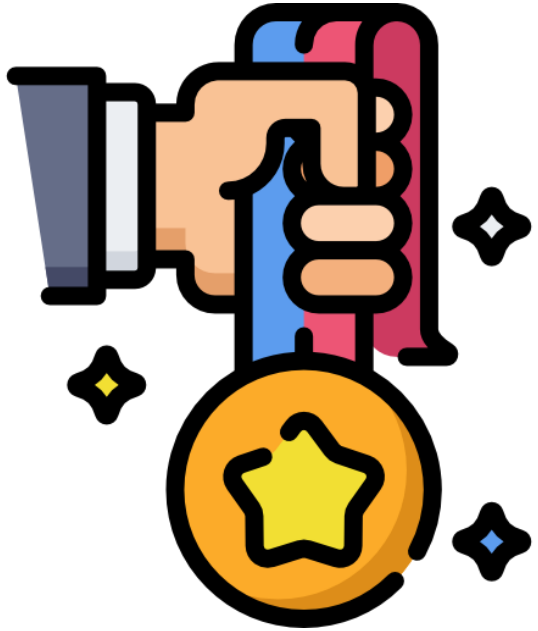
Estructura de Datos y Programación Orientada a Objetos

Excepciones

Semana 2

Logro de aprendizaje

- *Evalúa los errores más comunes y las diferentes excepciones al ejecutar un programa utilizando un lenguaje de programación..*



Contenidos

EXCEPCIONES

Observemos el siguiente vídeo

Tomar nota de las ideas importantes

https://www.youtube.com/watch?v=wldGsJTPJ1o

Los errores en Python

- La razón más común de un error en un programa Python es cuando cierta declaración no está de acuerdo con el uso prescrito. Tal error se llama error de sintaxis. El intérprete de Python lo informa de inmediato, generalmente junto con el motivo.

```
>>> print "hello"
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("hello")?
```

Veamos algunos errores más comunes

IndexError

Se genera al intentar acceder a un elemento en un índice no válido.

```
>>> L1 = [1,2,3]
>>> L1[3]
Traceback (most recent call last):
File "<pyshell#18>", line 1, in <module>
L1[3]
IndexError: list index out of range
```

ModuleNotFoundError

Se genera cuando no se puede encontrar un módulo.

```
>>> import notamodule
Traceback (most recent call last):
File "<pyshell#10>", line 1, in <module>
import notamodule
ModuleNotFoundError: No module named
'notamodule'
```

Veamos algunos errores más comunes

KeyError

Se lanza cuando no se encuentra una clave.

```
>>> D1 = {'1':"aa", '2':"bb", '3':"cc"}
>>> D1['4']
Traceback (most recent call last):
File "<pyshell#15>", line 1, in <module>
D1['4']
KeyError: '4'
```

ImportError

Se produce cuando no se puede encontrar una función específica.

```
>>> from math import cube
Traceback (most recent call last):
File "<pyshell#16>", line 1, in <module>
from math import cube
ImportError: cannot import name 'cube'
```

Veamos algunos errores más comunes

StopIteration

Se lanza cuando la next() función va más allá de los elementos del iterador.

```
>>> it = iter([1,2,3])
>>> next(it)
1
>>> next(it)
2
>>> next(it)
3
>>> next(it)
Traceback (most recent call last):
File "<pyshell#23>", line 1, in <module>
next(it)
StopIteration
```

TypeError

Se produce cuando una operación o función se aplica a un objeto de un tipo inapropiado.

```
>>> '2' + 2
Traceback (most recent call last):
File "<pyshell#23>", line 1, in <module>
'2'+2
TypeError: must be str, not int
```

Veamos algunos errores más comunes

ValueError

Se genera cuando el argumento de una función es de un tipo inapropiado.

```
>>> int('xyz')
Traceback (most recent call last):
File "<pyshell#14>", line 1, in <module>
int('xyz')
ValueError: invalid literal for int()
with base 10: 'xyz'
```

