

Department of Computer Science

Final Exam Summer 2020

Subject: Data Sciences

Time: 9 Am to 12:00 Am

BS (CS,SE)

Instructor: M.Ayub Khan

---

**Note:**

*At the top of the answer sheet there must be the ID, Name and semester of the concerned Student.*

*Students must have to provide the output of their respective programs. Students have same answers or programs will be considered fail. Programs in Python and codes should be explained clearly.*

*As this assignment is online so incase of any ambiguity my Whatsapp no. is 03449121116.*

<b>Name : Alamgir Khan</b>	<b>Id: 13379</b>
<b>BS(SE)</b>	<b>Course : Data Science</b>
<b>Section (A) 8<sup>th</sup> Semester</b>	<b>Date 23/09/20</b>

- Q1. a. What are variables in python explain with help of Python coded examples?  
b. What are the rules to define a variable in python?

**Answer # 01**

**Part(a)**

Variables are containers for storing data values. Unlike other programming languages python has no command for declaring a variable. A variable is created the moment you first assign a value to it. Variables do not need to be declared with any particular type and can even change type after they have been set.

```
CAWindows\python.exe
>>> x = 12
>>> print(x) # x is of type int
12
>>> x = "Alam"
>>> print(x) # x is of type string
Alam
>>>
>>> #String variable can be declared either by using single or duple quotes.
>>> x = "alam"
>>> print(x) # double quotes
alam
>>> x = 'alam'
>>> print(x) # single quotes
alam
>>>
>>> #Assign value to multiple variables
>>> x, y, z, = "Alam", "shaid", "junaid"
>>> print(x)
Alam
>>> print(y)
shaid
>>> print(z)
junaid
>>> #Datatypes used in python
>>> x = "Hello World"
>>>
>>> #display x:
>>> print(x)
Hello World
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'str'>
>>>
>>> #datatypes no 2.....
>>> x = 20
>>>
>>> #display x:
>>> print(x)
20
>>>
>>> #display the data type of x:
```

## Part(b)

### Rules to define a variable in python.

A variable can have a short name (like x and y) or more descriptive name (age, carname, total\_number). Rules for python variables are following:

A variable name must start with a letter or the underscore character.

A variable name cannot start with a number.

A variable name can only contain alpha-numeric character and underscore (A-z, 0-9 and \_).

Variable names are case-sensitive (age, Age and AGE are three different variables)

Q2. a. What are data types, how many data types are used in python explain with the help of Python coded examples ?

b. Write a program in python in which integer value is changed in to string data type as well as explain in detail.

**Answer # 02**

**Part(a)**

### Python Data Types:

Data types are the classification or categorization of data items. Data types represent a kind of value which determines what operations can be performed on that data. In programming, data types is in important concept. Variables can store data of different types and different types can do different things. Python has the following data types.

```
C:\Windows\py.exe
>>> #Datatypes used in python
>>> x = "Hello World"
>>>
>>> #display x:
>>> print(x)
Hello World
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'str'>
>>>
>>> #datatypes no 2.....
>>> x = 20
>>>
>>> #display x:
>>> print(x)
20
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'int'>
>>>
>>>
>>> datatypes no 3.....
File "<stdin>", line 1
datatypes no 3.....
^
SyntaxError: invalid syntax
>>> #datatypes no 3.....
>>> x = 20.5
>>>
>>> #display x:
>>> print(x)
20.5
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'float'>
>>>
>>>
>>> #datatypes no 4.....
>>> x = 1j
```

```
C:\Windows\py.exe
>>> #datatypes no 4.....
>>> x = 1j
>>>
>>> #display x:
>>> print(x)
1j
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'complex'>
>>>
>>>
>>> #datatypes no 5.....
>>> x = ["apple", "banana", "cherry"]
>>>
>>> #display x:
>>> print(x)
['apple', 'banana', 'cherry']
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'list'>
>>>
>>>
>>> #datatypes no 6....
>>> x = ("apple", "banana", "cherry")
>>>
>>> #display x:
>>> print(x)
('apple', 'banana', 'cherry')
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'tuple'>
>>>
>>>
>>> #datatypes no 7....
>>> x = range(6)
>>>
>>> #display x:
>>> print(x)
```

```
C:\Windows\py.exe
>>> #datatypes no 7....
>>> x = range(6)
>>>
>>> #display x:
>>> print(x)
range(0, 6)
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'range'>
>>>
>>>
>>> #datatypes no 8....
>>> x = {"name": "John", "age": 36}
>>>
>>> #display x:
>>> print(x)
{'name': 'John', 'age': 36}
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'dict'>
>>>
>>>
>>> #datatypes no 9....
>>> x = {"apple", "banana", "cherry"}
>>>
>>> #display x:
>>> print(x)
{'cherry', 'banana', 'apple'}
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'set'>
>>>
>>>
>>> #datatypes no 10.....
>>> x = frozenset({"apple", "banana", "cherry"})
>>>
>>> #display x:
>>> print(x)
frozenset({'apple', 'banana', 'cherry'})
>>>
```

```
C:\Windows\py.exe
>>> #datatypes no 10.....
>>> x = frozenset({"apple", "banana", "cherry"})
>>>
>>> #display x:
>>> print(x)
frozenset({'cherry', 'banana', 'apple'})
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'frozenset'>
>>>
>>>
>>> #datatypes no 11.....
>>> x = True
>>>
>>> #display x:
>>> print(x)
True
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'bool'>
>>>
>>>
>>> #datatypes no 12.....
>>> x = b"Hello"
>>>
>>> #display x:
>>> print(x)
b'Hello'
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'bytes'>
>>>
>>>
>>> #datatypes no 13.....
>>> x = bytearray(5)
>>>
>>> #display x:
>>> print(x)
bytearray(b'\x00\x00\x00\x00\x00')
```

```
C:\Windows\py.exe
>>> #datatypes no 13.....
>>> x = bytearray(5)
>>>
>>> #display x:
>>> print(x)
bytearray(b'\x00\x00\x00\x00\x00')
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'bytearray'>
>>>
>>>
>>> #datatypes no 14.....
>>> x = memoryview(bytes(5))
>>>
>>> #display x:
>>> print(x)
<memory at 0x0000027F3DDC4580>
>>>
>>> #display the data type of x:
>>> print(type(x))
<class 'memoryview'>
>>>
```

