A-level
Computer Science

7.2. Translators and IDE

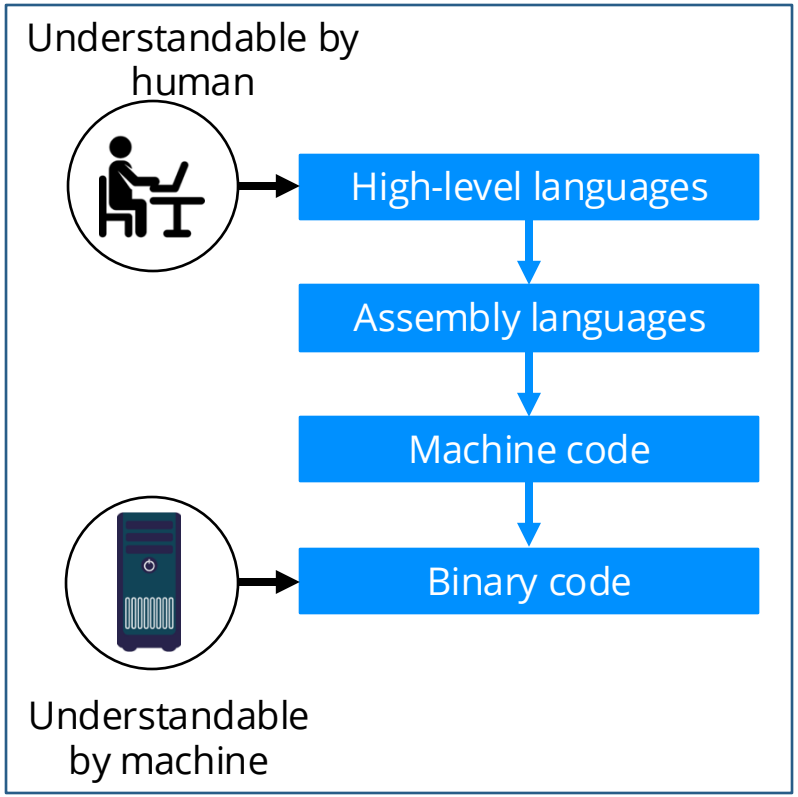
Lesson Objectives

Students will learn about:

- Translators
- Stages of compilation
- Dynamic link libraries
- Linkers & Loaders
- IDE
- Errors and debugging

Translators

Programs can be written in high-level or low-level languages, according to the requirements of the user.



High level languages

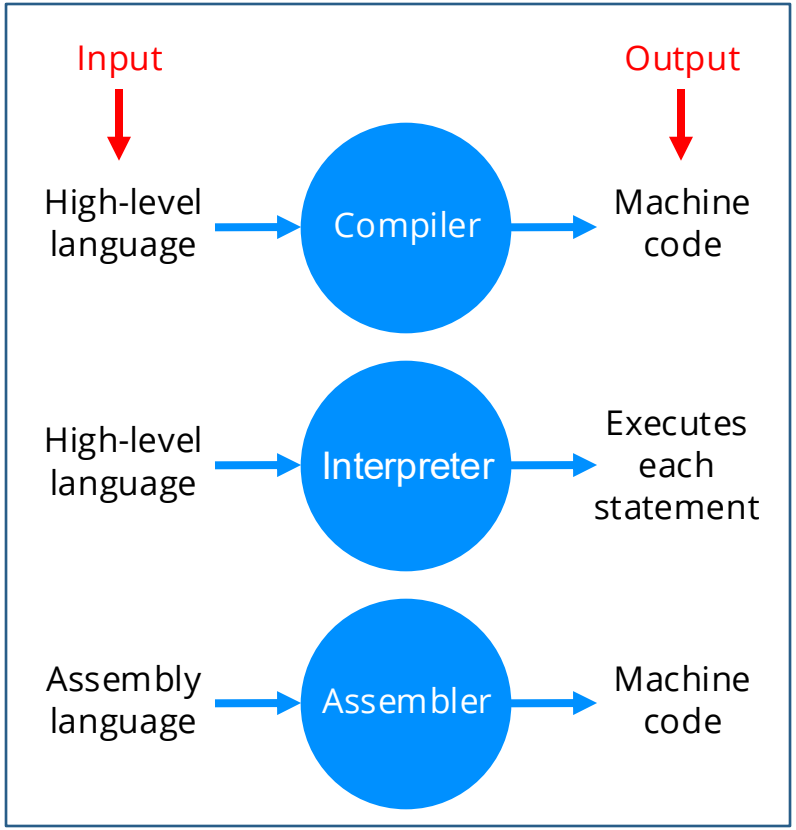
- High-level languages enable a programmer to write programs for a computer without the knowledge of the hardware and instruction sets of that computer.
- Portable programs: can be used in different systems
- For example: Java, C++ and Python
- Same programming concept can be applied to different high-level languages.

