**OBJECT ORIENTED PROGRAMMING**

**CMPE 103**

**MODULE 1 – PYTHON’S STRING MANIPULATION**

**STRINGS**

Like many other popular programming languages, strings in Python are arrays of bytes representing Unicode characters. However, ***Python does not have a character data type, a single character is simply a string with a length of 1. Square brackets can be used to access elements of the string***.

**HOW TO CHANGE OR DELETE A STRING?**

***Strings are immutable***.

This means that elements of a string cannot be changed once it has been assigned. We can simply reassign different strings to the same name.

>>> my\_strin = ‘CPE’

>>> my\_strin[5] = 'a'

***TypeError: 'str' object does not support item assignment.***

**HOW TO CREATE A STRING IN PYTHON?**

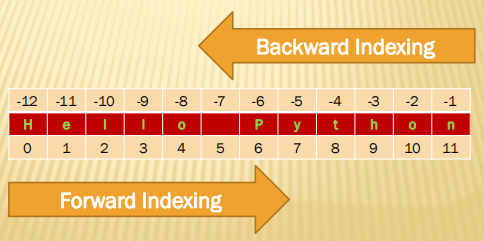
***How to create a string in Python?***

Strings can be ***created by enclosing characters inside a single quote or double quotes.*** Even triple quotes can be used in Python but generally used to represent multiline strings and docstrings.

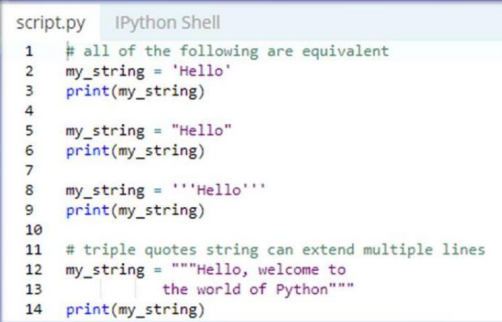
**REPRESENTATION OF STRING**

>>> s = “Hello Python”

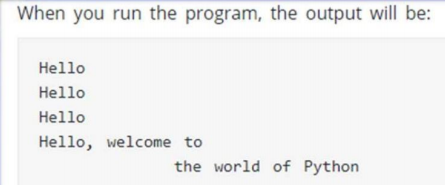
This is how Python would index the string:



**PROGRAMMING EXAMPLE**



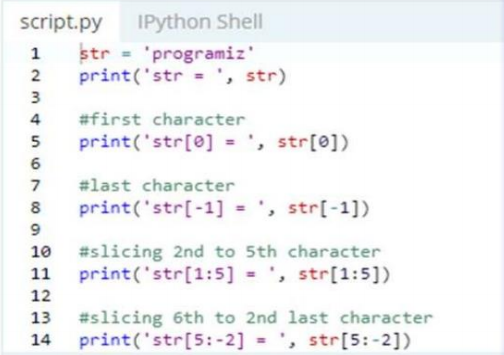
**OUTPUT**

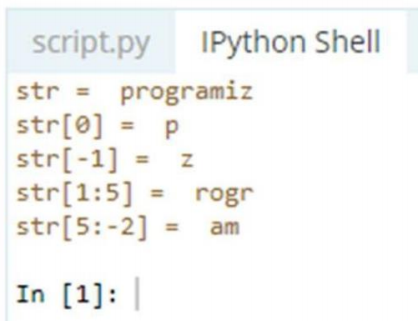


**HOW TO ACCESS CHARACTERS IN STRING**

We can ***access individual characters using indexing and a range of characters using slicing. Index starts from 0.*** Trying to access a character out of index range will raise an ***IndexError.* The index must be an integer. We can't use float or other types; this will result into TypeError.**

***Python allows negative indexing for its sequences. The index of -1 refers to the last item,*** -2 to the second last item and so on. We can access a range of items in a string by using the slicing operator (colon).