Q3. Why print() and type functions are used in python explain with the help of python coded examples for each function and explain in detail as well ?

### Answer # 03

#### Print() Function:

The print() function print the specified message to the screen or other standard output device. The message can be a string or any other object, the object will be converted into a string before written to the screen.

Any object and as many as you like. Will be converted to string before printed.

Optional. Specify how to separate the objects, if there is more than one.

```
C:\Windows\py.exe
>>> #Python print() function
>>> #Print message on to the screen
>>> print("Hello World")
Hello World
>>> # Print more than one object
>>> print("Hello", "how are you?")
Hello how are you?
>>>
>>> #Print a tuple
>>> x = ("apple", "banana", "cherry")
>>> print(x)
('apple', 'banana', 'cherry')
>>>
>>> #Print two message and specify the separator
>>> print("Hello", "how are you?", sep="--")
Hello--how are you?
>>>
```


## Type() Function:

The type() function returns class type of the argument(object) passed as parameter. Type() function is mostly used for debugging purpose. Two different types of argument can be passed to type() function, single and three argument. If single argument type(name, bases, dict) is passed, it returns a new type object. The type function give the type of object.

```
C:\Windows\py.exe
>>> # Python3 simple code to explain
>>> # the type() function
>>> print(type([]) is list)
True
>>>
>>> print(type([]) is not list)
False
>>>
>>> print(type(()) is tuple)
True
>>>
>>> print(type({}) is dict)
True
>>>
>>> print(type({}) is not list)
True
>>>
>>>
>>>
```

Q4. How addition operator is used to update the values of variables explain with the help of Python coded example as well as explain the program?

This means get the current value of x, add five, and then update x with the new value. The new value of x is the old value of x plus 1. Although this assignment statement may look a bit strange, remember that executing assignment is a two-step process. First, evaluate the right-hand side expression. Second, let the variable name on the left-hand side refer to this new resulting object. The fact that x appears on both sides does not matter. The semantics of the assignment statement makes sure that there is no confusion as to the result. The visualizer makes this very clear



```
CIWindows\py.exe
>>> x = 8 #initialize x
>>> x = x + 5 #update x
>>> print(x)
13
>>>
>>>
>>> # another example.....
>>> x = 12
>>> x = x - 3
>>> x = x + 5
>>> x = x * 4
>>> print(x)
56
>>>
>>>
```

Q5. What type of errors do occur in Python, write the a program with different types of errors as well as write separate correction code in python as well as explain the errors?

