# Clase6_Excepciones

April 17, 2023

# 1 Seminario de Lenguajes - Python

## 1.1 Cursada 2023

### 1.1.1 Clase 6. Introducción al manejo de excepciones en Python

# 2 Probemos los siguientes códigos:

[ ]:
```
#archivo = open("pppppppp.txt")
```

[ ]:
```
mis_notas = (10, 7, 7)
#mis_notas[0] = 11111
#print(mis_notas[5])
```

## 2.1 ¡Esto no puede pasar nunca en nuestros programas!

# 3 ¿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 ¿Qué situaciones pueden producir excepciones?

- Abrir un archivo que no existe o donde no tenemos permisos adecuados.
- Acceder a un elemento de un diccionario con una clave que no existe.
- Invocar a un método o función que no fue definida.
- Referirse a una variable que no fue definida.
- Mezclar tipos de datos sin convertirlos previamente.
- Etc.

# 5 Excepciones "sin manejar"

# 6 ¿Qué debemos investigar para trabajar con excepciones?

Primero: ¿el lenguaje de programación tiene soporte para el manejo de excepciones? - Si no presenta ningún mecanísmo para esto, podríamos simularlo con otros recursos. Ejemplo: Pascal o C. - Si

provee mecanismos para el manejo de excepciones: ¿cuáles? Ejemplo: **Python**, Javascript, Java, Ruby, etc.

### 6.1 Si el lenguaje provee manejo de 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 se definen los manejadores de excepciones?
- ¿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?

## 7 Excepciones en Python

Se utiliza el bloque **try: except:**

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

- +Info

## 8 Veamos un ejemplo

[ ]:
```
x = 10
try:
    print(XX)
except NameError:
    print("Usaste una variable que no está definida")
```

### 8.1 Analicemos el siguiente código:

[ ]:
```
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: ")
decada = int(input("ingresá a qué década pertenece: "))
mi_musica[decada].append(tema)
```

- ¿Cuándo se puede producir una excepción en este código?
- ¿Qué excepciones se pueden producir?
- ¿Dónde ubicamos las sentencias que **manejen** estas excepciones?

## 9 Agregamos los manejadores

[ ]:
```
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: ")

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.")
except KeyError:
    print("Por ahora, sólo tengo registradas las décadas: 70, 80 y 2000.␣
 ↪Ingresá una de ellas. ")
```

- Ahora el programa no se **rompe**.

### 9.0.1 ¿Qué podemos decir del siguiente ejemplo? ¿Dónde pondríamos los manejadores?

[ ]:
```
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): ")

print("Terminé de procesar mi música favorita")
```

## 10 Analicemos esta propuesta:

```
[ ]: 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): ")
     try:
         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): ")
         print("Terminé de procesar mi música favorita")
     except ValueError:
         print("Para ingresar la decada, tenés que ingresar un número.")
     except KeyError:
         print("Por ahora, sólo tengo registradas las décadas: 70, 80 y 2000.␣
     ↪Ingresá una de ellas. ")
```

### 10.1 Python FINALIZA el bloque que levanta la excepción

- ¿Cuál es el bloque que finaliza?

  Este mecanismo se conoce como **terminación**.

## 11 En resumen

El manejo de excepciones hizo que al intentar acceder al diccionario con una clave inexistente o intentar realizar la conversión a **int** errónea: - el programa **no se rompa**, - tomamos alguna acción. En este caso, **informamos cuál es el problema** y, - el programa continúa con la ejecución hasta el final.

Pero: **¿el while finaliza siempre con FIN?**

## 12 ¿Y si no queremos que se corte el ingreso de datos?

```
[ ]: 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): ")
print("Terminé de procesar mi música favorita")
```

- ¿Cuál es el bloque que finaliza su ejecución?
- Afectamos el manejador de excepciones SOLO a la sentencia que intenta acceder el diccionario.

## 13 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): ")
print("Terminé de procesar mi música favorita")
```

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

## 14 ¿Cómo buscamos el manejador?

[ ]:
```
bandas_rock = {0:"Led Zeppelin", 2:"Deep Purple", 3:"Black Sabbath"}
try:
    for x in range(0,6):
        try:
            print(bandas_rock[z])      # OJO que estamos usando la variable z
        except KeyError:
            print("Ups! Parece que hubo un problema..")
except NameError:
    print('OJO! Se está usando una variable que no existe')
```

```
print('Sigo con mi programa....')
```

## 15 ¿Y en este otro caso?

[ ]:
```
def retornar_banda(indice):
    bandas_rock = {0:"Led Zeppelin", 2:"Deep Purple", 3:"Black Sabbath"}
    try:
        return bandas_rock[indice] # OJO estamos usando una variable no␣
 ↪definida!
    except NameError:
        print("Ups! Hay un problema con algo mal definido.")

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_banda(elem)}")
    except KeyError:
        print("Usás una clave inexistente.")

    elem = int(input('Ingresá clave para acceder al diccionario: (999 para␣
 ↪finalizar) '))

print('Sigo con mi programa....')
```

## 16 ¿Qué sucedió?

- La excepción KeyError se levantó dentro de la función **retornar_elemento**.
- **Busca estáticamene** 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.

## 17 ¿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..

## 18 Ya respondimos algunas de las preguntas iniciales:

- **¿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?

## 19 Desafío

Analizar el código del visualizador de csv que vimos la clase pasada respecto al manejo de excepciones que realiza.

## 20 En realidad... la sentencia completa es:

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

## 21 Tarea para el hogar:

- Investigar para qué se utilizan las cláusulas **finally** y **else** en el bloque try.. except

## 22 Seguimos la próxima ...