**SLICING STRINGS EXAMPLES**

For example:

>>>“Program”[3:5] will result in: ‘gr ’

>>>“Program”[3:6] will yield: ‘gra’

>>>p = “Program”

>>>p [:4] ‘Prog’

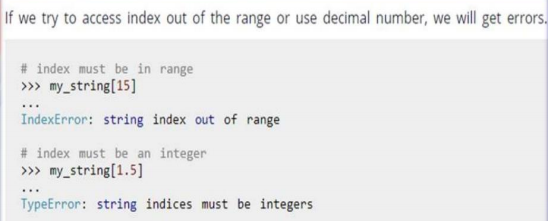
>>>p = “Program”

>>>p [4:] ‘ram’

>>>p = “Program”

>>>p [3:6] ‘gra’

**STRINGS –INDEX ERROR**



**MORE FUNCTIONALITY OF STRING**

**Finding Length of string**

>>> len(“Computer Engineering”)

**String Concatenation**

>>> print(“CMPE” + “103”)

**String Repeat**

>>> print(“A” \* 4 ) AAAA

**Substring Tests**

>>> ”A” in “Computer” True

>>> ”pr” in “computer” False

>>> ”pr” not in “computer” True

**MORE FUNCTIONALITY OF STRING**

>>> name1="computer"

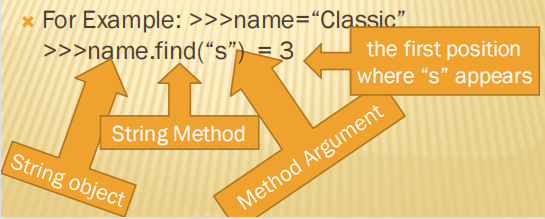
>>> name2=name1[3:5]

>>> name2

**STRING METHODS**

String Methods in Python, a method is a function that is defined with respect to a particular object.

Syntax: object.method(arguments)



**1. CAPITALIZE() METHOD**

Capitalizes first letter of string

>>> name=“computer”

>>> name.capitalize()

=== ‘Computer’

**2. LSTRIP() & 3. RSTRIP() METHODS**

lstrip() method is used to remove left padded spaces in a given string

>>> name1=“ a “

>>> name1.lstrip()

=== ‘a ‘

Removing left spaces

rstrip() method is used to remove right padded spaces in a given string

>>> name1.rstrip()

=== ‘ a’

Removing right spaces

**4. STRIP() METHOD**

strip() method is used to remove left and right padded spaces in a given string

>>> name1=“ a “

>>> name1.strip()

=== ‘a‘

Removing left and right spaces for a given string

**5. LOWER() METHOD**

lower() method is used to convert given string into lower case.

>>> name1=“ COMPUTER“

>>> name1.lower()

=== computer

**6. UPPER() METHOD**

upper() method is used to convert given string into upper case.

>>> name1=“ computer“

>>> name1.upper()

=== COMPUTER

**7. TITLE() METHOD**

title() method is used to convert given string into title case. Every first character of word of a given string is converted to title case.

>>> name1=“ cpe python syllabus“

>>> name1.title()

=== Cpe Python Syllabus

**8. SWAPCASE() METHOD**

swapcase() method is toggle the case. Meaning upper to lower and lower to upper case.

>>> name1=“ Computer “

>>> name1.swapcase()

=== cOMPUTER

- Every character case is changed

**9. LJUST() METHOD**

ljust() method is used to add spaces to the left side of the given string

>>> name1=“anand “

>>> name1.ljust(15)

=== ‘anand ’

Left side padded with spaces

Note: string length is 5 and 10 spaces added to the left side of string.

**10. RJUST() METHOD**

rjust() method is used to add spaces to the left

side of the given string

>>> name1=“anand “

>>> name1.rjust(15)

=== ‘ anand’

Left side padded with spaces

Note: string length is 5 and 10 spaces added to the left side of string.

