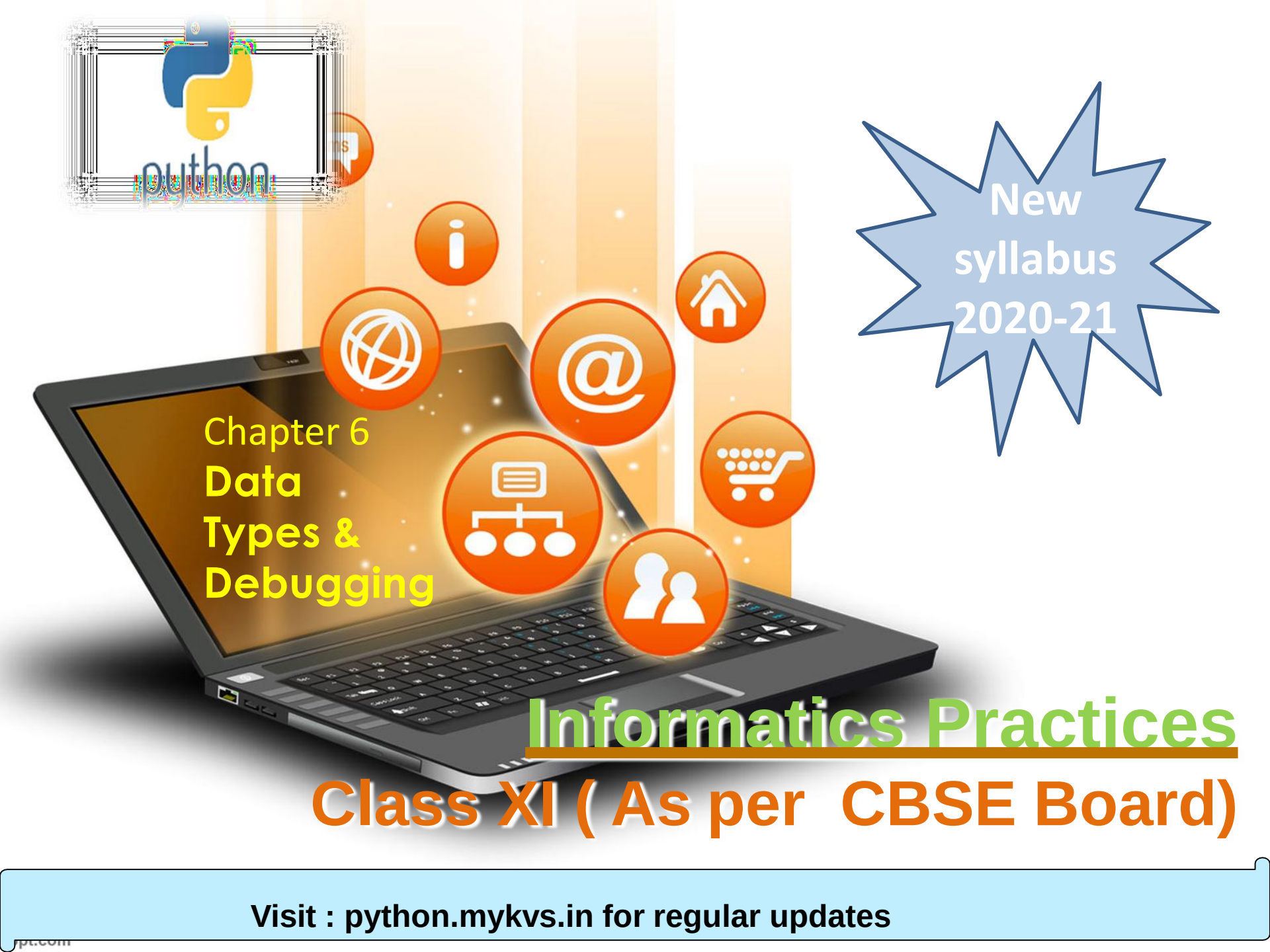

python
New syllabus 2020-21
Chapter 6
Data Types & Debugging
Informatics Practices
Class XI (As per CBSE Board)
Visit : python.mykvs.in for regular updates

Data handling

Most of the computer programming language support data type, variables,operator and expression like fundamentals.Python also support these.

Data Types

Data Type specifies which type of value a variable can store. type() function is used to determine a variable's type in Python.

fppt.c...

Data type continue

Data Types In Python

1. Number
2. String
3. Boolean
4. List
5. Tuple
6. Set
7. Dictionary

fppt.c...

