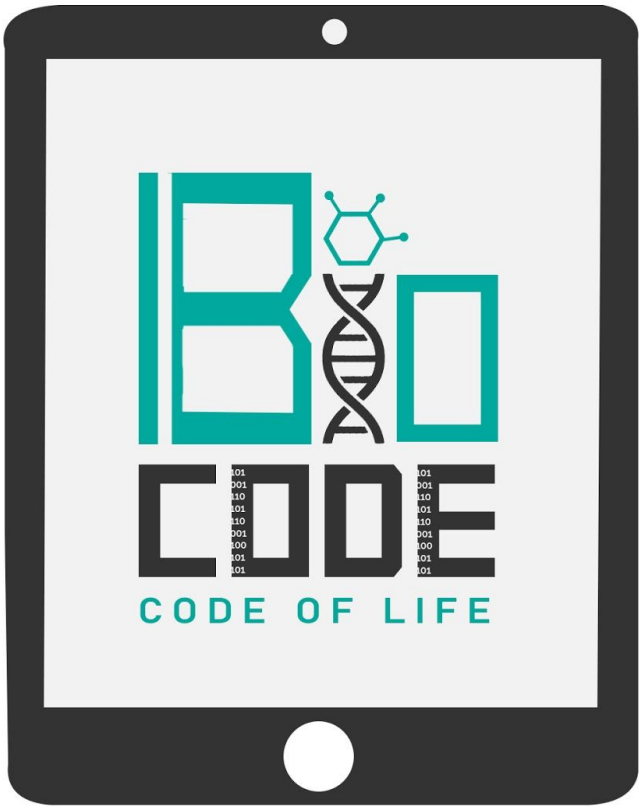

LEARN PYTHON & R FOR BIOINFORMATICS

Prerequisite Terminologies:

In order to understand 'Error handling' in python, you should know about the fundamental knowledge of the following term:

- **Errors & its types**

Introduction:

Error handling refers to the response and recovery procedures from error conditions that can be occurred while executing a program. Error handling is provided built-in within the python that can be utilized through certain keywords. It is an efficient way to improve your Python code that

you're trying to work with. When you think that you have a code that can cause an error you can utilize error handling to avoid the error causing code to crash your program.

➢ **Keywords for Error Handling:**

Error handling is utilized within python through four keywords. Three of the keywords are dedicated to error handling and one to if-else.

Keywords	How they work
try:	Code block 1 #....Run this code
except:	Code block 2 #....Execute this code when there is an error
else:	Code block 3 #....No error? Run this code
finally:	Code block 4 #....Always run this code

Steps:

- We'll first go through with the error handling keywords then utilize them to handle errors within python code.

```
try:
code block 1
#. . .some error prone code. . .
except:
code block 2
#. . .do something with the error. . .
else:
code block 3
#. . .to do when there is no error. . .
finally:
code block 4
#. . .some clean up code. . .
```

➔ **try:**

This segment will try to run the code and see if there is any error within your code. For example our code block 1 is causing an error, in that case the code that causes error will execute 'except' block of code.

➔ **except:**

When an error occurs in the 'try' block of code, then this block of code will run that says, you can do anything with the error you've received.

➔ **else:**

If there is no error in the 'try' code block, then any code in the 'else' block will run.

➔ **finally:**

Then code block 4 will run which is not related to the other three keywords. 'finally' block always executed after leaving the 'try' statement.

➢ **Possible Errors:**

- In any programming language, there are possible scenarios in which errors can occur while writing the code. For example, you've written code that reads a file and outposts the data available in the file. In case when you're doing it for personal use, you just provide the file name, try to read the file, open it and print out its results through python code, this would work fine. But when you share the file or you're trying to read other files, in this case you've to change the file and the directory.
- Error depends on how you write the code. So you should be efficient with writing the code or able to handle errors quite efficiently and provide the error in a human readable form.
- Create your file in PyCharm as **.py** and get working directory.

```
Import os
print(os.getcwd())
```

[It'll print out the working directory]

- Open and read a particular file (that is not present in the working directory), providing filename and read note.

```
file = open("filename" . "r")
print(file.read())
```

[It'll give an error as "FileNotFoundError" & No such file or directory: "filename that you provided"]

[So here we get an error while running the code because it is not a professional way to read a particular file as files can change or you can get new code within your directory over time.]

➢ **Utilizing Error Handling Keywords:**

- We'll try to handle error within the same code utilizing the keywords discussed earlier.

```
try:
    file = open("filename" , "r")
except:
     print("We didn't find the file you're trying to work with.")
     print("Please create a file named 'filename'.")
else:
     data = file.read()
     print(data)
```

[So here how the above code will work]

try	'Try to open and read the file' As no such file, it'll enter the 'except' block.

except	Show you error in human readable form as: “We didn’t find the file you’re trying to work with” and “Please create a file named ‘filename’.”
else	Due to an error, it’ll not enter the ‘else’ block.

- Now create a file that you were trying to open and read before and store any data in the file and run the following code:

```
Import os
print(os.getcwd())
 try:
    file = open(“filename” , “r”)
 except:
        print(“We didn’t find the file you’re trying to work with.”)
        print(“Please create a file named ‘filename’.”)
 else:
        data = file.read()
        print(data)
```

[Here how the above code will work]

try	‘Try to open and read the given file’ As the file now exists in the directory so it means there is no error.
except	As there is no error it’ll not enter the ‘except’ block.
else	No error, so it’ll open and print the data in the file.

Summary:

In this video, we learned how we can improve our code in Python and handle the errors in code utilizing error handling. We also got to see how error handling keywords work to handle the error within the Python code in an efficient way.