**11. CENTER(WIDTH, FILLCHAR) METHOD**

The method center() returns centered in a string of length width. Padding is done using the specified fillchar. Default filler is a space. Centered string

>>> name=“Anand”

>>> name.center(36,”a”)

=== aaaaaaaaaaaaaaaAnandaaaaaaaaaaaaaaaa

>>> name.center(20,”\*”)

=== \*\*\*\*\*\*\*Anand\*\*\*\*\*\*\*\*

**12. ZFILL() METHOD**

zfill() method is used to fill the zero to a given string.

>>> name1=“ 123“

>>> name1.zfill(15)

=== ‘00123 ’

- Filling Zeros

**13. FIND() METHOD**

find() method is used to find a particular character or string in a given string.

>>> name1=“Internet"

>>> name1.find("e")

=== 3

e is present at 3rd location (first appearance) in a given string

**14. COUNT() METHOD**

count() method is used to the number of times

character or string appears in a given string.

>>> name1=“Internet “

>>> name1.count(“n”)

=== 2

2 times n appears in a given string

**15. STRATSWITH() METHOD**

startswith() method is used check string start with particular string or not

>>> name1=“Delhi“

>>> name1.startswith(“a”)

=== False

Given string not starting with “a”

**16. ENDSWITH() METHOD**

endswith() method is used check string ends with particular string or not

>>> name1=“Dairy“

>>> name1.endswith(“ry”)

=== True

Given string ends with “en”

**17. ISDIGIT() METHOD**

isdigit() method is used check string is digit (number) or not and returns Boolean value true or false.

>>> name2=“123”

>>> name2.isdigit()

=== True

>>> name1=“123keyboard“

>>> name1.isdigit()

=== False

Given string not number so false

**18. ISNUMERIC() METHOD**

isnumeric() is similar to isdigit() method and is used check string is digit (number) or not and returns Boolean value true or false.

>>> name2=“123”

>>> name2.isnumeric()

=== True

>>> name1=“123keyboard“

>>> name1.isnumeric()

=== False

Given string not number so false

**19. ISDECIMAL() METHOD**

isnumeric(),isdigit() and isdecimal() methods are used to check string is digit (number) or not and returns Boolean value true or false.

>>> name2=“123”

>>> name2.decimal()

=== True

>>> name1=“123keyboard“

>>> name1.isnumeric()

=== False

Given string not number so false

**20. ISALPHA() METHOD**

isalpha() method is used check string is digit or not and returns Boolean value true or false.

>>> name2=“123”

>>> name2.isalpha()

=== False

(Given string does not contain string )

>>> name1=“123computer“

>>> name1.isalpha()

=== False

(Given string is not a string it contains digits)

>>> name3=“Keynoard”

>>> Name3.isalpha()

=== True (It’s a string )

**21. ISALNUM() METHOD**

isalnum() method is used check string is alpha numeric string or not.

>>> name2=“123”

>>> name2.isalnum()

=== True

(True Given string is alpha numeric)

>>> name1=“123computer“

>>> name1.isalnum()

=== True

(True Given string is alpha numeric )

>>> name3=“Praveen”

>>> name3.isalnum()

=== True

(Given string is alpha numeric )

**22. ISLOWER() METHOD**

islower() method is used check string contains all lowercase letters or not, it returns true or false result.

>>> name2=“Anand”

>>> name2.islower()

=== False

(Given string is not lowercase string)

>>> name1=“anand“

>>> name1.islower()

True

(Given string is lower case string )

**23. ISUPPER() METHOD**

isupper() method is used check string contains all letters upper case or not, it returns true or false result.

>>> name2=“Anand”

>>> name2.isupper()

=== False

(Given string is not uppercase string)

>>> name1=“ANAND”

>>> name1.isupper()

=== True

(Given string is upper case string )

**24. ISSPACE() METHOD**

isspace() method is used check string contains space only or not.

>>> name2=“ ”

>>> name2.isspace()

=== True

(Given string contains space only )

>>> name1=“Anandalaya Anand “

>>> name1.isspace()

=== False

(Given string not containing space only)

**25. FIND() METHODS**

find() method is used to find a particular string (substring) in a given string.