Data type continue

Mutable and Immutable Data type

A mutable data type can change its state or contents and immutable data type cannot.

Mutable data type:

list, dict, set, byte array

Immutable data type:

int, float, complex, string, tuple, frozen set [note: immutable version of set], bytes

Mutability can be checked with id() method.

```
x=10
print(id(x))
x=20
print(id(x))
#id of both print statement is different as integer is immutable
```


fppt.c...

Data type continue

1. Number In Python

It is used to store numeric values

Python has three numeric types:

1. Integers
2. Floating point numbers
3. Complex numbers.

fppt.c...

Data type continue

1. Integers

Integers or int are positive or negative numbers with no decimal point. Integers in Python 3 are of unlimited size.

e.g.

```
a= 100
b= -100
c= 1*20
print(a)
print(b)
print(c)
```

Output :-

```
100
-100
200
```


fppt.c...

Data type continue

Type Conversion of Integer

int() function converts any data type to integer.

e.g.

```
a = "101" # string
b=int(a) # converts string data type to integer.
c=int(122.4) # converts float data type to integer.
print(b)
print(c)Run Code
```

Output :-

101

122

fppt.c

Data type continue

2. Floating point numbers

It is a positive or negative real numbers with a decimal point.

e.g.

```
a = 101.2
b = -101.4
c = 111.23
d = 2.3*3
print(a)
print(b)
print(c)
print(d)Run Code
```

Output :-

```
101.2
-101.4
111.23
6.8999999999999995
```


fppt.c...

Data type continue

Type Conversion of Floating point numbers

float() function converts any data type to floating point number.

e.g.

```
a='301.4' #string
b=float(a) #converts string data type to floating point number.
c=float(121) #converts integer data type to floating point number.
print(b)
print(c)Run Code
```

Output :-

```
301.4
121.0
```


fppt.c...

Data type continue

3. Complex numbers

Complex numbers are combination of a real and imaginary part.Complex numbers are in the form of X+Yj, where X is a real part and Y is imaginary part.

e.g.

```
a = complex(5) # convert 5 to a real part val and zero imaginary part
print(a)
b=complex(101,23) #convert 101 with real part and 23 as imaginary part
print(b)Run Code
```

Output :-

```
(5+0j)
(101+23j)
```


fppt.c...

Data type continue

2. String In Python

A string is a sequence of characters. In python we can create string using single (' ') or double quotes (" ").Both are same in python.

e.g.

```
str='computer science'
print('str-',   str) # print string
print('str[0]-',   str[0]) # print first char 'h'
print('str[1:3]-',   str[1:3]) # print string from postion 1 to 3 'ell'
print('str[3:]-',   str[3:]) # print string staring from 3rd char 'llo world'
print('str *2-',   str *2 ) # print string two times
print("str +'yes'-",   str +'yes') # concatenated string
```

Output

```
str- computer science
str[0]- c
str[1:3]- om
str[3:]- puter science
str *2- computer sciencecomputer science
str +'yes'- computer scienceyes
```


fppt.c...

Data type continue

Iterating through string

e.g.

```
str='comp sc'
for i in str:
    print(i)
```

Output

```
c
o
m
p

s
c
```


fppt.c...

Data type continue

3. Boolean In Python

It is used to store two possible values either true or false

e.g.

```
str="comp sc"
boo=str.isupper() # test if string contains upper case
print(boo)
```

Output

False

fppt.c...

Data type continue

4.List In Python

List are collections of items and each item has its own index value.

5. Tuple In Python

List and tuple, objects mean you cannot modify the contents of a tuple once it is assigneboth are same except ,a list is mutable python objects and tuple is immutable Python objects. Immutable Python d.

e.g. of list

```
list =[6,9]
list[0]=55
print(list[0])
print(list[1])
```

OUTPUT

55

9

e.g. of tuple

```
tup=(66,99)
Tup[0]=3 # error message will be displayed
print(tup[0])
print(tup[1])
```


fppt.c

Data type continue

6. Set In Python

It is an unordered collection of unique and immutable (which cannot be modified)items.

