Clase7_Excepciones cont.

April 24, 2023

1 Seminario de Lenguajes - Python

1.1 Cursada 2023

1.1.1 Clase 7: continuamos con las excepciones

2 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.

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 Excepciones en Python

Habíamos visto la siguiente estructura:

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

- +Info

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

4.0.1 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...""")
    except:
        print("Ups! Hubo un error inesperado... Intentá de nuevo")
        # Esto no lo vamos a poder resolver ingresando nuevamente

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

5 Repasemos un poco más con este otro ejemplo:

[ ]:
```
dicci = {1980:"Soda Stereo", 2010:"Jauria", 1990:"Los Piojos"}
```

[ ]:
```
try:
    print("Entrando al primer try ...")
    clave = 1970
    try:
        print (f"La mejor banda de {clave} es {dicci[clave]}")
    except NameError:
        print("TRY INTERNO: clave inexistente.")

    print("Saliendo del primer try ... ")

except KeyError:
        print("TRY EXTERNO: clave inexistente.")
print('Sigo con mi programa....')
```

- ¿Se ejecuta línea 9?

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

6 ¿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..

[ ]:
```
dicci = {1980:"Soda Stereo", 2010:"Jauria", 1990:"Los Piojos"}
```

[ ]:
```
def super_bandas(clave):
    try:
        return dicci[clave]
    except NameError:
        print("Ups! Hay un problema con algo mal definido.")

try:
    print("Entrando al primer try ...")
    clave = 1970
    try:
        print (f"La mejor banda de {clave} es {super_bandas(clave)}")
    except KeyError:
        print("TRY INTERNO: clave inexistente.")

    print("Saliendo del primer try ... ")

except KeyError:
        print("TRY EXTERNO: clave inexistente.")
print('Sigo con mi programa....')
```

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 La sentencia completa

```
try:
    sentencias
except excepcion1, excepcion2:
    sentencias
except:
    sentencias
else:
    sentencias
finally:
    sentencias
```

8.0.1 Teníamos una tarea....

9 Veamos este ejemplo sencillo

[ ]:
```
XY = 10
try:
    print(XY)
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")
```

9.0.1 Entonces, ¿para qué usamos else y finally?

10 else y finally

Se utiliza la cláusula **else** para incluir el código que se debería ejecutar si **no se levanta ninguna excepción** en el bloque try..except.

Se utiliza la cláusula **finally** para incluir el código que se ejecuta **siempre**, independientemente si se levanta o no alguna excepción en el bloque try..except

11 Observemos el bloque finally en este otro ejemplo:

[ ]:
```
dicci = {1980:"Soda Stereo", 2010:"Jauria", 1990:"Los Piojos"}
```

[ ]:
```
try:
    print("Entrando al primer try ...")
    clave = 1970
    try:
        print (f"La mejor banda de {clave} es {dicci[clave]}")
```

```
    except NameError:
        print("TRY INTERNO: clave inexistente.")
    finally:
        print("Saliendo del primer try ... ")

except KeyError:
        print("TRY EXTERNO: clave inexistente.")
print('Sigo con mi programa....')
```

- Podemos observar que la línea 9 ahora se ejecuta.

12 Podemos levantar explícitamente excepciones

[ ]:
```
dicci = {1980:"Soda Stereo", 2010:"Jauria", 1990:"Los Piojos"}
```

[ ]:
```
try:
    print ('Entramos al bloque try')
    for x in [1980, 1990, 2000, 2005, 2010]:
        if x == 2000:
            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 2005?

13 También es posible:

[ ]:
```
try:
    for x in [1980, 1990, 2000, 2005, 2010]:
        if x == 2000:
            raise
        else:
            print(dicci[x])
    print('Continuamos con el proceso..')
except KeyError:
    dicci[x] = 'NUEVO'
    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.

[ ]:
```
try:
    print ('Entramos al TRY EXTERNO')
    try:
        print("Entramos TRY INTERNO")
        for x in [1980, 1990, 2000, 2005, 2010]:
            if x == 2000:
                raise KeyError
            else:
                print(dicci[x])
        print('Continuamos con el proceso..')
    except KeyError:
        print("Manejando KeyError en el TRY INTERNO")
        raise
except KeyError:
    print("Manejando KeyError en el TRY EXTERNO")
except:
    print("Esto es por cualquier otra...")
```

14 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

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

[ ]:
```
dicci = {1980:"Soda Stereo", 2010:"Jauria", 1990:"Los Piojos"}
```

[ ]:
```
try:
    print ('Entramos al bloque try')
    for x in [1980, 1990, 2000, 2005, 2010]:
        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
print(z)
```

16 En resumen:

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

17 Hacemos una autoevaluación de lo visto