Clase6_GUI_excepciones

April 18, 2022

1 Seminario de Lenguajes - Python

1.1 Cursada 2022

1.1.1 Clase 6: GUI - Introducción al manejo de excepciones en Python.

2 ¿Saben qué son las GUI?

2.1 GUI: Graphical User Interface

3 GUI en Python

- Vamos a usar PySimpleGUI
- Es un **framework** bastante simple para desarrollar interfaces gráficas en Python.
- Es software libre: https://github.com/PySimpleGUI/PySimpleGUI
- Se instala con pip: **pip install pysimplegui**
- Vamos a ir de a poco...
- Más info en: https://pysimplegui.readthedocs.io/en/latest/cookbook/
- También en https://realpython.com/pysimplegui-python/

4 ¿Con qué elementos podemos trabajar?

5 Popups: las ventanas más sencillas

[ ]:
```
import PySimpleGUI as sg
sg.Popup('Mi primera ventanita')
```

- **import PySimpleGUI as sg**, permite acceder a los recursos por el nombre **sg**.

[ ]:
```
import PySimpleGUI as sg

sg.Popup('Mi  primera ventanita', button_color=('black', 'red'))

sg.PopupYesNo('Mi  primera ventanita', button_color=('black', 'green'))
```

```
sg.PopupOKCancel('Mi  primera ventanita', button_color=('black', 'grey'))

texto = sg.PopupGetText('Titulo', 'Ingresá algo')
sg.Popup('Resultados', 'Ingresaste el siguiente texto: ', texto)
```

6 Creamos una ventana en PySimpleGUI

[ ]:
```
import PySimpleGUI as sg

sg.Window(title="Hola Mundo!", layout=[[]], margins=(100, 50)).read()
```

- **margins**: tamaño de la ventana en pixeles.
- **read()**: devuelve una **tupla** con un **evento** y **los valores** de todos los elementos de entrada en la ventana.

7 Sobre los eventos

[ ]:
```
import PySimpleGUI as sg

layout = [[sg.Text("Hola Mundo!")], [sg.Button("OK")]]

window = sg.Window("Primera Demo", layout, margins=(200, 150))

while True:
    event, values = window.read()

    if event == "OK" or event == sg.WIN_CLOSED:
        break

window.close()
```

- ¿Qué representa **sg.WIN_CLOSED**?
- ¿Cuándo termina este programa?

8 ¿Cómo recuperamos los valores ingresados desde la UI?

[ ]:
```
import PySimpleGUI as sg

layout = [ [sg.Text('Ingresá primer valor'), sg.InputText()],
           [sg.Text('Ingresá segundo valor'), sg.InputText()],
           [sg.Button('Ok'), sg.Button('Cancel')] ]

window = sg.Window("Segunda Demo", layout, margins=(200, 150))

while True:
```

```
    event, values = window.read()

    if event == "Cancel" or event == sg.WIN_CLOSED:
        break
    print('Datos ingresados: ', values)

window.close()
```

- ¿De qué tipo es **values**?

9 Layout: ¿cómo organizamos la UI?

Representa al esquema o diseño de nuestra UI: **cómo se distribuyen los elementos en la UI**.

[ ]:
```
layout = [ [sg.Text('Ingresá primer valor'), sg.InputText()],
           [sg.Text('Ingresá segundo valor'), sg.InputText()],
           [sg.Button('Ok'), sg.Button('Cancel')] ]
```

- ¿De qué tipo es la variable layout?
- ¿Qué elementos estamos incluyendo?

10 Elementos de la UI

- Acá van algunos disponibles en PySimpleGUI
  - Buttons: File Browse, Folder Browse, Color chooser, Date picker, etc.
  - Checkbox, Radio Button, Listbox
  - Slider, Progress Bar
  - Multi-line Text Input, Scroll-able Output
  - Image, Menu, Frame, Column, Graph, Table
- Referencias: https://pysimplegui.readthedocs.io/en/latest/call%20reference/#element-and-function-call-reference

11 Agreguemos elementos a nuestra ventana

[ ]:
```
import PySimpleGUI as sg

layout = [[sg.Input('Ingresa algo')],
          [sg.Listbox(values=('Item 1', 'Item 2', 'Item 3'))],
          [sg.OK()]]

window = sg.Window("Elementos básicos", layout, margins=(200, 150))
event, values = window.read()
```

```
sg.Popup(event, values, line_width=200)
```

[ ]:
```
sg.ChangeLookAndFeel('DarkAmber')

layout = [[sg.Listbox(values=('Item 1', 'Item 2', 'Item 3'),␣
 ↪background_color='yellow', size=(20,3)),
          [sg.Input('Last input')],
          [sg.ColorChooserButton("  Elegi color")],
          [sg.OK()]] ]
```

12 PySimpleGUI en GitHub

- En GitHub hay mucha información y referencias útiles.
- Por ejemplo, hay varios ejemplos que pueden servir de base para nuestros desarrollos.