>>> name=“Classic”

>>> name.find(“s”)

=== 3

The first position where “s” appears in the string.

**26. STR() METHOD**

str() method is used convert non string data into string type.

>>> str(576)

=== ‘576’

(576 is number converted to string)

**27. LEN() METHOD**

len() method is used get a length of string.

>>> len(“Naveen”)

=== 6

(Gives the string length)

**28. MAX() METHOD**

max() method is used get a max alphabet of

string.

>>> max(“Praveen”)

=== v

(Gives max character )

**29. MIN() METHOD**

min() method is used get a max alphabet of string.

>>> min(“Anand”)

=== A

(Gives min character A because it has ASCII Value 65 )

**30. SPLIT() METHOD**

split() method is used split a string.

>>> name=“Anandalaya NDDB Campus Anand”

>>> name.split()

=== [“Anandalaya”,“NDDB”,“ Campus”,“Anand”]

Split into several words or substrings

**30. SPLIT() METHOD**

split() method is used split a string according to delimiter.

>>> name=“Anandalaya NDDB Campus Anand”

>>> name.split(“Ca”)

=== [“Anandalaya NDDB”,“ mpus Anand”]

Split into several words or substrings according to delimiter

**31. INDEX() METHOD**

Same as find(), but raises an exception if str

not found.

>>> name="Sainik“

>>> name.index("a",3,5)

ValueError: substring not found

>>> name.index("a",1,5)

=== 1

(Character found, returning the position )

**32. ORD METHOD**

rd() method is used get a ASCII value for a character.

>>> ord(“a”)

=== 97

(97 is the ASCII value for character ‘a’ )

**33. CHR() METHOD**

chr() method is used get a character for an ASCII value.

>>> chr(97)

=== ‘a’

( ‘a’ ASCII value is 97)

**OBJECT ORIENTED PROGRAMMING**

**CMPE 103**

**MODULE 2 – FILE HANDLING IN PYTHON**

**Learning Goals/Objectives**

Be able to read, comprehend, trace, adapt and create Python code that:

• Opens a file

• Reads data from a file into a program

• Writes data from a program into a file

• Appends data from a program into a file

• Closes a file

**WHAT IS FILE?**

File is a named location on disk to store related information. It is used to permanently store data in a non-volatile memory (e.g., hard disk).

Since Random Access Memory (RAM) is volatile which loses its data when computer is turned off, we use files for future use of the data.

When we want to read from or write to a file, we need to open it first. When we are done, it needs to be closed, so that resources that are tied with the file are freed.

Hence, in Python, a file operation takes place in the following order.

1. Open a file

2. Read or write (perform operation)

3. Close the file

**FILE TYPES?**

In Python, there are two types of files. They are:

❑ Text files

❑ Binary files

Text files store the data in the form of characters. For example, if we store employee name “Jerald”, it will be stored as 6 characters and the employee salary 8900.75 is stored as 7 characters.

**TEXT FILES**

are used to store characters or strings.

**BINARY FILES**

store entire data in the form of bytes, i.e., a group of 8 bits each. For example, a character is stored as a byte and an integer is stored in the form of 8 bytes (on a 64-bit machine). When the data is retrieved from the binary file, the programmer can retrieve the data as bytes.

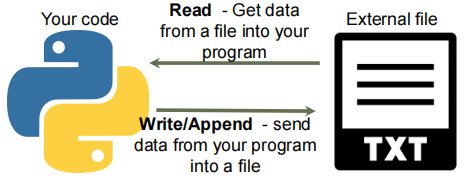
**Binary files** can be used to store text, images, audio and video. Image files are generally available in .jpg, .gif or .png formats.

We cannot use text files to store images as the images do not

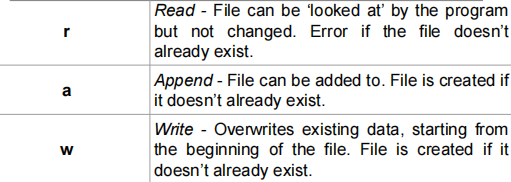
contain characters.

