**Name syed Muhammad salman khan**

**Id: 13662**

**Program: BS(software engineering)**

**Subject: Data science**

**Semester: Final Exam Summer 2020**

**Submitted to: M.Ayub khan**

Q1. a. What are variables in python explain with help of Python coded examples?

**Variables and types**: Python is completely object oriented, and not "statically typed". You do not need to declare variables before using them, or declare their type. Every variable in Python is an object.

**Numbers:**

Python supports two types of numbers - integers and floating point numbers.

To define an **integer**, we use the following syntax:

**Example:**

```
myint = 7
print(myint)
```

***output:***

```
7
```

To define a **floating point** number, we may use one of the following notations:

**Example:**

```
myfloat = 7.0
print(myfloat)
myfloat = float(7)
print(myfloat)
```

***output:***

```
7.0
7.0
```

**String:** The difference between the two is that using double quotes makes it easy to include apostrophes (whereas these would terminate the string if using single quotes)

**Example :**

```
mystring = "Don't worry about apostrophes"
print(mystring)
```

***output:***

```
Don't worry about apostrophes
```

b. What are the rules to define a variable in python?

## Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables:

- 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 characters and underscores (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 ?

**Python Data Types**: Every value in Python has a data type. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

There are various data types in Python. Some of the important types are listed below.

Python Numbers Integers, floating point numbers and complex numbers fall under Python numbers category. They are defined as int, float and complex classes in Python.

We can use the type() function to know which class a variable or a value belongs to. Similarly, the isinstance() function is used to check if an object belongs to a particular class.

**Example**:

```
a = 5
print(a, "is of type", type(a))

a = 2.0
print(a, "is of type", type(a))

a = 1+2j
print(a, "is complex number?", isinstance(1+2j,complex))
```

**output:**

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
```

```
(1+2j) is complex number? True
```

**Python list** : List is an ordered sequence of items. It is one of the most used datatype in Python and is very flexible. All the items in a list do not need to be of the same type.

Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [].

**Example :**

```
a = [5,10,15,20,25,30,35,40]

# a[2] = 15
print("a[2] = ", a[2])

# a[0:3] = [5, 10, 15]
print("a[0:3] = ", a[0:3])

# a[5:] = [30, 35, 40]
print("a[5:] = ", a[5:])
```

**output :**

```
a[2] =  15
a[0:3] =  [5, 10, 15]
a[5:] =  [30, 35, 40]
```

## python Tuple:

Tuple is an ordered sequence of items same as a list. The only difference is that tuples are immutable. Tuples once created cannot be modified.

Tuples are used to write-protect data and are usually faster than lists as they cannot change dynamically.

It is defined within parentheses () where items are separated by commas.

**Example :**

```
t = (5,'program', 1+3j)

# t[1] = 'program'
print("t[1] = ", t[1])

# t[0:3] = (5, 'program', (1+3j))
print("t[0:3] = ", t[0:3])

# Generates error
# Tuples are immutable
t[0] = 10
```

**Output**:

```
t[1] =  program
```

```
t[0:3] =  (5, 'program', (1+3j))
Traceback (most recent call last):
  File "test.py", line 11, in <module>
    t[0] = 10
TypeError: 'tuple' object does not support item assignment
```

## Python string :

String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, ''' or """.

## Example :

```
s = "This is a string"
print(s)
s = '''A multiline string'''
print(s)
```

## output:

```
This is a string
A multiline
String
```

## Python Dictionary

Dictionary is an unordered collection of key-value pairs.

It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value.

In Python, dictionaries are defined within braces {} with each item being a pair in the form key:value. Key and value can be of any type.

## Example :

```
d = {1:'value','key':2}
print(type(d))

print("d[1] = ", d[1]);

print("d['key'] = ", d['key']);

# Generates error
print("d[2] = ", d[2]);
```

## output:

```
<class 'dict'>
d[1] =  value
```

```
d['key'] =  2
Traceback (most recent call last):
  File "<string>", line 9, in <module>
KeyError: 2
```

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

**Python Int to String**

We can do this by using the str()method:

**Example:**

```
raw_user_age = input("What is your age?")
user_age = int(raw_user_age)
as_string = str(user_age)

print("Your age is: " + as_string)
```

**output:**

```
What is your age?
12
Your age is: 12
```

*We've successfully converted our integer to a string using str(). Both values are now strings, which means that we can now concatenate the message Your age    is with the user's age.*

**Conclusion:**

The int() method is used to convert a string to an integer in Python. This can be useful if you need to store a value as an integer or perform mathematical operations on a value stored as a string. The str() method is used to convert an integer to a string.

Q 3 :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 ?

**print**: The print() function prints 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.

**Example:**

```
x = ("apple", "banana", "cherry")
print(x)
```

**Output:**

```
apple banana cherry
```

**Type ():**

The type() function returns the type of the specified object
Python type is a built in function.
There are two variants of type ().
Type(object):returns the type of object
Type (name,bases,dict):creates data class dynamically with the given arguments
The type() is used to print the type of function parameters for debugging purposes.

**Example:**

```
a = ('apple', 'banana', 'cherry')
b = "Hello World"
c = 33

x = type(a)
y = type(b)
z = type(c)
```

**output:**

```
<class 'tuple'>
<class 'str'>
<class 'int'>
```

---

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?

**Example :**

```
Add_three=10+20+30

Print(add_three)
```

**Output:**

```
60
```

And if we want to update the value of Add_three we can write the code as follow

```
Add_three=10+20+30+40

Print(Add_three)
```

**Output:**

```
100
```

And if we want to make it double the value of Add_three then we will write code just like that

```
Add_six=Add_three+Add_three

Print(add_six)
```

**Output:**

```
200
```

**Example 2:**

```
Name=python

Greeting=”I love ”+Name

Print(Greeting)
```

**Output***:*

```
I love python
```

---

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?

Syntax errors – usually the easiest to spot, syntax errors occur when you make a typo. Not ending an if statement with the colon is an example of an syntax error, as is misspelling a Python keyword (e.g. using whille instead of while). Syntax error usually appear at compile time and are reported by the interpreter. Here is an example of a syntax error:

**Example of syntax error**:

```
x = int(input('Enter a number: '))

whille x%2 == 0:
   print('You have entered an even number.')
else:
   print ('You have entered an odd number.')
```

**output:**

```
C:Python34Scripts>python error.py
  File "error.py", line 3
```

```
    whille x%2 == 0:
           ^
SyntaxError: invalid syntax
```

### Example of syntax error free code :

```
x = int(input('Enter a number: '))

while x%2 == 0:
    print('You have entered an even number.')
else:
    print ('You have entered an odd number.')
```

### output:

```
12
You have entered an even number
```

2. Logical errors – also called semantic errors, logical errors cause the program to behave incorrectly, but they do not usually crash the program. Unlike a program with syntax errors, a program with logic errors can be run, but it does not operate as intended. Consider the following example of an logical error:

### Example of logical code error:

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

z = x+y/2
print ('The average of the two numbers you have entered is:',z)
```

### output:

```
>>>
Enter a number: 3
Enter a number: 4
The average of the two numbers you have entered is: 5.0
>>>
```

**To rectify this problem, we will simply add the parentheses: z = (x+y)/2**

### Example of logical code free from errors:

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

z = (x+y)/2
print ('The average of the two numbers you have entered is:',z)
```

### output:

```
>>>
Enter a number: 3
Enter a number: 4
```

```
The average of the two numbers you have entered is: 3.5
>>>
```