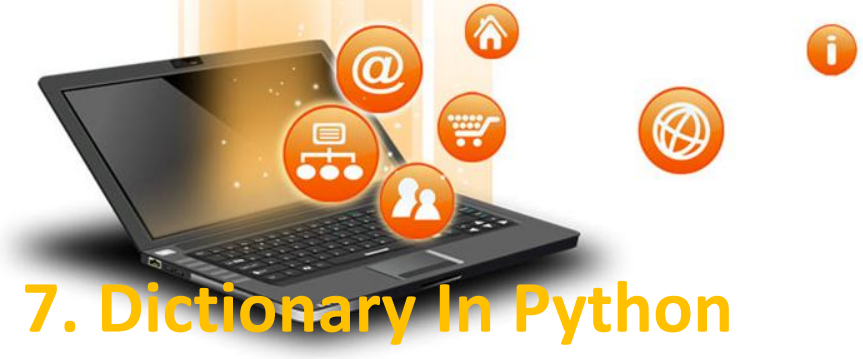
e.g.

```
set1={11,22,33,22}
print(set1)
```

Output

{33, 11, 22}

fppt.c...



Data type continue

7. Dictionary In Python

It is an unordered collection of items and each item consist of a key and a value.

e.g.

```
dict = {'Subject': 'comp sc', 'class': '11'}
print(dict)
print ("Subject : ", dict['Subject'])
print ("class : ", dict.get('class'))
```

Output

```
{'Subject': 'comp sc', 'class': '11'}
Subject :  comp sc
class :  11
```


fppt.

Type conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.

Python has two types of type conversion.

- Implicit Type Conversion
- Explicit Type Conversion

Implicit Type Conversion:

In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

e.g.

```
num_int = 12
num_flo = 10.23
num_new = num_int + num_flo
print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))
print("Value of num_new:",num_new)
print("datatype of num_new:",type(num_new))
```

OUTPUT

```
('datatype of num_int:', <type 'int'>)
('datatype of num_flo:', <type 'float'>)
('Value of num_new:', 22.23)
('datatype of num_new:', <type 'float'>)
```


fppt.c...

Type conversion

Explicit Type Conversion:

In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like int(),float(),str() etc.

e.g.

```
num_int = 12
num_str = "45"
print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))
num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))
num_sum = num_int + num_str
print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
```

OUTPUT

```
('Data type of num_int:', <type 'int'>)
('Data type of num_str before Type Casting:', <type 'str'>)
('Data type of num_str after Type Casting:', <type 'int'>)
('Sum of num_int and num_str:', 57)
('Data type of the sum:', <type 'int'>)
```


fppt.c...

Debugging

Debugging means the process of finding errors, finding reasons of errors and techniques of their fixation.

An error, also known as a bug, is a programming code that prevents a program from its successful interpretation.

Errors are of three types –

- Compile Time Error
- Run Time Error
- Logical Error

fppt.c...

Debugging

Compile time error :

These errors are basically of 2 types –

Syntax Error :Violation of formal rules of a programming language results in syntax error.

For ex-

```
len('hello') = 5
  File "<stdin>", line 1
SyntaxError: can't assign to function call
```

Semantics Error: Semantics refers to the set of rules which sets the meaning of statements. A meaningless statement results in semantics error.

For ex-

```
x * y = z
```


fppt.c

Debugging

Logical Error

If a program is not showing any compile time error or run time error but not producing desired output, it may be possible that program is having a logical error.

Some example-

- Use a variable without an initial value.
- Provide wrong parameters to a function
- Use of wrong operator in place of correct operator required for operation

X=a+b (here – was required in place of + as per requirement

fppt.c...

Debugging

Run time Error

These errors are generated during a program execution due to resource limitation.

Python is having provision of checkpoints to handle these errors.

For ex-

```
a=10
b=int(input("enter a number"))
c=a/b
```

Value of b to be entered at run time and user may enter 0 at run time,that may cause run time error,because any number can't be devided by 0

fppt.c...

Debugging

Run time Error

In Python, try and except clauses are used to handle an exception/runtime error which is known as exception handling

```
try:
# code with probability of exception will be written here.
a=10
b=int(input("enter a number"))
c=a/b
except:
#code to handle exception will be written here.
print("devide by zero erro")
```


fppt.c...

Debugging

Available exception in python

Exception Name	Description
IOError	This exception generates due to problem in input or output.
NameError	This exception generates due to unavailability of an identifier.
IndexError	This exception generates when subscript of a sequence is out of range.
ImportError	This exception generates due to failing of import statement.
TypeError	This exception generates due to wrong type used with an operator or a function.
ValueError	This exception generates due to wrong argument passed to a function.
ZeroDivisionError	This exception generates when divisor comes to zero.
OverFlowError	This exception generates when result of a mathematical calculation exceeds the limit.
KeyError	This exception generates due to non-availability of key in mapping of dictionary.
FOFError	This exception generates when end-of-file condition comes without reading input of a built in function.

fppt.c...