13 ¡¡Las ventajas del software libre!!

14 Probemos los siguientes códigos:

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

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

14.1 ¡Esto no puede pasar nunca en nuestros programas!

15 ¿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.

16 ¿Qué situaciones pueden levantar 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.

17 Excepciones “sin manejar”

18 ¿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.

19 Si tenemos recursos para manejar excepciones...

19.0.1 ¿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?

Vamos a responder esto en estas clases

20 Excepciones en Python

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

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

- +Info

21 Veamos un ejemplo

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

22 ¿Dónde debería estar el manejador de la excepción?

[ ]:
```
bandas = {
    "William Campbell": {"ciudad": "La Plata", "ref": "www.instagram.com/
 ↪williamcampbellok"},
    "Buendia": {"ciudad": "La Plata", "ref":"https://buendia.bandcamp.com/"},
    "Lúmine": {"ciudad": "La Plata", "ref": "https://www.instagram.com/luminelp/
 ↪"},
    "La Renga": {"ciudad": "XXXX", "ref": "https://www.larenga.com/"},
    "Divididos": {"ciudad": "XXXX", "ref": "http://divididos.com.ar/"}}
mis_bandas = []
nombre_banda =  input("Ingresá el nombre de la banda que te gusta")
mis_bandas.append(bandas[nombre_banda]["ref"])

print(mis_bandas)
```

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

[ ]:
```
bandas = {
    "William Campbell": {"ciudad": "La Plata", "ref": "www.instagram.com/
 ↪williamcampbellok"},
    "Buendia": {"ciudad": "La Plata", "ref":"https://buendia.bandcamp.com/"},
    "Lúmine": {"ciudad": "La Plata", "ref": "https://www.instagram.com/luminelp/
 ↪"},
    "La Renga": {"ciudad": "XXXX", "ref": "ALGUNA"},
    "Divididos": {"ciudad": "XXXX", "ref": "xxx"}}
mis_bandas = []
nombre_banda =  input("Ingresá el nombre de la banda que te gusta")

try:
     mis_bandas.append({"banda": nombre_banda, "url":bandas[nombre_banda]})
except KeyError:
    print("Ingresaste el nombre de una banda que no tengo registrada")

print(mis_bandas)
```

- Ahora el programa no se **rompe** si ingreso un valor que NO es clave en mi diccionario.

23 Seguimos con las bandas...

[ ]:
```
mis_bandas = []
nombre_banda =  input("Ingresá el nombre de la banda que te gusta")
try:
    while nombre_banda != "FIN":
        mis_bandas.append({"banda": nombre_banda, "datos":bandas[nombre_banda]})
```

```
        nombre_banda =  input("Ingresá el nombre de la banda que te gusta")
except KeyError:
    print("Ingresaste el nombre de una banda que no tengo registrada. Intentalo␣
 ↪de nuevo.")
print(mis_bandas)
```

23.0.1 ¿Qué podemos decir de este ejemplo? ¿Es cierto que siempre finalizamos la iteración con "FIN"?

- ¿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?

24 Python FINALIZA el bloque que levanta la excepción

- En el caso anterior: ¿cuál es el bloque que finaliza?

[ ]:
```
mis_bandas = []
nombre_banda =  input("Ingresá el nombre de la banda que te gusta")
try:
    while nombre_banda != "FIN":
        mis_bandas.append({"banda": nombre_banda, "datos":bandas[nombre_banda]})
        nombre_banda =  input("Ingresá el nombre de la banda que te gusta")
    print("Hola")
except KeyError:
    print("Ingresaste el nombre de una banda que no tengo registrada. Intentalo␣
 ↪de nuevo.")
print(mis_bandas)
```

El manejo de excepciones hizo que al intentar acceder al diccionario con una clave inexistente: - el program **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?

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

[ ]:
```
mis_bandas = []

nombre_banda =  input("Ingresá el nombre de la banda que te gusta")
while nombre_banda != "FIN":
    try:
        mis_bandas.append({"banda": nombre_banda, "datos":bandas[nombre_banda]})
    except KeyError:
        print("Ingresaste el nombre de una banda que no tengo registrada.␣
 ↪Intentalo de nuevo.")

    nombre_banda =  input("Ingresá el nombre de la banda que te gusta")
```

```
print(mis_bandas)
```

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

26 Tarea para el hogar:

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

27 Seguimos la próxima ...