Clase7_Excepciones

April 25, 2022

1 Seminario de Lenguajes - Python

1.1 Cursada 2022

1.1.1 Clase 7: manejo de excepciones

2 ¿Excepciones?

2.1 Desafío 1: analicemos este código

¿Dónde se puede producir una excepción? ¿Cuál o cuáles?

[ ]:
```
mi_musica = {70: ["Stairway to heaven", "Bohemian Rhapsody"],
             80: ["Dancing in the dark", "Welcome to the jungle", "Under␣
 ↪pressure"],
             2000:["Given up", "The pretender"]}

tema = input("Ingresá un nuevo tema (FIN para terminar): ")
while tema !="FIN":
    decada = int(input("ingresá a qué década pertenece: "))
    mi_musica[decada].append(tema)

    tema = input("Ingresá un nuevo tema (FIN para terminar): ")
```

3 Recordemos: ¿qué es un excepción?

Una **excepción** es un acontecimiento, que **ocurre durante la ejecución** de un programa, que **interrumpe** el **flujo normal** de las instrucciones del programa.

4 Excepciones en Python

Habíamos visto la siguiente estructura:

```
try:
    sentencias
except nombreExcepción:
    sentencias
except nombreExcepción:
```

```
    sentencias
except:
```

- +Info

5 ¿Cómo incluímos el manejo de excepciones en nuestro código?

5.0.1 Analicemos esta solución al desafío:

[ ]:
```
mi_musica = {70: ["Stairway to heaven", "Bohemian Rhapsody"],
             80: ["Dancing in the dark", "Welcome to the jungle", "Under␣
 ↪pressure"],
             2000:["Given up", "The pretender"]}

tema = input("Ingresá un nuevo tema (FIN para terminar): ")
while tema !="FIN":
    try:
        decada = int(input("ingresá a qué década pertenece: "))
        mi_musica[decada].append(tema)
    except ValueError:
        print("Para ingresar la decada, tenés que ingresar un número. Empecemos␣
 ↪de nuevo...")
    except KeyError:
        print("""Por ahora, sólo tengo registradas las décadas: 70, 80 y 2000.␣
 ↪Ingresá una de ellas.
                Empecemos de nuevo...""")

    tema = input("Ingresá un nuevo tema (FIN para terminar): ")
```

6 Podríamos haber manejado de ambas excepciones juntas:

[ ]:
```
mi_musica = {70: ["Stairway to heaven", "Bohemian Rhapsody"],
             80: ["Dancing in the dark", "Welcome to the jungle", "Under␣
 ↪pressure"],
             2000:["Given up", "The pretender"]}
tema = input("Ingresá un nuevo tema (FIN para terminar): ")
while tema !="FIN":
    try:
        decada = int(input("ingresá a qué década pertenece: "))
        mi_musica[decada].append(tema)
    except (ValueError, KeyError):
        print("Hubo un error en el ingreso de datos. Intentá de nuevo")
    except:
        print("Ups! Algo ocurrió")
    tema = input("Ingresá un nuevo tema (FIN para terminar): ")
```

- ¿Cuándo se imprimiría el mensaje: “Ups! Algo ocurrió”?

7 Recordemos lo planteado la clase pasada: ¿qué debemos investigar para trabajar con excepciones?

- ¿Qué acción se toma después de levantada y manejada una excepción? ¿Se continúa con la ejecución de la unidad que lo provocó o se termina?
- ¿Cómo se alcanza una excepción?
- ¿Cómo especificar los manejadores de excepciones que se deben ejecutar cuando se alcanzan las mismas?
- ¿Qué sucede cuando no se encuentra un manejador para una excepción levantada?
- ¿El lenguaje tiene excepciones predefinidas?
- ¿Podemos levantar en forma explícita una excepción?
- ¿Podemos crear nuestras propias excepciones?

8 Ya respondimos algunas:

- **¿Qué acción se toma después de levantada y manejada una excepción? ¿Se continúa con la ejecución de la unidad que lo provocó o se termina?**
- **¿Cómo se alcanza una excepción?**
- **¿Cómo especificar los manejadores de excepciones que se deben ejecutar cuando se alcanzan las mismas?**
- ¿Qué sucede cuando no se encuentra un manejador para una excepción levantada?
- **¿El lenguaje tiene excepciones predefinidas?**
- ¿Podemos levantar en forma explícita una excepción?
- ¿Podemos crear nuestras propias excepciones?