Debugging

In python debugging can be done through

- **Print line debugger**
- **Debugging tool**

fppt.c...

Debugging

Print line debugger

– At various points in your code, insert print statements that log the state of the program

- You will probably want to print some strings with some variables
- You could just join things together like this:

```
>>>x=9
>>>print 'Variable x is equal to ' + str(x)
Output : Variable x is equal to 9
```

- … but that gets unwieldy pretty quickly
- The format function is much nicer:

```
>>>x=3
>>>y=4
>>>z=9
>>>print 'x, y, z are equal to {}, {}, {}'.format(x,y,z)
Output : x, y, z are equal to 6, 4, 8
```


fppt.c...

Debugging

Print line debugger

- Python Debugger: pdb

– insert the following in your program to set a breakpoint

– when your code hits these lines, it'll stop running and launch an interactive prompt for you to inspect variables, step through the program, etc.

```
import pdb
pdb.set_trace()
```

n to step to the next line in the current function

s to step into a function

c to continue to the next breakpoint

you can also run any Python command, like in the interpreter

fppt.c

Debugging

Create a.py file with below code and run it in python use n to step next line.

```
num_list = [500, 600, 700]
alpha_list = ['x', 'y', 'z']

import pdb
pdb.set_trace()                    #debugging code
def nested_loop():
    for number in num_list:
        print(number)
        for letter in alpha_list:
            print(letter)

if __name__ == '__main__':
    nested_loop()
```

While executing above code whole program will be traced.

Another way is to invoke the pdb module from the command line.

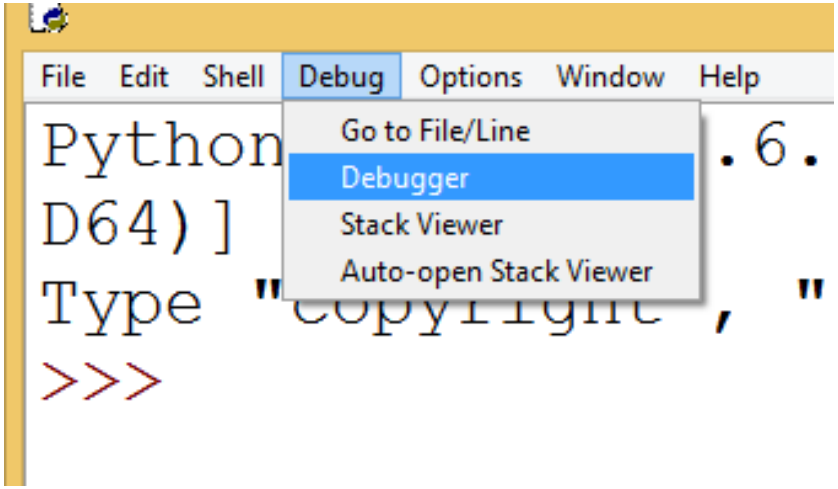
fppt.c...

Debugging

Debugger tool

Another technique for removing an error is code tracing. In this technique, lines are to be executed one by one and their effect on variables is to be observed. Debugging tool or debugger tool is provided in Python for this.