**WHAT IS FILE HANDLING?**

Computer programs can import data from and export data to files outside the code.



**FILE PERMISSIONS**



***READ FROM A FILE***

**Read From A File - The Algorithm**

1. Connect to and open the file.

a. Give the file name and path

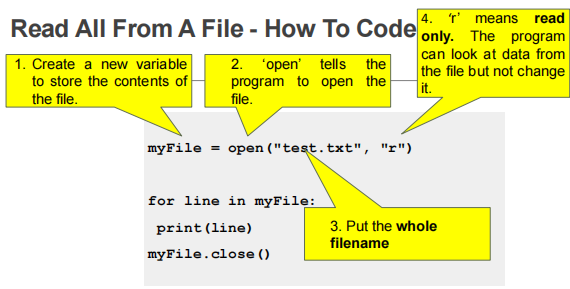
b. Set the permissions for opening

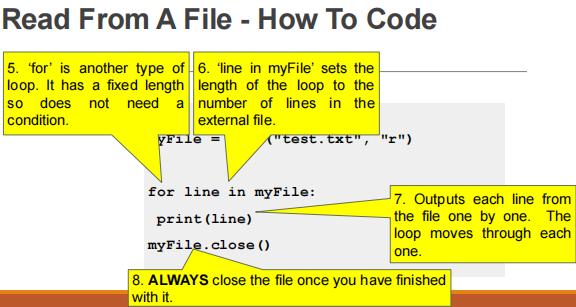
2. Read the contents into a variable

3. Output the variable.

4. Close the file.

**READ ALL FROM A FILE – How To Code**





**READ FUNCTIONS**

**read(n)**

– Read at most n characters form the file. Reads till end of file if it is negative or None.

For example:

*f = open(“text.txt)*

*print (f.read()) # Reads the entire file*

*f.close()*

*f = open(“text.txt)*

*print(f.read(5)) # Reads the first 5 characters of the file*

*f.close()*

**readline(n=-1)**

– Read and return one line from the file. Reads in at most n bytes if specified.

For example:

*f = open(“text.txt)*

*print(f.readline())#Reads the first line of the file*

*f.close()*

**readlines(n=-1)**

– Read and return a list of lines from the file. Reads in at most n bytes/characters if specified.

For example:

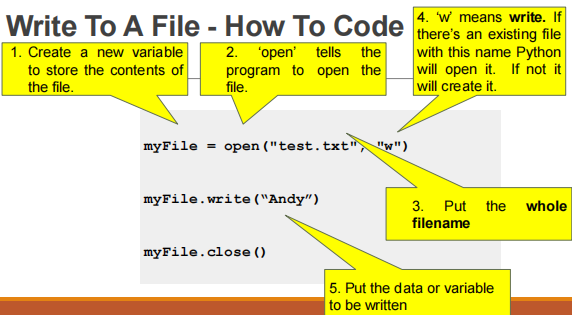
*f = open(“text.txt”)*

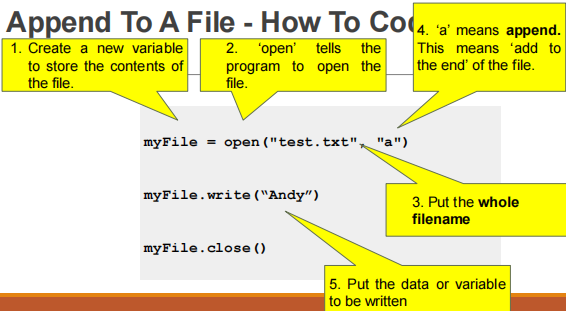
*data = f.readlines()*

*for line in data:*

*print(line)*

***WRITE & APPEND TO A FILE***





**OBJECT ORIENTED PROGRAMMING**

**CMPE 103**

**MODULE 3 - EXCEPTION HANDLING**

**Table of Contents:**

1. What are Exceptions
2. Example of Exceptions cases
3. Types of Exceptions
4. Handling an Unchecked Exceptions
5. Examples

**WHAT IS AN EXCEPTION?**

An Exception is an error that happens during execution of a program. When that error occurs, Python generate an exception that can be handled, which avoids your program to crash.

