New syllabus 2020-21

Chapter 9
Program Execution & Error Handling

Computer Science

Class XI (As per CBSE Board)

Generally, the programs developed in high level language like C, C++, Java etc.,cannot understand by the computer . It can understand only low level language. So, the program written in high level language to be converted into low level language to make it understandable for the computer. This conversion is performed using either Interpreter or Compiler.The basic flow of any program execution is shown in diagram.
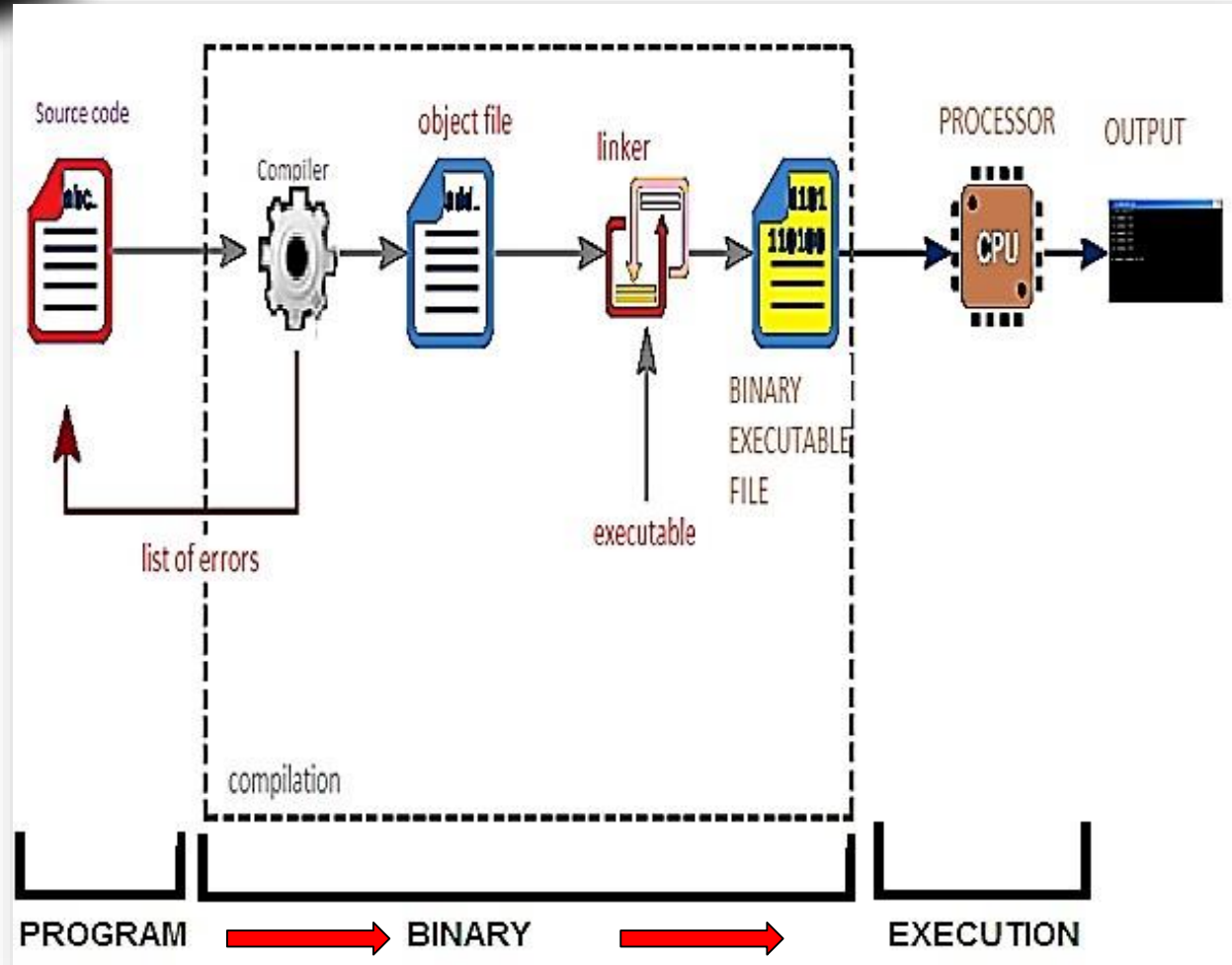
In python IDLE program is executed from run module option of run menu

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

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

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

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

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")

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.

In python debugging can be done through
- **Print line debugger**
- **Debugging tool**