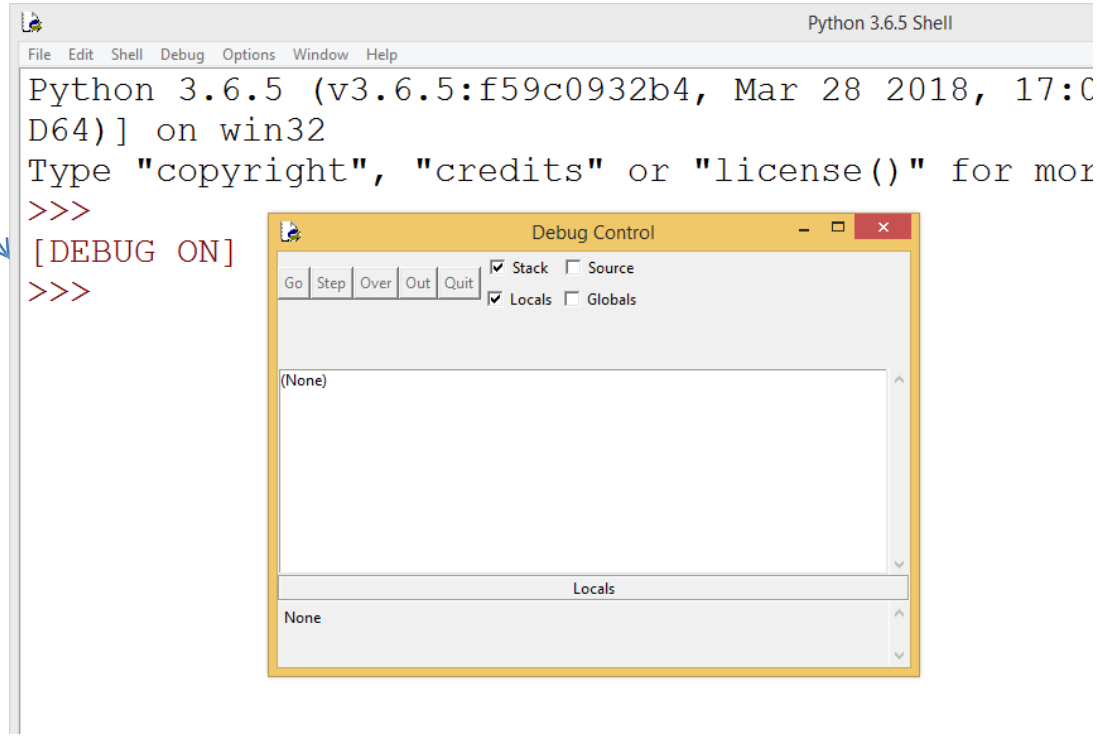
In Python3.6.5, to make debugger tool available, click on debugger option in debug menu.



Debugging

Debugger tool

Then, a box will be opened and a message will come saying DEBUG ON



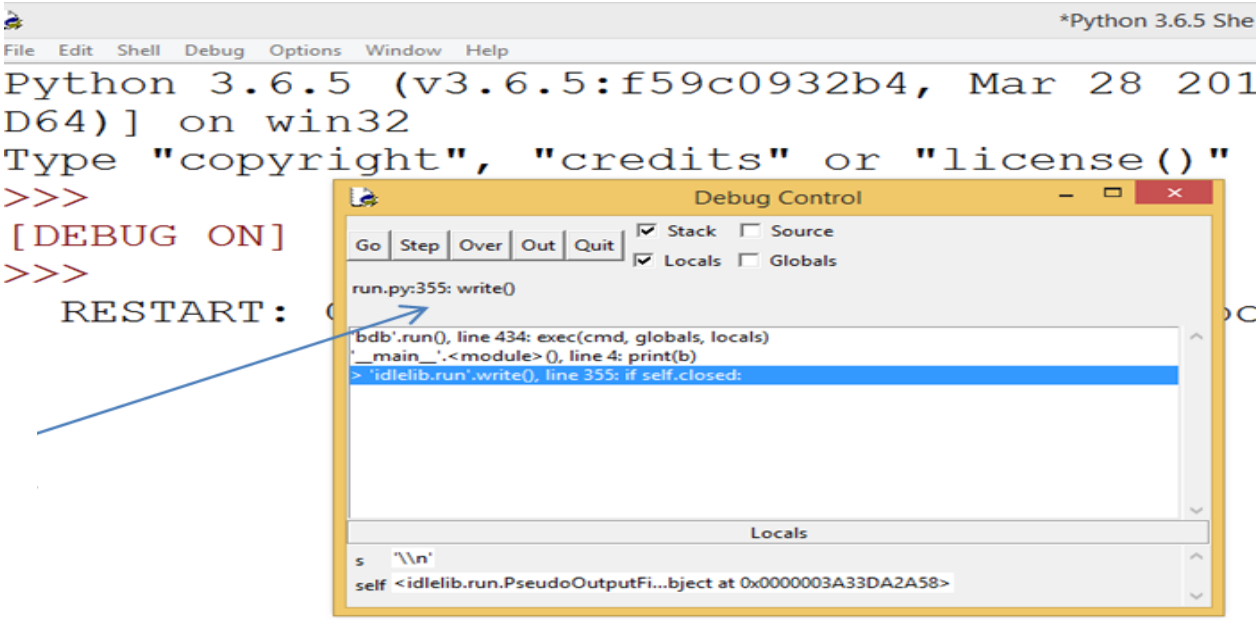
Then, we will open our program from file menu and will run it.

fppt.c...

Debugging

Debugger tool

Then after it will be shown like this in debugger.



Click on STEP button for each line execution one by one and result will be displayed in output window. When we will get wrong value, we can stop the program there and can correct the code.

fppt.c...