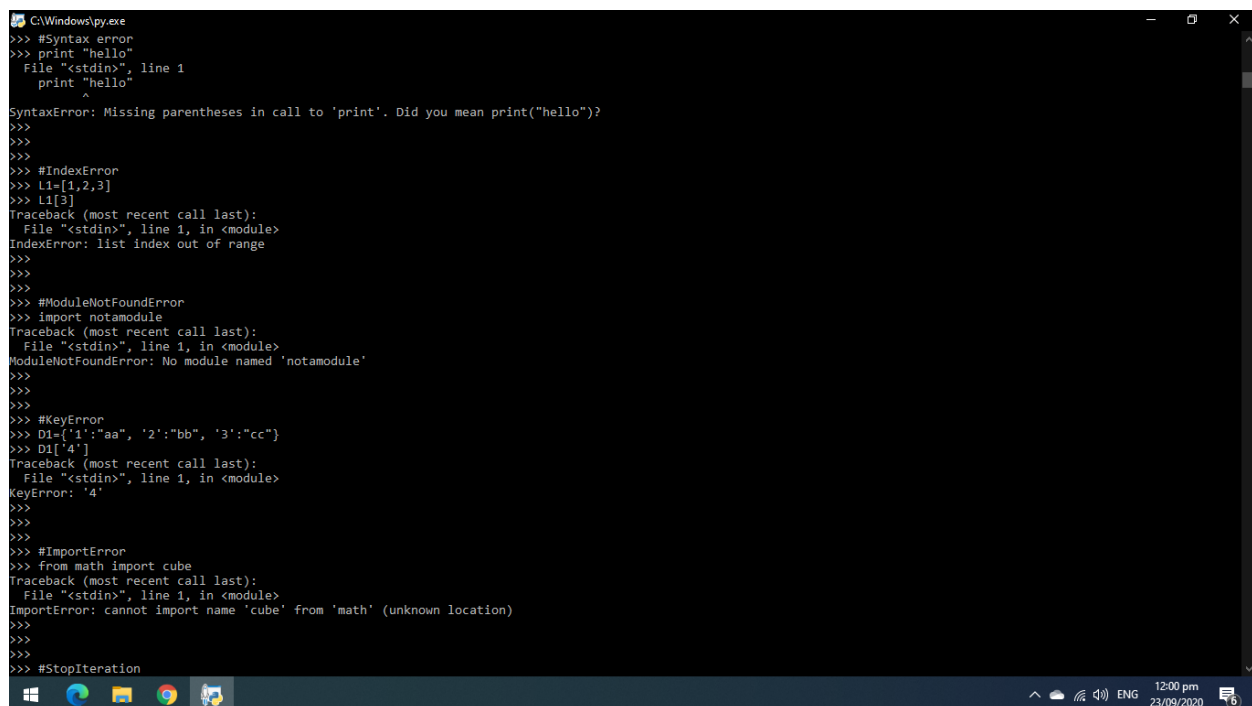
### Answer # 05

The most common reason of an error in a Python program is when a certain statement is not in accordance with the prescribed usage. Such an error is called a syntax error. The Python interpreter immediately reports it, usually along with the reason.

In Python 3.x, print is a built-in function and requires parentheses. The statement above violates this usage and hence syntax error is displayed.

Many times though, a program results in an error after it is run even if it doesn't have any syntax error. Such an error is a runtime error, called an exception. A number of built-in exceptions are defined in the Python library. Let's see some common error types.

- 1) Index Error: is thrown when trying to access an item at an invalid index.
- 2) ModuleNotFoundError is thrown when a module could not be found.
- 3) KeyError is thrown when a key is not found.
- 4) ImportError is thrown when a specified function can not be found.
- 5) TypeError is thrown when an operation or function is applied to an object of an inappropriate type.
- 6) NameError is thrown when an object could not be found.
- 7) ZeroDivisionError is thrown when the second operator in the division is zero.



```
C:\Windows\py.exe
>>> #Syntax error
>>> print "hello"
  File "<stdin>", line 1
    print "hello"
    ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print("hello")?
>>>
>>>
>>> #IndexError
>>> l1=[1,2,3]
>>> l1[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>>
>>>
>>> #ModuleNotFoundError
>>> import notamodule
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'notamodule'
>>>
>>>
>>> #KeyError
>>> D1={'1':"aa", '2':"bb", '3':"cc"}
>>> D1['4']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: '4'
>>>
>>>
>>> #ImportError
>>> from math import cube
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'cube' from 'math' (unknown location)
>>>
>>>
>>> #StopIteration
```



```
C:\Windows\py.exe
>>> #TypeError
>>> '2'+2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
>>> #NameError
>>> age
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'age' is not defined
>>>
>>>
>>> #ZeroDivisionError
>>> x=100/0
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero
>>>
>>>
>>> #KeyboardInterrupt
>>> name=input('enter your name')
enter your nameenter your name^c
>>>
_
```