Name M.Mubeen Alam

Id # 6906

Semester #7th

Submitted to Sir Ayub

QUESTIONS#1: What type of errors do occur in Python, write the a program with different types of errors as well as write separate correction code?

Ans: Errors

Blunders or errors in a program are frequently alluded to as bugs. They are quite often the shortcoming of the software engineer. The way toward finding and wiping out blunders is called troubleshooting. Mistakes can be arranged into three significant gatherings:

1. **Syntax errors**
2. **Runtime Errors**
3. **Logical Errors**

1 : Syntax errors

Syntax errors are those in which Python will discover these sorts of blunders when it attempts to parse your program, and exit with a mistake message without running anything. Punctuation blunders are botches in the utilization of the Python language and are closely resembling spelling or syntax botches in a language like English: for instance, the sentence Would you some tea??? does not bode well .it is feeling the loss of an action word.

Common Python syntax errors include:

- **It's leaving out a keyword**
- **It's putting a keyword in the wrong place**
- **It's leaving out a symbol, such as a colon(;) comma(,) or brackets().**
- **It's misspelling a keyword.**
- **It's incorrect indentation.**
- **It's empty block.**

Its best to tell where is error is located python detect and solve this error.,

Some major and imp example about python.

My function(x, y):

```
    Return    x + y

else:

    print("Hello!")

if mark >= 50

    print("You passed!")

if arriving:

    print("Hi!")esle:

    print("Bye!")

if flag:print("Flag is set!")
```

2: RunTime errors

On the off chance that the system is linguistically correct - that is, relieved of language structure errors - it will be managed by a Python translator. In any case, the system may run out of blue during the execution of the accident that it encounters a runtime error - a problem that was not detected when the system was disassembled, but may have been turned on when a particular line was created. At the point where the system stops looking at the runtime error, it means it's dead. Some examples of Python running time errors:

- **divisible by zero**
- **Doing work with non-conforming types**
- **uses an anonymous identifier**
- **access the list item, dictionary value or attribute attribute**
- **trying to access a missing file**

Runtime errors tend to fall if you do not consider all possible variable values that may contain, especially when processing user input. You should always try to add checks to your code to make sure it handles the best inputs and cases better. We will look at this in more detail in a chapter on different management

3: Logical Errors

Logical errors are very difficult to fix. They occur when the system runs without crashing, but produces a negative result. An error is caused by an error in program logic. You will not receive an error message, because no syntax or runtime error has occurred. You will have to figure out the problem yourself by reviewing all the relevant parts of your code - although some tools may flag suspicious code that looks like it might cause unexpected behavior.

Uses a different wrong name

- Placing the block at the wrong level
- It uses wide segmentation instead of segmentation
- Getting the user layer first
- Error on bond statement
- It's one-by-one, with some numerical errors

If you miss the identifier name, you may get a runtime error or a logical error, depending on if the replace term is defined.

```
x = float (input ('Enter number:'))

y = float (input ('Enter number:'))



z = x + y / 2

print ('The average of two numbers you have entered is:', z)
```

The above example should measure the average of the two values the user entered. However, because of the arithmetic sequence of operations (classification tested before addition) the system will not provide the correct answer:

QUESTION#2: What are Boolean String test, write the code for each Boolean string test code?

ANS: Boolean values are the two (True or False) objects . They are used to represent truth values (the values that can also be considered as false or true). In numeric contexts (for example, when used as the argument to anarithmetic operator), they behave like the integers 0 and 1, respectively.The built-in function bool() can be used to cast any value to a Boolean,if the value can be interpreted as a truth valueThey are written as False and True, respectively Boolean Values.

In every programming you often need to know if an expression is true and false.

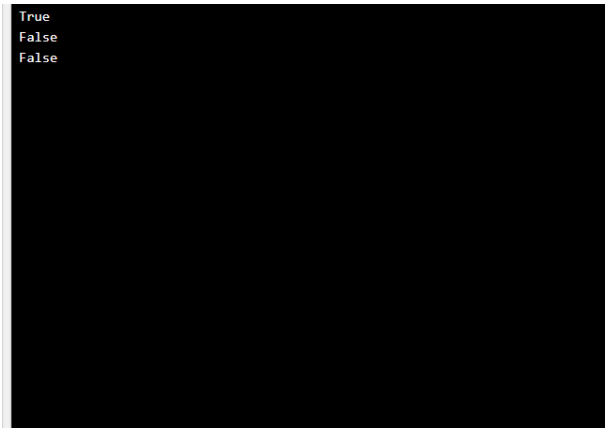
You can evaluate any expression in Python aud get one of two answers True and False.

One string can be used for testing in truth value in python. And the return value will Boolean value may be its (True 0R False) so, we create first a new variable and give some value in it. Let's make an example.

Example

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```



Example

The following will return True:

```
bool("abc")
bool(123)
bool(["apple", "cherry", "banana"])
```

```
print(bool("abc"))
print(bool(123))
print(bool(["apple", "cherry", "banana"]))
```

Example

The following will return False:

```
bool(False)
bool(None)
bool(0)
bool("")
bool(())
bool([])
bool({})
```

```
print(bool(False))
print(bool(None))
print(bool(0))
print(bool(""))
print(bool(()))
print(bool([]))
print(bool({}))
```

QUESTION#3: What is formatting string input mean in Python, write a program in which formatting string input is used?

Answer : Formatting string input is formatting the input we receive from the user. There are different built in methods that can be applied to format the input from the user. As the input is always of the string type, this is why it is called formatting string input.

1. Upper()
2. Lower()
3. Capitalize()

The input() pauses program execution to allow the user to type in a line of input from the keyboard. Once the user presses the Enter key all characters typed are read and returned as a string and the program is following.

The code with the respective output is attached

upper()

```
value = input("enter your city name: ").upper()
```

```
print("name is : ",value )
```

```
enter your name: abc
name is :  ABC
```

We can see that the user gave the input in the lower case, however the upper method formats the string input to uppercase.

lower()

```
value = input("enter your city name: ").lower()

print("name is : ",value )
```

```
enter your name: XYZ
name is :  xyz
```

capitalize()

```
value = input("enter name of the course: ").capitalize()

print("name is : ",value )
```

```
enter name of the course: data science
name is :  Data science
```

This method formats the input such that the first letter of each word is converted to uppercase.

----End-----