Day - 30

**Exception Handling**

**& JSON**

try, except, else and finally, JSON

**30.1 Catching Exceptions:** try**,** except, else **and** finally

* Types of different ERR:

#*FileNotFoundError*

**with** **open**("test\_exception.txt") **as** file:

    file**.read**()

#*KeyError*

demo\_dict = {"key" : "value"}

value = demo\_dict["text"]

#*IndexError*

demo\_list = ["peach", "apple", "orange"]

fruit = demo\_list[3]

#*TypeError*

text = "abc"

num = text + 6



* Catching the exceptions:

1. try: Something that might cause an exception

**with** **open**("test\_exception.txt") **as** file:

    file**.read**()

1. except: Do this if there was an exception. According to the PEP 8 *recommendations*, you should never use a bare ***except*** and the reason for this is because when you have an except clause, then ***it's actually going to ignore all errors***.

#*FileNotFoundError*

**try**:

    file = **open**("test\_exception.txt")

    #*Following error will be skipped*

    demo\_dict = {"key" : "value"}

    value = demo\_dict["text"]

**except**:

**print**("File does not exist. New file Crated")

    file = **open**("test\_exception.txt", "a")

* Too broad exception clause: Using bare ***except***: For example, in above block once the file is created the key-error is never gonna caught. The reason for this is because inside the try block:
  + It's first trying to open up this file and if that file exists, it moves on to the next line. It creates a dictionary and it tries to get hold of the value with a key. This line actually fails and creates an exception, but that exception is caught by the following broad exception " **except:** ".
* Specify the error: Do not use bare except, always specify the ERR-type

**except** FileNotFoundError:

**print**("File does not exist. New file Crated")

    file = **open**("test\_exception.txt", "a")

* Multiple **except**:

**try**:

    file = **open**("test\_exception.txt")

    #*Following error will be skipped*

    demo\_dict = {"key" : "value"}

    value = demo\_dict["text"]

**except** FileNotFoundError:

**print**("File does not exist. New file Crated")

    file = **open**("test\_exception.txt", "a")

**except** KeyError:

**print**("Key does not exist.")

* Holding/Printing error message:

**except** KeyError **as** err\_msg:

**print**(f"Key {err\_msg} does not exist.")

1. else: Do this if there were no exceptions. If no exceptions that were thrown from any block of code, then it's going to jump to the else block. Remember, if any exception appears, this else block is never going to be triggered.

**try**:

    file = **open**("test\_exception.txt")

    #*Following error will be skipped*

    demo\_dict = {"key" : "value"}

    value = demo\_dict["text"]

**except** FileNotFoundError:

**print**("File does not exist. New file Crated")

    file = **open**("test\_exception.txt", "a")

    file**.write**("something")

**except** KeyError **as** err\_msg:

**print**(f"Key {err\_msg} does not exist.")

**else**:

    content = file**.read**()

**print**(content)

1. finally: Do this no matter what happens. this finally is basically some code that's gonna run no matter what happens

#*FileNotFoundError*

**try**:

    file = **open**("test\_exception.txt")

    #*Following error will be skipped*

    demo\_dict = {"key" : "value"}

    value = demo\_dict["text"]

**except** FileNotFoundError:

**print**("File does not exist. New file Crated")

    file = **open**("test\_exception.txt", "a")

    file**.write**("something")

**except** KeyError **as** err\_msg:

**print**(f"Key {err\_msg} does not exist.")

**else**:

    content = file**.read**()

**print**(content)

**finally**:

    file**.close**()

**print**("File was closed")

**30.2 Raising your own Exceptions**

As a Python developer you can choose to throw an exception if a condition occurs. To throw (or raise) an exception, use the ***raise*** keyword. The ***raise*** keyword is used to raise an exception.

* Example: Raise an error and stop the program if x is lower than 0:

#*Raise an error and stop the program if x is lower than 0:*

x = -1

**if** x **<** 0:

**raise** **Exception**("Sorry, no numbers below zero")

* Example: You can define what kind of error to raise, and the text to print to the user.