NameError

Se produce cuando no se puede encontrar un objeto.

```
>>> age
Traceback (most recent call last):
File "<pyshell#6>", line 1, in <module>
age
NameError: name 'age' is not defined
```

Veamos algunos errores más comunes

ZeroDivisionError

Se lanza cuando el segundo operador en la división es cero.

```
>>> x = 100 / 0
Traceback (most recent call last):
File "<pyshell#8>", line 1, in <module>
x=100/0
ZeroDivisionError: division by zero
```

KeyboardInterrupt

Se lanza cuando se presiona la tecla de interrupción al ejecutarse un programa.

```
>>> name = input('enter your name')
enter your name (ctrl + c)
Traceback (most recent call last):
File "<pyshell#9>", line 1, in <module>
name=input('enter your name')
KeyboardInterrupt
```

Importantes excepciones integradas en python

Excepción	Descripción
AssertionError	Se genera cuando falla la declaración de afirmación.
AttributeError	Levantado en la asignación de atributo o la referencia falla.
EOFError	Se genera cuando la función input () alcanza la condición de fin de archivo.
FloatingPointError	Se genera cuando falla una operación de coma flotante.
GeneratorExit	Se genera cuando se llama al método close () de un generador.
ImportError	Se genera cuando no se encuentra el módulo importado.
IndexError	Se eleva cuando el índice de una secuencia está fuera de rango.
KeyError	Se genera cuando no se encuentra una clave en un diccionario.
KeyboardInterrupt	Se genera cuando el usuario presiona la tecla de interrupción (Ctrl + c o eliminar).
MemoryError	Se genera cuando una operación se queda sin memoria.

Importantes excepciones integradas en python

Excepción	Descripción
NameError	Se genera cuando no se encuentra una variable en el ámbito local o global.
NotImplementedError	Criado por métodos abstractos.
OSError	Se genera cuando una operación del sistema provoca un error relacionado con el sistema.
OverflowError	Se genera cuando el resultado de una operación aritmética es demasiado grande para ser representado.
ReferenceError	Se genera cuando se usa un proxy de referencia débil para acceder a un referente recolectado como basura.
RuntimeError	Se genera cuando un error no se incluye en ninguna otra categoría.
StopIteration	Levantado por la función next () para indicar que el iterador no devolverá ningún elemento adicional.
SyntaxError	Levantado por el analizador cuando se encuentra un error de sintaxis.
IndentationError	Levantado cuando hay una sangría incorrecta.
TabError	Se genera cuando la sangría consiste en tabulaciones y espacios inconsistentes.

Importantes excepciones integradas en python

Excepción	Descripción
SystemError	Se genera cuando el intérprete detecta un error interno.
SystemExit	Levantado por la función sys.exit ().
TypeError	Se genera cuando una función u operación se aplica a un objeto de un tipo incorrecto.
UnboundLocalError	Se genera cuando se hace referencia a una variable local en una función o método, pero no se ha vinculado ningún valor a esa variable.
UnicodeError	Se genera cuando se produce un error de codificación o decodificación relacionado con Unicode.
UnicodeEncodeError	Se genera cuando se produce un error relacionado con Unicode durante la codificación.
UnicodeDecodeError	Se genera cuando se produce un error relacionado con Unicode durante la decodificación.
UnicodeTranslateError	Se genera cuando se produce un error relacionado con Unicode durante la traducción.
ValueError	Se genera cuando una función obtiene un argumento del tipo correcto pero de valor incorrecto.
ZeroDivisionError	Se genera cuando el segundo operando de una operación de división o módulo es cero.

Manejo de excepciones en Python (try - except)

- La causa de una excepción a menudo es externa al programa en sí. Por ejemplo, una entrada incorrecta, un dispositivo de E/S defectuoso, etc.
- Debido a que el programa termina abruptamente al encontrar una excepción, puede dañar los recursos del sistema, como los archivos.
- Por lo tanto, las excepciones deben manejarse adecuadamente para evitar una terminación abrupta del programa.