9 Se puede agregar algo más...

La sentencia completa:

```
try:
    sentencias
except excepcion1, excepcion2:
    sentencias
except excepcion3 as variable:
    sentencias
except:
    sentencias
else:
    sentencias
finally:
    sentencias
```

9.0.1 Teníamos una tarea....

10 Veamos este ejemplo sencillo

```
[ ]: XX = 10
try:
    print(XX)
except NameError:
    print("Usaste una variable que no está definida")
else:
    print("Este mensaje se imprime porque  NO se levantó la excepción")
finally:
    print("Este mensaje se imprime SIEMPRE")
```

10.0.1 Entonces, ¿para qué usamos else y finally?

11 Repasemos todo con este otro ejemplo:

```
[ ]: dicci = {0:"Led Zeppelin", 2:"Deep Purple", 3:"Black Sabbath"}
y = 9
try:
    print('Vamos a entrar a otro bloque TRY')
    try:
        for x in range(1,6):
            print (dicci[z])      # OJO que estamos usando la variable z
    except KeyError:
        dicci[x] = 'Agregado'

    y = y + 1
    print(f"El valor de y es {y}")
except NameError:
    print('OJO! Se está usando una variable que no existe')

print('Se sigue con las siguientes sentencias del programa')
```

- ¿Se ejecutaron las líneas 11 y 12?

11.0.1 El ejemplo demuestra que Python aplica el mecanismo de TERMINACIÓN.

12 Observemos el bloque finally en este otro ejemplo:

```
[ ]: dicci = {0:"Led Zeppelin", 2:"Deep Purple", 3:"Black Sabbath"}
y = 9
try:
    print('Vamos a entrar a otro bloque TRY')
    try:
        for x in range(1,6):
```

```
            print (dicci[z])      # OJO que estamos usando la variable z
    except KeyError:
        dicci[x] = 'Agregado'
    finally:
        y = y + 1
        print(f"El valor de y es {y}")
    print("Este texto no se imprime nunca!!!")
except NameError:
    print('OJO! Se está usando una variable que no existe')

print('Se sigue con las siguientes sentencias del programa')
```

- ¿Se ejecutaron las líneas 11 y 12?

12.1 Y.. ¿qué sucede si no encuentra un manejador en esa primera búsqueda?

13 Observemos el siguiente ejemplo:

[ ]:
```
def retornar_elemento(x):
    dicci = {0:"Led Zeppelin", 2:"Deep Purple", 3:"Black Sabbath"}
    try:
        return dicci[x]
    except NameError:
        x = 0

print('Este código sirve para mostrar propagación dinámica.')
elem = int(input('Ingresá una clave para acceder al diccionario: (999 para␣
 ↪finalizar) '))
while elem!=999:
    try:
        print (f"El valor del elemento: {elem} es {retornar_elemento(elem)}")
    except KeyError:
        print ('Ups! Entraste una clave inexistente. Probá de nuevo!')
    elem = int(input('Ingresá clave para acceder al diccionario: (999 para␣
 ↪finalizar) '))
```

- ¿Qué excepción se produce?¿Dónde se levanta? ¿Cuál es el bloque que termina?
- ¿Qué podemos decir sobre la forma de buscar el manejador de la excepción?
  - Cuando **no se encuentra** un manejador para la excepción levantada, ¿dónde busca?

14 ¿Qué sucedió?

- La excepción KeyError se levantó dentro de la función **retornar_elemento**.
- **Busca estáticamente** si el bloque está encerrado en otro bloque try except.
- Al no encontrar un manejador para esa excepción en la función …
- **Busca dinámicamente** a quién llamó a la función.
- Si no encuentra un manejador… entonces termina el programa… con error.

15 ¿Cómo es la forma de propagación que utiliza Python?

- Primero busca **estáticamente**.
- Si no se encuentra, busca **dinámicamente** a quién llamó a la función.
- Si no encuentra un manejador… entonces termina el programa … con error..