#*You can define what kind of error to raise, and the text to print to the user.*

x = "hello"

**if** **not** **type**(x) **is** int:

**raise** **TypeError**("Only integers are allowed")

* Example:

y = "Hollow - Point"

**print**(isinstance(y, str))

**if** **isinstance**(y, str):

**raise** **Exception**("Oh Boy!!! This is a String. Give some number man! ")

#*print(isinstance(y, str))*

* Type comparison:

**if** **isinstance**(obj, MyClass): **do\_foo**(obj)

Example:

y = "Hollow - Point"

**print**(isinstance(y, str))

* Exercise 30.1: BMI calculator: Catching Exception

height = **float**(input("Enter the height :"))

weight = **float**(input("Enter the weight :"))

**if** height **>** 3:

**raise** **ValueError**("Human hieight should be below 3 meters!! ")

BMI = weight/height\*\*2

**print**(BMI)

**30.3 IndexError** Handling**, KeyError** Handling **with Exercise**

* Exercise 30.2:

fruits = ["Apple", "Pear", "Orange"]

#*TODO: Catch the exception and make sure the code works*

**def** **make\_pie**(index):

**try**:

        fruit = fruits[index]

**print**(fruit + " pie")

**except** IndexError:

**print**("Fruit pie")

**make\_pie**(4)

#*python index\_err\_handl.py*

* Exercise 30.3:

facebook\_posts = [

    {'Likes': 21, 'Comments': 2},

    {'Likes': 13, 'Comments': 2, 'Shares': 1},

    {'Likes': 33, 'Comments': 8, 'Shares': 3},

    {'Comments': 4, 'Shares': 2},

    {'Comments': 1, 'Shares': 1},

    {'Likes': 19, 'Comments': 3}

]

total\_Likes = 0

**for** post **in** facebook\_posts:

**try**:

        total\_Likes = total\_Likes + post['Likes']

**except** KeyError:

**pass**

**print**(f"Total Likes : {total\_Likes}")

#*python keyword\_err\_handl.py*

* Exercise 30.4: 6.1 NATO Phonetic Alphabet for the Code Exercise

#*Keyword Method with iterrows()*

#*{new\_key:new\_value for (index, row) in df.iterrows()}*

**import** pandas

data = pandas**.read\_csv**("nato\_phonetic\_alphabet.csv")

phonetic\_dict = {row**.**letter: row**.**code **for** (index, row) **in** data**.iterrows**()}

**print**(phonetic\_dict)

**def** **nato\_aplhbt**():

    word = **input**("Enter a word: ")**.upper**()

**try** :

        output\_list = [phonetic\_dict[letter] **for** letter **in** word]

**except** KeyError:

**print**("Sorry, only letters in the alphabet please.")

**nato\_aplhbt**()

**else**:

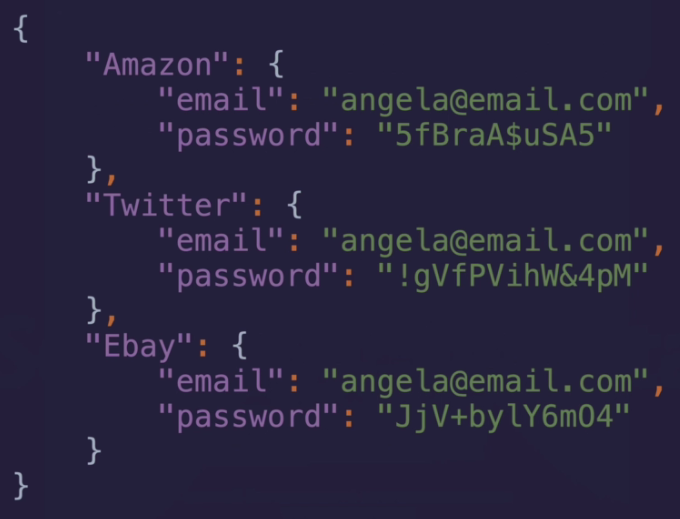
**print**(output\_list)

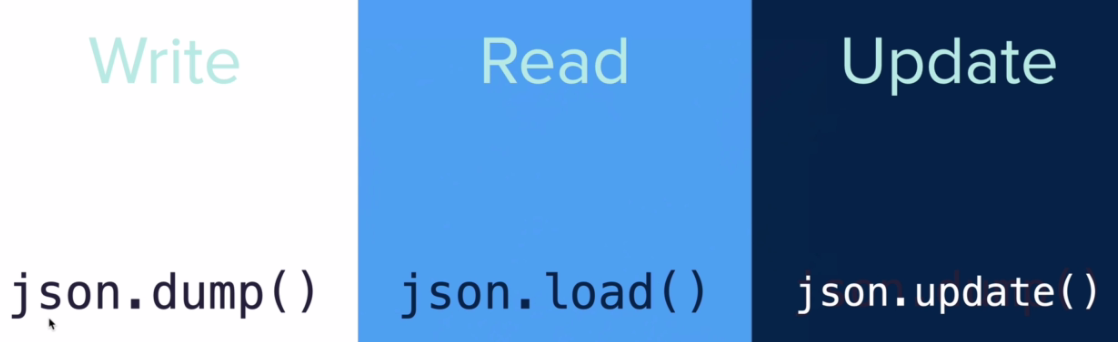
**nato\_aplhbt**()

#*python nato\_phonetic\_alphabet\_main.py*

**3.4 JSON: JavaScript Object Notation**

A JSON is essentially ***composed*** of a bunch of ***nested*** ***lists*** and ***dictionaries***, and it has that key -value pair *data structure*.