Assembly language

- Assembly language is used by programmers to make use of special hardware.
- Instructions used are dependent upon the type of machine.
- The code does not take up much space of primary memory and performs its task quickly.

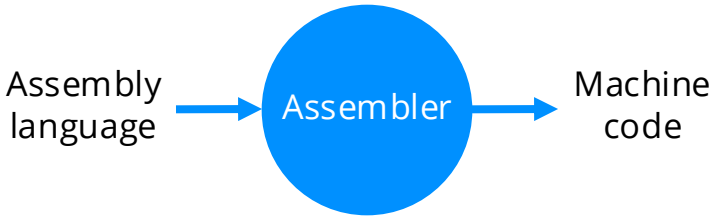
Translators

A utility program that translates the program written by programmers in assembly or high-level language into binary form, which is understandable by the computer, is called a translator.



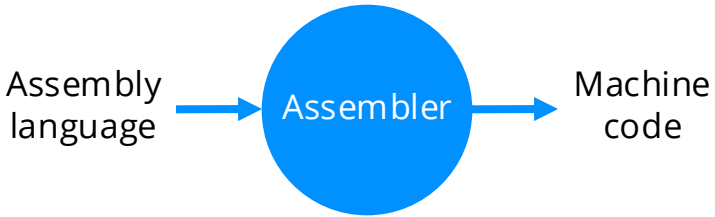
Assembler

- A computer program that translates a program written in an assembly language into machine code.
- Sometimes, assembly language programs are converted to an intermediate code called bytecode.
- Once assembled, the same code can be used again multiple times without re-assembly.



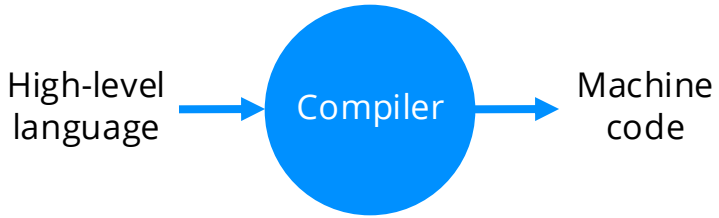
Assembler

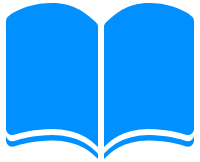
- Sometimes, the assembler generates an intermediate code called bytecode.
- Processors based on different architectures have different assembly language instructions.
- The input to the assembler is called source code.



Compiler

- A computer program that translates a program written in a high-level language into machine code that can be directly used by a computer to perform required tasks.





Compiler

- Once compiled, the same code can be used again multiple times without recompilation.
- A compiler optimises the code and errors are picked up only after the complete compilation of a program.

- Once a code is compiled, it is hard to convert it back to its source code.
- Hence, it is hard to modify compiled codes.

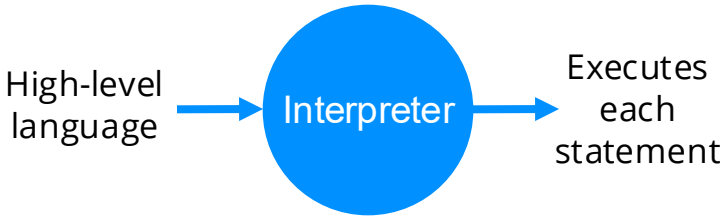
Compiler

- The program in a high-level language is source code.
- Hardware specific object code is produced.
- Different platforms will require different compilers.

- Windows OS + intel processors → platform.
- Apple's operating system + PowerPC → platform.