Python usa las palabras clave `try` y `except` para manejar excepciones. Ambas palabras clave van seguidas de bloques sangrados.

SINTAXIS:

```
try:
    # declaraciones en el bloque try
except:
    # se ejecuta cuando hay error en el bloque try
```

Veamos algunos ejemplos

Ejemplo 1:

```
try:
    a = 5
    b = '0'
    print(a/b)

except:
    print("Se produjo un error")
```

Ejemplo 2:

```
try:
    a = 5
    b = '0'
    print(a+b)

except TypeError:
    print("Operación no soportada.")
```

Veamos algunos ejemplos

Ejemplo 3:

```
try:
    a = 5
    b = 0
    print(a/b)

except TypeError:
    print("Operación no soportada.")

except ZeroDivisionError:
    print("División por cero no permitida")
```

Ejemplo 4:

```
try:
    a = 5
    b = '0'
    print(a/b)

except TypeError:
    print("Operación no soportada.")

except ZeroDivisionError:
    print("División por cero no permitida")
```

Manejo de excepciones en Python (else - finally)

- En Python, las palabras clave else y finally también se pueden usar junto con las cláusulas try y except. Si bien el bloque except se ejecuta si la excepción ocurre dentro del bloque try, el bloque else se procesa si se encuentra que el bloque try está libre de excepciones.

SINTAXIS:

```
try:
    # declaraciones en el bloque try
except:
    # se ejecuta cuando el error es en
    el bloque try
else:
    # se ejecuta si el bloque try está
    libre de errores
finally:
    # se ejecutado independientemente de la
    excepción ocurrida o no
```

Veamos algunos ejemplos

Ejemplo 5:

```
try:
    print("Bloque try")
    x = int(input("Ingresa un número: "))
    y = int(input("Ingresa otro número: "))
    z = x/y

except ZeroDivisionError:
    print("Bloque except ZeroDivisionError")
    print("División por cero no aceptada.")

else:
    print("Bloque else")
    print("División = ", z)

finally:
    print("Bloque finally")
```

Aplicación:

ACTIVIDAD 1:

Consigna: Trabajo Individual: Resolución de ejercicios

- Cada alumno deberá realizar un programa que tenga 1 función que permita dividir dos números enteros, a su vez deberá pedir dichos números enteros, el programa debe tener manejo de excepciones.

Recursos: Pc o Laptop, Python, Visual Studio Code

Tiempo: 15 minutos

Aplicación:

ACTIVIDAD 2:

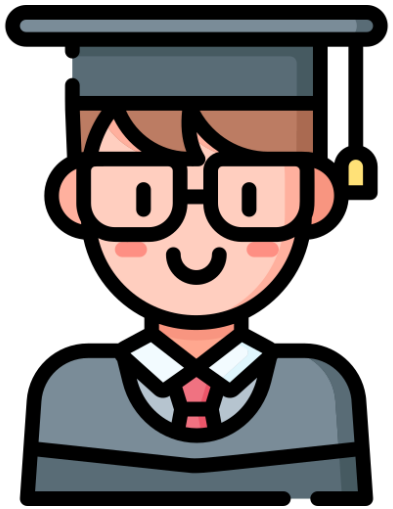
Consigna: Trabajo Individual: Resolución de ejercicios

- *Cada alumno deberá realizar un programa que tenga 4 funciones que permita sumar, restar, multiplicar y dividir, a su vez deberá pedir dos números enteros, además de ingresar la operación que desea realizar, usando condicionales deberá aplicar una de las operaciones, el programa debe tener manejo de excepciones.*

Recursos: Pc o Laptop, Python, Visual Studio Code

Tiempo: 20 minutos

Término



1. ¿Qué aprendimos hoy?

2. ¿Por qué el tema tratado es importante en mi formación como programador?

Muchas Gracias
por su atención

¿Alguna pregunta?

¿No?
Excelente