**WHY USE EXCEPTIONS?**

Exceptions are convenient in many ways for handling errors and special conditions in a program. When you think that you have a code which can produce an error then you can use exception handling.

**RAISING AN EXCEPTION**

You can raise an exception in you own program by using the raise exception statement. Raising an exception breaks current code execution and returns the exception back until it is handled.

**Example for an Exception Cases:**

IOError - If the file cannot be opened.

ImportError - If phyton cannot find the module.

ValueError - Raised when a function receives an argument that has the right type but an inappropriate value.

**TYPES OF EXEMPTIONS**

**Checked (Compile-time)**

- Checked exceptions are checked at compile-time.

- Exceptions are “checked” because they are subject to the “catch” or “specify requirement” otherwise, the program code will not compile.

Examples: Invalid Syntax, Incorrect statements.

**Unchecked (Runtime)**

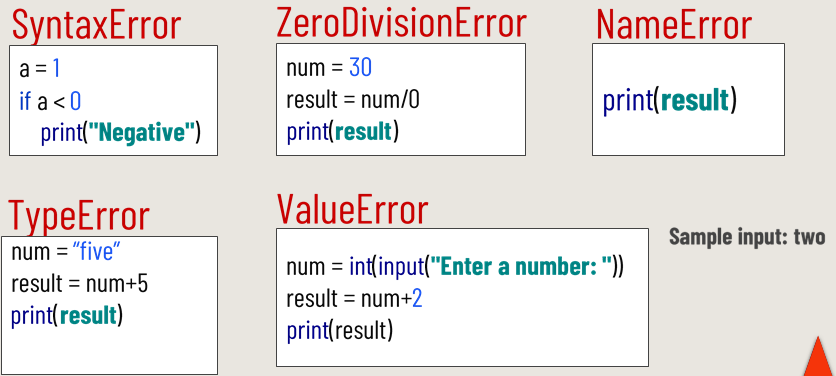
- Unchecked exceptions are not checked at compile time.

- Errors are not subject to the “catch” or “specify requirement”.

- It occurs during Runtime

Examples: incalid input, incalid arithmetic operations, number divided by zero.

**PYTHON UNCHECKED EXCEPTIONS**



**Handling Unchecked Exception**

* **try** is used to test a block of code for errors.
* **except** is used to handle errors.
* **finally** is used to execute block of code regardless of the result of the try and except blocks.