* To work with JSON data in Python, we can use the inbuilt JSON library, and we're going to use it to write, read, and update data to a JSON file.

1. Create JSON Data Dictionary format

    new\_data = {

        website : {

            "email" : email,

            "password": password

        }

    }

1. Dump: json module just cannot append data to the existing file. First it lodes data by ***load()*** and ***update*** that data and then truncate/delete all data from file and ***writes*** the ***new*** data. ***dump()*** works onn "***w***" mode:

        #*Now we create a Json file. Change "a" mode to "w" mode*

**with** **open**("data.json", "w") **as** data\_file:

            #*Dump "new\_data" to json.dump(). Use indent for "readability"*

            json**.dump**(new\_data, data\_file, indent=4) #*This is SERIALIZE*

1. Load: Loads the ***Json*** data and converts to ***Dictionary*** data format, it works on "r" mode

**with** **open**("data.json", "r") **as** data\_file:

            #*Dump "new\_data" to json.dump(). Use indent for "readability"*

            #*json.dump(new\_data, data\_file, indent=4) # This is SERIALIZE*

            json\_data = json**.load**("data.json")

**print**(**type**(json\_data ))

**print**(json\_data)

**30.5 Write, read and update JSON data in the Password Manager**

#*Import Json Module*

**import** json

**def** **save**():

    website = website\_entry**.get**()

    email = email\_entry**.get**()

    password = password\_entry**.get**()

    #*Create JSON Data Dictionary format*

    new\_data = {

        website : {

            "email" : email,

            "password": password

        }

    }

**if** **len**(website) **==** 0 **or** **len**(password) **==** 0:

        messagebox**.showinfo**(title="Oops", message="Please make sure you haven't left any fields empty.")

**else**:

        #*Now we create a Json file. Change "a" mode to "w" mode*

**with** **open**("data.json", "w") **as** data\_file:

            #*Dump "new\_data" to json.dump(). Use indent for "readability"*

            json**.dump**(new\_data, data\_file, indent=4) #*This is SERIALIZE*

             website\_entry**.delete**(0, END)

             password\_entry**.delete**(0, END)

* Serialize/De-Serialize : LOAD-Update-DUMP all works in 3-steps with read "r" file mode and write "w" file mode

#*---------------------------- SAVE PASSWORD ------------------------------- #*

**def** **save**():

    website = website\_entry**.get**()

    email = email\_entry**.get**()

    password = password\_entry**.get**()

    #*Create json Data Dictionary format*

    new\_data = {

        website : {

            "email" : email,

            "password": password

        }

    }

**if** **len**(website) **==** 0 **or** **len**(password) **==** 0:

        messagebox**.showinfo**(title="Oops", message="Please make sure you haven't left any fields empty.")

**else**:

        #*Now we create a Json file. Change "a" mode to "w" mode. "dump" in "w" mode but "load" in "r" mode*

        #*"load" in "r" mode*

**with** **open**("data.json", "r") **as** data\_file:

            #*Dump "new\_data" to json.dump(). Use indent for "readability"*

            #*json.dump(new\_data, data\_file, indent=4) # This is SERIALIZE*

            #*Load old data*

            json\_data = json**.load**(data\_file) #*This is de-SERIALIZE*

            #*Update old data with new data. Notice the object is used not the "json"*

            json\_data**.update**(new\_data)

        #*"dump" in "w" mode*

**with** **open**("data.json", "w") **as** data\_file:

            #*Saving the new data. Notice "json module" is used*

            json**.dump**(json\_data, data\_file, indent=4)

**print**(**type**(json\_data ))

**print**(json\_data)

            website\_entry**.delete**(0, END)

            password\_entry**.delete**(0, END)

**30.6 Handling Exceptions in the Password Manager**

1. To prevent file not found error put ***load()*** and read file "***r***" in try block.
2. Then ***create file*** if doesn't exist. In "***except***"
3. Put ***update()*** and ***dump()*** with write file "***w***".
4. In ***finally*** clear the ***input*** field.

