunos no han podido ser analizados.**
- **Si en cambio el usuario espera resultados pero no sabe que textos deben analizarse puede que no necesite saber que hay algunos que no están disponibles.**

¿Qué errores debemos reportar?

- **Dar al usuario información que no necesita puede ser contraproducente y reducir la utilidad de tu programa.**
- **Un código bien escrito es proclive a muy pocos errores lógicos o de sintaxis**

Posibles focos de error:

Siempre que tu programa dependa de información externa...

**input del usuario
existencia de un archivo
disponibilidad de red de conexión**

...

CÔNQUER BLOCKS