16 Podemos levantar explícitamente excepciones

[ ]:
```
dicci = {0:"Led Zeppelin", 1:"Deep Purple", 4:"Black Sabbath"}

try:
    print ('Entramos al bloque try')
    for x in range(1,6):
        if x == 2 or x == 3:
            raise KeyError
        else:
            print (dicci[x])
    print('Continuamos con el proceso..')
except KeyError:
    dicci[x] = 'NUEVO'

dicci
```

¿Por qué no continúa con la iteración e ingresa un elemento con clave 3?

17 También es posible:

[ ]:
```
try:
    print ('Entramos al bloque try')
    for x in range(1,6):
        if x == 2 or x == 3:
            raise
        else:
            print (dicci[x])
    print('Continuamos con el proceso..')
except KeyError:
    print("Manejando KeyError")
```

- ¿Qué excepción se levantó?
- **raise**: vuelve a lanzar la última excepción que estaba activa en el ámbito actual. Si no hay ninguna excepción activa en el alcance actual, se lanza una **RuntimeError** que indica que se trata de un error.

[ ]:
```
dicci = {0:"Led Zeppelin", 2:"Deep Purple", 3:"Black Sabbath"}
try:
    print("Ingresamos al bloque try externo...")
```

```
    try:
        print ("Entramos al bloque try interno")
        for x in range(1,6):
            if x == 2 or x == 3:
                raise KeyError
            else:
                print (dicci[x])
        print("Continuamos con el proceso..")
    except KeyError:
        dicci[x] = 'NUEVO'
        raise
except KeyError:
    print("Vuelvo a manejar KeyError...")
except:
    print("Esto es por cualquier otra...")
dicci
```

18 Algunas de las excepciones predefinidas (Built-in)

- **ImportError**: error con importación de módulos.
- **ModuleNotFoundError**: error por módulo no encontrado.
- **IndexError**: error por índice fuera de rango.
- **KeyError**: error por clave inexistente.
- **NameError**: error por nombre no encontrado.
- **SyntaxError**: error por problemas sintácticos
- **ZeroDivisionError**: error por división por cero.
- **IOError**: error en entrada salida.

Listado completo

19 ¿Es posible acceder a la información de contexto de la excepción?

[ ]:
```
dicci = {0:"Led Zeppelin", 2:"Deep Purple", 3:"Black Sabbath"}
try:
    print ('Entramos al bloque try')
    for x in range(1,6):
        if x == 2 or x == 3:
            raise KeyError
        else:
            print (dicci[x])
    print('Continuamos con el proceso..')
except KeyError as exc:
    dicci[x] = 'NUEVO'
    datos_exc = exc
    #import sys
```

```
    #print(sys.exc_info())
print(datos_exc)
```

- **sys.exc_info()**

[ ]:
```
x = 10
y = 0
try:
    z = x/y
except ZeroDivisionError as e:
    z = e # representation: "<exceptions.ZeroDivisionError instance at␣
 ↪0x817426c>"
print(z) # output: "integer division or modulo by zero"
```

20 En resumen:

```
try:
    sentencias
except excepcion1, excepcion2:
    sentencias
except excepcion3 as variable:
    sentencias
except:
    sentencias
else:
    sentencias
finally:
    sentencias
```

21 Desafío 2

- Dado el dataset con datos de video juegos del Apple store.
- Armar un menú con PySimpleGUi que permita:
    1. listar los juegos gratuitos disponibles en idioma español.
    2. los íconos (OPCIONAL en formato imagen, sino la url) de los 10 juegos con más calificaciones de usuarios (User Rating Count).
- Los que deseen, lo pueden subir a GitHub y compartir solución con @clauBanchoff
- Incluir manejo de excepciones donde consideren adecuado.
- También pueden descargar el archivo en: https://archivos.linti.unlp.edu.ar/index.php/s/D0YR0jqOx1GQtSI

Ayuda: para recuperar las imágenes de los íconos podemos usar el **módulo requests** - Se instala con pip: **pip install requests** - +Info

[ ]:
```python
# Ayuda para el punto 2
import requests
import os

juego = "Gloomhaven"
archivo = os.path.join(os.getcwd(), "archivos", juego)

icon_url = "https://cf.geekdo-images.com/original/img/
 ↪lDN358RgcYvQfYYN6Oy2TXpifyM=/0x0/pic2437871.jpg"
icono = requests.get(icon_url)
```

[ ]:
```python
with open(f'{archivo}.jpg', 'wb') as f:
    f.write(icono.content)
```

22 Hablamos del trabajo integrador