Name : Sufyan Ahmad

ID : 13062

Subject : Data Sciences

Instructor : M.Ayub Khan

Degree : BS.SE

Q1. What type of errors do occur in Python, write the a program with different types of errors as well as write separate correction code?

Answer :

Errors or mistakes in a phython program are often reffered to as bugs.there are almost always the fault of the programmer.the process of finding errors is called debugging .Errors can be classidied into there major groups:

- try errors
- except errors
- finally errors

The **try** block lets you test a block of code for errors in phyton.
The **except** block lets you handle the error in phyton .
The **finally** block lets you execute code, regardless of the result to try- and except blocks.

- **Try Errors:**

When an error occurs, or exception as we call it, Python will normally stop and generate an error message.These exceptions can be handled using the try statement:

Program:

The try block program will generate an exception, because x is not defined:

#The try block will generate an error, because x is not defined:

try:

```
  print(x)
except:
  print("An exception occurred")
```

Result : An exception occurred

Since the try block raises an error, the except block will be executed.

- **Except Error:**

You can define as many exception blocks as you want in phyton , if you want to execute a special block of code for a special kind of error:

Program :

Print one message if the try block raises a NameError and another for other errors:

```
#The try block will generate a NameError, because x is not defined:
try:
  print(x)
except NameError:
  print("Variable x is not defined")
except:
  print("Something else went wrong")
```

Result : Variable x is not defined

Finally Error:

The finally block, if specified, will be executed regardless if the try block raises an error or not.

#The finally block gets executed no matter if the try block raises any errors or not:

```
try:
  print(x)
except:
  print("Something went wrong")
finally:
  print("The 'try except' is finished")
```

Result :

Something went wrong

The 'try except' is finished

Q2. What are Boolean String test, write the code for each Boolean string test

code?

Answer:

A Boolean string test in Python can be tested for truth value.The return type will be in Boolean value (True or False).

Let’s make an program, by first create a new variable and give it a value.

```
my_string = "Hello World"

my_string.isalnum()          #check if all char are numbers
my_string.isalpha()          #check if all char in the string are alphabetic
my_string.isdigit()          #test if string contains digits
my_string.istitle()          #test if string contains title words
my_string.isupper()          #test if string contains upper case
my_string.islower()          #test if string contains lower case
my_string.isspace()          #test if string contains spaces
my_string.endswith('d')          #test if string endswith a d
my_string.startswith('H')    #test if string startswith H
```

To see what the return value (True or False) will be, simply print it out.

```
my_string="Hello World"
print my_string.isalnum()          #Falseprint my_string.isalpha()
      #Falseprint my_string.isdigit()          #Falseprint
my_string.istitle()          #Trueprint my_string.isupper()
      #Falseprint my_string.islower()          #Falseprint
my_string.isspace()          #Falseprint my_string.endswith('d')
      #Trueprint my_string.startswith('H')          #True
```

Q3. What is formatting string input mean in Python, write a program in which

formatting string input is used?

Answer:

The string formatting input means to create new, formatted strings. The "%" operator is used to format a set of variables enclosed in a "tuple" (a fixed size list), together with a format string, which to contains normal text together with "argument specifiers", special symbols like "%s" and "%d".

The format() method allows you to format input selected parts of a string.

Sometimes there are parts of a text that you do not control input, maybe they come from a database, or user input?

To control such values, add placeholders (curly brackets {}) in the text, and run the values through the format() input method:

Program :

Add a placeholder where you want to display the price:

```
price = 49

txt = "The price is {} dollars"

print(txt.format(price))
```

Result :

The price is 49 dollars