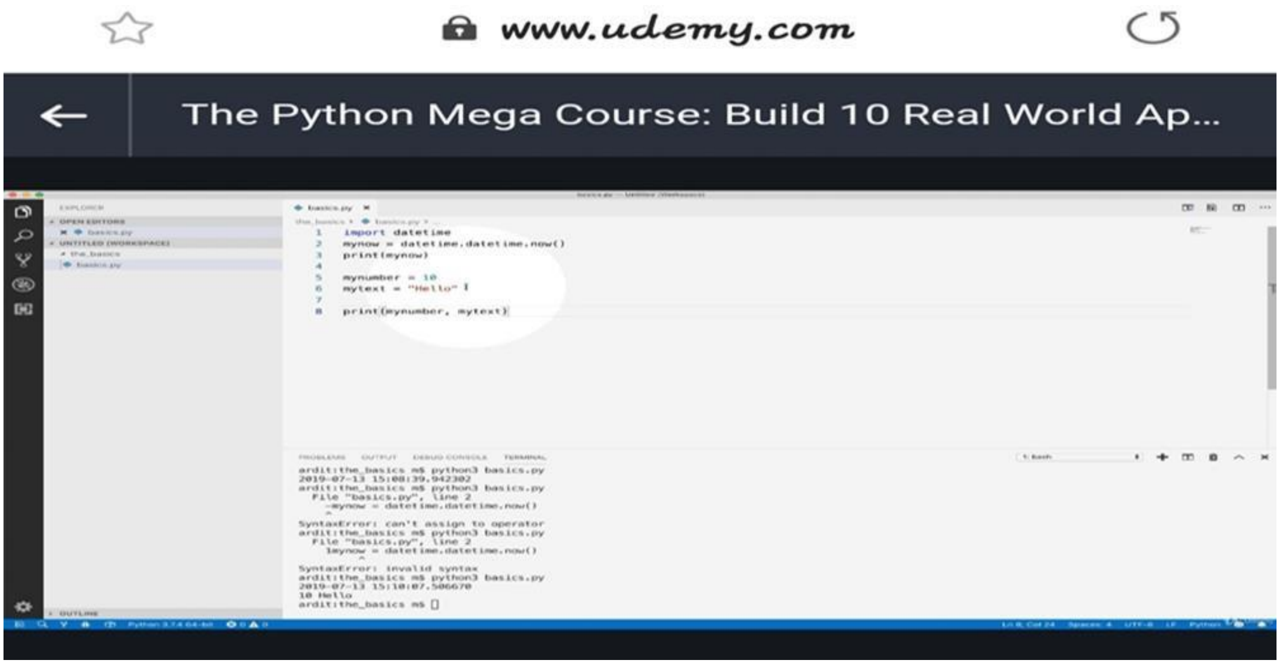
DAILY ASSESSMENT 3

AFTERNOON SESSION DETAILS

Date:	**20-05-2020**	**Name:**	**Shilpa S**
Course:	**python**	**USN:**	**4al14ec078**
Topic:	**Variables,integer strings and float , List types,ranges**	**Semester & Section:**	**8th A SEC**
Github Repository:	**Shilpa-online**		

Image of session



```
import datetime
mynow = datetime.datetime.now()
print(mynow)

mynumber = 10
mytext = "Hello"

print(mynumber, mytext)
```

```
ardit:the_basics m$ python3 basics.py
2019-07-13 15:08:39.942302
ardit:the_basics m$ python3 basics.py
  File "basics.py", line 2
    -mynow = datetime.datetime.now()
    ^
SyntaxError: can't assign to operator
ardit:the_basics m$ python3 basics.py
  File "basics.py", line 2
    1mynow = datetime.datetime.now()
         ^
SyntaxError: invalid syntax
ardit:the_basics m$ python3 basics.py
2019-07-13 15:10:07.506670
10 Hello
ardit:the_basics m$ 
```

Variables

Creating Variables

- 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.

Example

```
x= 5
y= "John"
print(x)
print(y)
```

- Variables do not need to be declared with any particular type and can even change type after they have been set.

❖ 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)

❖ Global Variables

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

Example : Create a variable outside of a function, and use it inside the function

```
x = "awesome"
def                myfunc():
  print("Python is " + x)
```

```
myfunc()
```

The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the global keyword.

CREATE INTEGER ,STRINGS AND FLOAT

Integer

We already know the following operators which may be applied to numbers: +, -, * and **. The division operator / for integers gives a floating-point real number (an object of type float). The exponentiation ** also returns a float when the power is negative.

❖ Floating numbers

When we read an integer value, we read a line with input() and then cast a string to integer using int(). When we read a floating-point number, we need to cast the string to float using float().Floats may also be in scientific notation, with E or e indicating the power of 10 (2.5e2 = 2.5 x 102 = 250).

❖ String lists

Python is often used to process textual data. With strings, and string lists, we store and can handle this data in an efficient way.

Example : create a list of three strings.

```
strings = ["one", "two", "THREE"]
# ... Display length of list.
print(len(strings))
# ... Display all string elements in list.
```

```
for st in strings:
    print(st)
```

- **List types**

There are four collection data types in the Python programming language:

- List is a collection which is ordered and changeable. Allows duplicate members.
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- Set is a collection which is unordered and unindexed. No duplicate members.
- Dictionary is a collection which is unordered, changeable and indexed. No duplicate members.
- When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and, it could mean an increase in efficiency or security.

- **Ranges**

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

Syntax :

range(start, stop, step)

- **Parameter Values**

- start   Optional: An integer number specifying at which position to start. Default is 0
- stop   Required: An integer number specifying at which position to stop (not included).
- step   Optional: An integer number specifying the incrementation. Default is 1.