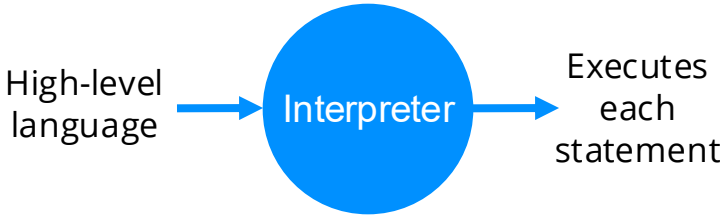
Interpreter

- A computer program that reads a statement from a program written in a high-level language, converts it into intermediate code and then proceeds to the next statement and so on.
- Intermediate code: assembly language



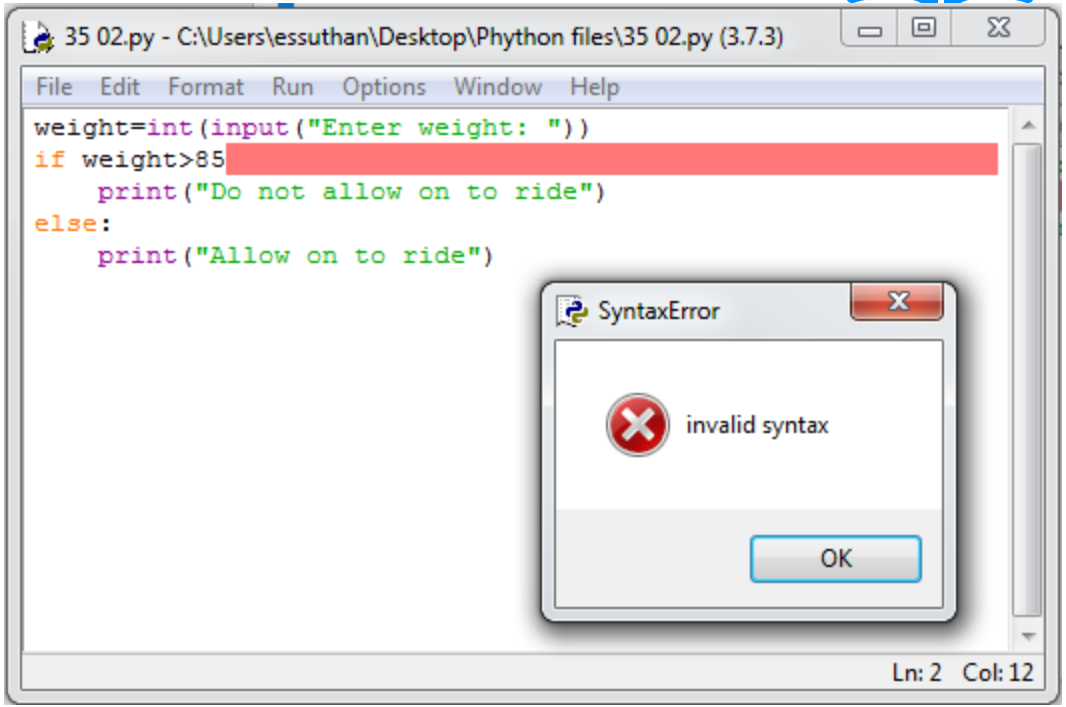
Interpreter

- This intermediate code is then converted to machine code and then, executed.
- In case an error is found, the interpreter prompts the user to correct the error.
- Code does not get optimised.



Interpreter

- Interpreter finds out errors before the execution of program.
- Syntax error in Python IDE for missing semicolon.



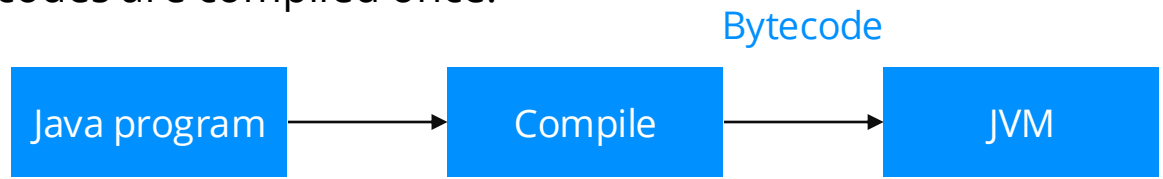
Compiler	Interpreter
Source code is compiled separately for each architecture.	*Used in case the source code is used in machines of different architectures. The interpreted programs are then, compiled. Interpreted codes are free of errors. But the time taken to execute the code increases.*
Executable file of machine code is produced.	*No executable file of machine code is produced.*
Complete translation of the code and errors are picked up only after the completed translation.	*Step-by-step execution and errors are picked up once found.*
Optimisation of code.	*No optimisation of code.*
Compiled programs work independently without re-compilation.	*Interpreted programs can only be used with the interpreter.*