#*Import Json Module*

**import** json

#*---------------------------- SAVE PASSWORD ------------------------------- #*

**def** **save**():

    website = website\_entry**.get**()

    email = email\_entry**.get**()

    password = password\_entry**.get**()

    #*Create json Data Dictionary format*

    new\_data = {

        website : {

            "email" : email,

            "password": password

        }

    }

**if** **len**(website) **==** 0 **or** **len**(password) **==** 0:

        messagebox**.showinfo**(title="Oops", message="Please make sure you haven't left any fields empty.")

**else**:

        #*Now we create a Json file. Change "a" mode to "w" mode. "dump" in "w" mode but "load" in "r" mode*

        #*"load" in "r" mode*

**try**:

**with** **open**("data.json", "r") **as** data\_file:

                #*Dump "new\_data" to json.dump(). Use indent for "readability"*

                #*json.dump(new\_data, data\_file, indent=4) # This is SERIALIZE*

                #*Load old data*

                json\_data = json**.load**(data\_file) #*This is de-SERIALIZE*

**except** FileNotFoundError:

**with** **open**("data.json", "w") **as** data\_file:

                #*Saving the new data. Notice "json module" is used*

                json**.dump**(new\_data, data\_file, indent=4)

**else**:

            #*Update old data with new data. Notice the object is used not the "json"*

            json\_data**.update**(new\_data)

            #*"dump" in "w" mode*

**with** **open**("data.json", "w") **as** data\_file:

                #*Saving the new data. Notice "json module" is used*

                json**.dump**(json\_data, data\_file, indent=4)

**finally**:

**print**(**type**(json\_data ))

**print**(json\_data)

            website\_entry**.delete**(0, END)

            password\_entry**.delete**(0, END)

**30.7 Search for a Website in the Password Manager.**

Practiced Version

**def** **seArch**():

    website = website\_entry**.get**()

**with** **open**("data.json", "r") **as** data\_file:

                search\_data = json**.load**(data\_file) #*This is de-SERIALIZE*

**if** website **in** search\_data:

        messagebox**.showinfo**(title="Found Data", message=f"Email: {search\_data[website]['email']} \n Pasword {search\_data[website]['password']}")

**else**:

        messagebox**.showinfo**(title="Data Notfound", message=f"Sorry There is no data")

search\_btn = **Button**(text = "Search", width = 10, command = seArch)

search\_btn**.grid**(row=1, column=2)

Instructor version

#*----------------- find Password ---------------------*

**def** **seArch**():

    website = website\_entry**.get**()

**try** :

**with** **open**("data.json", "r") **as** data\_file:

                    search\_data = json**.load**(data\_file) #*This is de-SERIALIZE*

**except** FileNotFoundError:

        messagebox**.showinfo**(title="Error", message=f"No Data File found .")

**else**:

        #*We could use "raise" in following for unstored data but if-else doing the job*

**if** website **in** search\_data:

            email = search\_data[website]['email']

            password = search\_data[website]['password']

            messagebox**.showinfo**(title="Found Data", message=f"Email: {email} \n Pasword {password}")

**else**:

            messagebox**.showinfo**(title="Data Notfound", message=f"Sorry There is no data")

**30.8 When use exceptions and when use If-Else?**

***Only use exception handling when you don't have an easy alternative.***

1. The if-else block works preemptively and stops the error from occurring while the try-except block handles the error after it has occurred. So, In try-except block system usage is more than if-else block.
2. If you can do something with ***if-else*** very easily, then you should stick to ***if-else***. If you can't do it with ***if-else*** very easily, and it's actually an error that's going to be thrown that you don't have any other way of dealing with, then you should be using the ***try-except-else-finally***, keywords.
3. The other way to think about it is that an ***exception*** is something that is meant to be exceptional. It's something that happens very rarely.

* Whereas ***if-else*** catches things that happen ***frequently***. For example, it's pretty often that your user might search through the website thinking that they saved some sort of details for a website. But in fact, they didn't.