**Exception Handling - Structure**

*try:*

*#Try block of Statements*

*except<<Exception Class>>*

*#Exception handling statements*

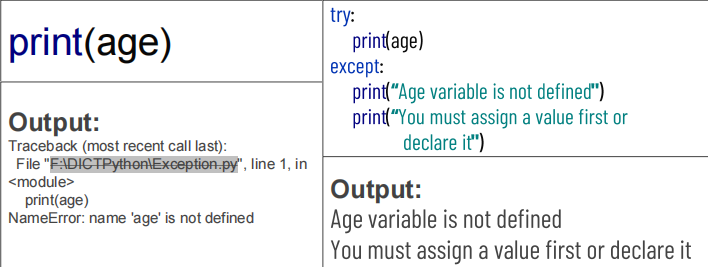
*else:*

*#Else block statements*

*finally:*

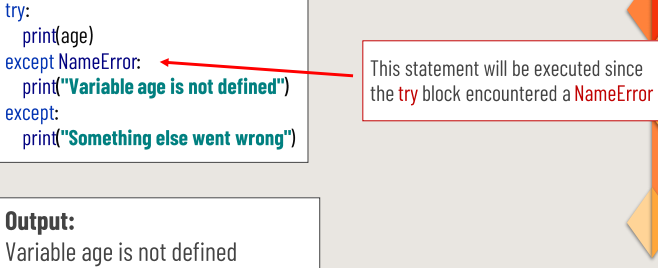
*#Finally block of statements*

Using try…catch



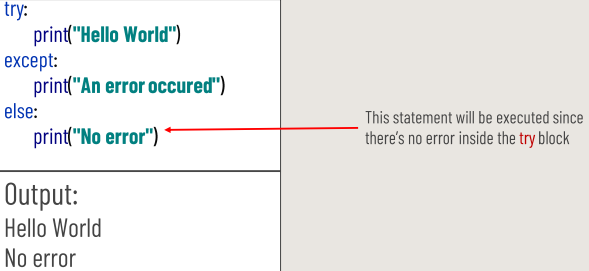
Print a specific error message if the try block raises a

**NAMEERROR**



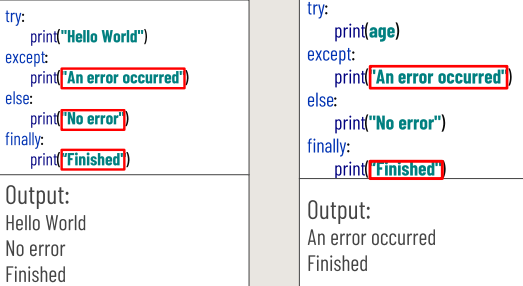
**ELSE**

**else** is used to define a block of code to be executed if no errors were raised



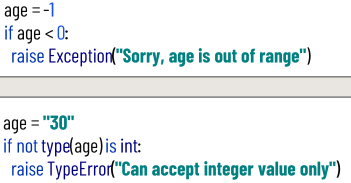
**FINALLY**

**finally** block will be executed regardless if the **try** block raises an error or not.



**RAISE AN EXECPTION**

raise is used to throw an exception if a condition occurs.



**EXCEPTION HANDLING – Example 01**

x = 10

y = 5

try:

result = x // y

print (“Yeah ! Your answer is : ”, result)

except ZeroDivisionError:

print(“Sorry! You are dividing by zero ”)

finally:

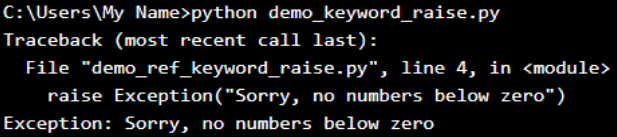
print (“End of Program”)

**EXCEPTION HANDLING – Example 02**

x = -1

if x < 0

raise Exception (“Sorry, no numbers below zero”)

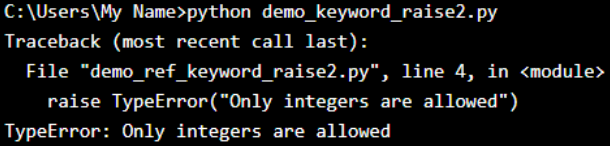


**EXCEPTION HANDLING – Example 03**

x = “hello”

if not type (x) is int:

raise TypeError(“Only integers are allowes”)



**EXCEPTION HANDLING – Exemption 04**

try:

x = int(input(“Enter the value of X : ”))

y = int(input(“Enter the value of Y : ”))

result = x/y

print (“ Answer : ”, result)

except ZeroDivisionError:

print(“Sorry ! You are dividing by zero ”)

except ValueError:

print(“Sorry ! You are dividing by zero ”)

finally:

print(“End of Program”)

**EXCEPTION HANDLING – Example 05**

try:

f = open(“demo.txt”)

f.write(“Python Programming”)

except:

print(“Something went wrong when writing to the file”)

finally:

f.close()

**EXCEPTION HANDLING – Example 06**

try:

a = 3

if a < 4 :

b = a / (a-3)

print ( “Value of b = ”, b)

except (ZeroDivisionError, NameError) :

print( “Error Occurred and Handled” )

except:

print ( “Unknown Exception” )

**EXCEPTION HANDLING – Example 07**

try:

x = int(input(