fppt.com

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
```

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

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.

$ python -m pdb mycode.py

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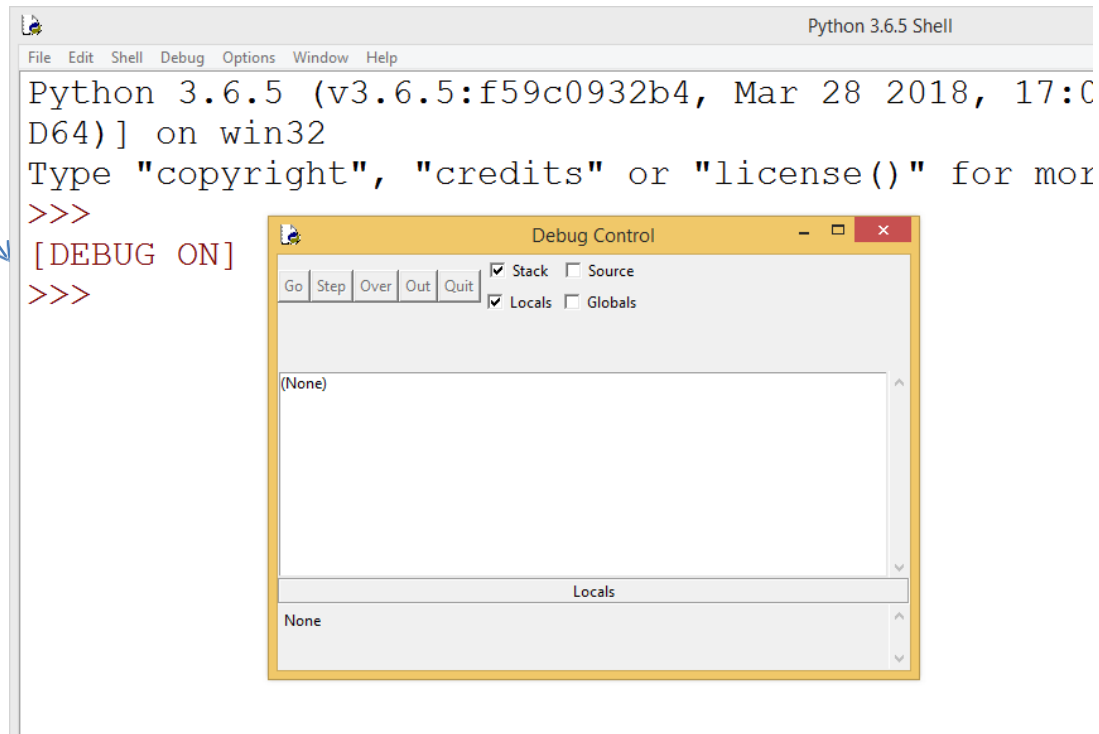
In Python3.6.5, to make debugger tool available, click on debugger option in debug menu.
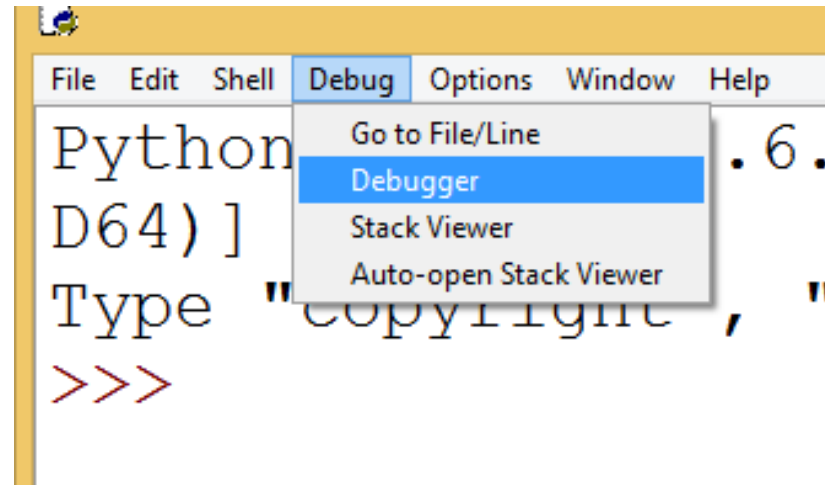
Debugger tool

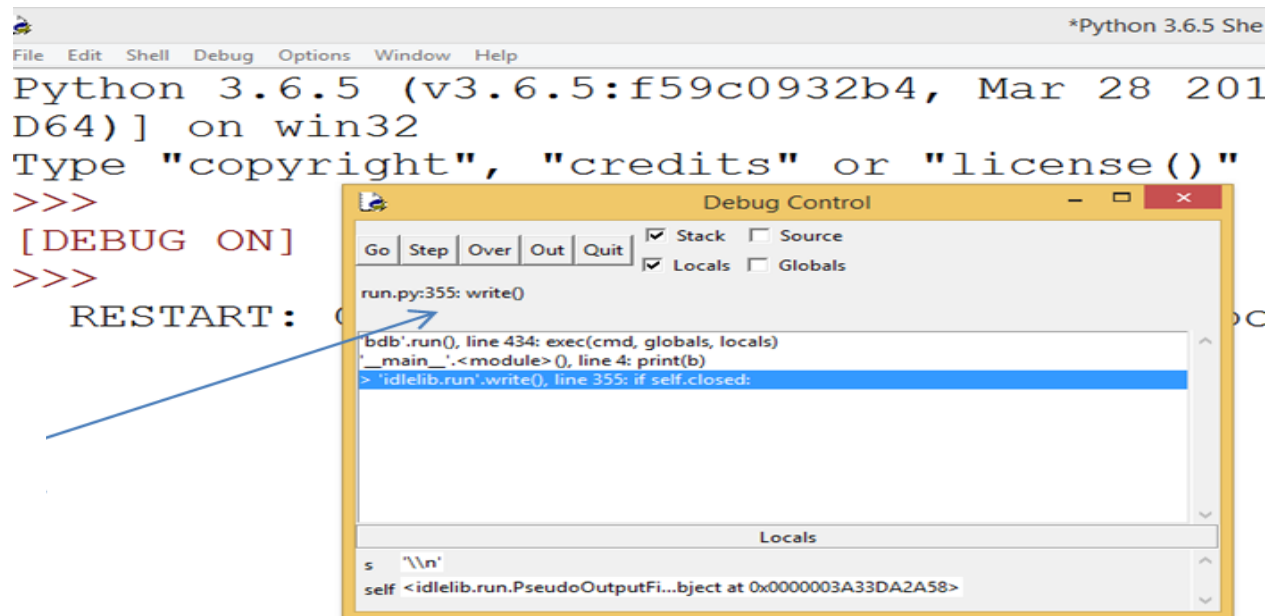Then, a box will be opened and a message will come saying DEBUG ON

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

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.