All code at once

**from** tkinter **import** \*

**from** tkinter **import** messagebox

**from** random **import** choice, randint, shuffle

**import** pyperclip

#*Import Json Module*

**import** json

#*---------------------------- PASSWORD GENERATOR ------------------------------- #*

#*Password Generator Project*

**def** **generate\_password**():

    letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']

    numbers = ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']

    symbols = ['!', '#', '$', '%', '&', '(', ')', '\*', '+']

    password\_letters = [**choice**(letters) **for** \_ **in** **range**(**randint**(8, 10))]

    password\_symbols = [**choice**(symbols) **for** \_ **in** **range**(**randint**(2, 4))]

    password\_numbers = [**choice**(numbers) **for** \_ **in** **range**(**randint**(2, 4))]

    password\_list = password\_letters + password\_symbols + password\_numbers

**shuffle**(password\_list)

    password = ""**.join**(password\_list)

    password\_entry**.insert**(0, password)

    pyperclip**.copy**(password)

#*---------------------------- SAVE PASSWORD ------------------------------- #*

**def** **save**():

    website = website\_entry**.get**()

    email = email\_entry**.get**()

    password = password\_entry**.get**()

    #*Create json Data Dictionary format*

    new\_data = {

        website : {

            "email" : email,

            "password": password

        }

    }

**if** **len**(website) **==** 0 **or** **len**(password) **==** 0:

        messagebox**.showinfo**(title="Oops", message="Please make sure you haven't left any fields empty.")

**else**:

        #*Now we create a Json file. Change "a" mode to "w" mode. "dump" in "w" mode but "load" in "r" mode*

        #*"load" in "r" mode*

**try**:

**with** **open**("data.json", "r") **as** data\_file:

                #*Dump "new\_data" to json.dump(). Use indent for "readability"*

                #*json.dump(new\_data, data\_file, indent=4) # This is SERIALIZE*

                #*Load old data*

                json\_data = json**.load**(data\_file) #*This is de-SERIALIZE*

**except** FileNotFoundError:

**with** **open**("data.json", "w") **as** data\_file:

                #*Saving the new data. Notice "json module" is used*

                json**.dump**(new\_data, data\_file, indent=4)

**else**:

            #*Update old data with new data. Notice the object is used not the "json"*

            json\_data**.update**(new\_data)

            #*"dump" in "w" mode*

**with** **open**("data.json", "w") **as** data\_file:

                #*Saving the new data. Notice "json module" is used*

                json**.dump**(json\_data, data\_file, indent=4)

**finally**:

            website\_entry**.delete**(0, END)

            password\_entry**.delete**(0, END)

#*----------------- find Password ---------------------*

**def** **seArch**():

    website = website\_entry**.get**()

**try** :

**with** **open**("data.json", "r") **as** data\_file:

                    search\_data = json**.load**(data\_file) #*This is de-SERIALIZE*

**except** FileNotFoundError:

        messagebox**.showinfo**(title="Error", message=f"No Data File found .")

**else**:

        #*We could use "raise" in following for unstored data but if-else doing the job*

**if** website **in** search\_data:

            email = search\_data[website]['email']

            password = search\_data[website]['password']

            messagebox**.showinfo**(title="Found Data", message=f"Email: {email} \n Pasword {password}")

**else**:

            messagebox**.showinfo**(title="Data Notfound", message=f"Sorry There is no data")

#*---------------------------- UI SETUP ------------------------------- #*

window = **Tk**()

window**.title**("Password Manager")

window**.config**(padx=50, pady=50)

canvas = **Canvas**(height=200, width=200)

logo\_img = **PhotoImage**(file="logo.png")

canvas**.create\_image**(100, 100, image=logo\_img)

canvas**.grid**(row=0, column=1)

#*Labels*

website\_label = **Label**(text="Website:")

website\_label**.grid**(row=1, column=0)

email\_label = **Label**(text="Email/Username:")

email\_label**.grid**(row=2, column=0)

password\_label = **Label**(text="Password:")

password\_label**.grid**(row=3, column=0)

#*Entries*

website\_entry = **Entry**(width=25)

website\_entry**.grid**(row=1, column=1)

website\_entry**.focus**()

email\_entry = **Entry**(width=35)

email\_entry**.grid**(row=2, column=1, columnspan=2)

email\_entry**.insert**(0, "angela@gmail.com")

password\_entry = **Entry**(width=21)

password\_entry**.grid**(row=3, column=1)

#*Buttons*

generate\_password\_button = **Button**(text="Generate Password", command=generate\_password)

generate\_password\_button**.grid**(row=3, column=2)

add\_button = **Button**(text="Add", width=36, command=save)

add\_button**.grid**(row=4, column=1, columnspan=2)

search\_btn = **Button**(text = "Search", width = 10, command = seArch)

search\_btn**.grid**(row=1, column=2)

window**.mainloop**()

#*python main.py*