	Compiler	Interpreter	Assembler
What it does	Translates a high-level language into machine code.	Executes one statement of high-level language at a time and then proceeds to the next.	Translates a low-level language into machine code.
How it works	One statement in high-level language can be translated into several machine code instructions.	One statement in high-level language may require several machine code instructions to be executed.	One statement in low-level language is translated into one machine code instruction.

	Compiler	Interpreter	Assembler
Output	Executable file of machine code is produced. Compiled programs work independently without re-compilation.	No executable file of machine code is produced. Interpreted programs can only be used with the interpreter.	Executable file of machine code is produced. Assembled programs work independently without re-assembly.

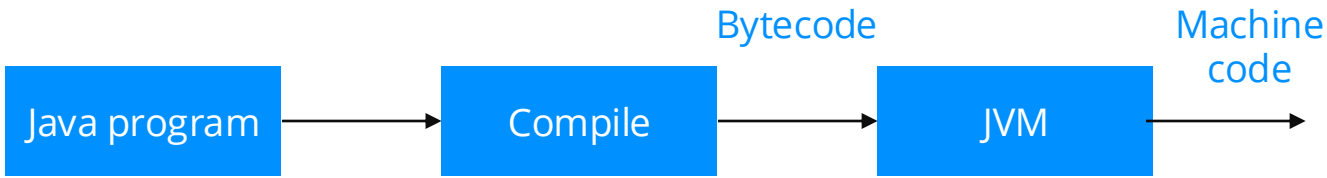
Bytecode

- An intermediate representation that combines compiling and interpreting.
- Executed by bytecode interpreter.
- Java codes are compiled once.



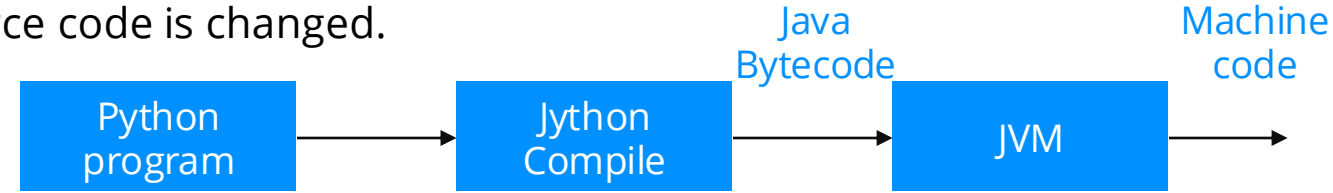
Java Virtual machine

- Converts the bytecode to machine code understandable to that machine.
- Now, machine independence is achieved for Java programs.

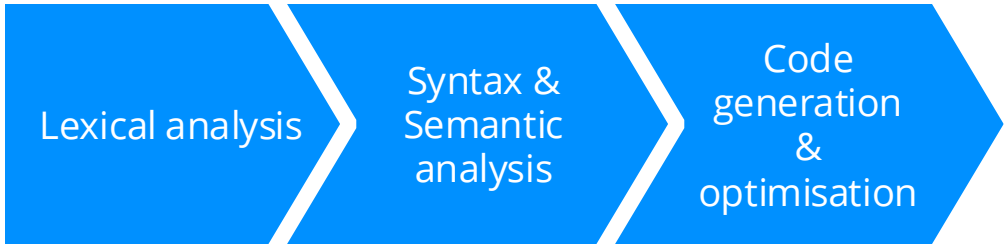


Advantages

- Enhances security: Bytecode of programs from unknown sources is run in JVM first. If it is found not to be malicious, the main program is run.
- Machine independence.
- In Python, the bytecode is generated before execution every time the source code is changed.



Stages of compilation



Lexical analysis: functions

- Comments in a program are specified using # or // symbol. These comments are removed. Unwanted spaces are also removed.

  For example: num = int (input (“Enter a number”)) is converted to num=int(input(“Enter a number”))

- There are rules to name variables in programming languages. One of the rules in Python is that name of a variable cannot start from a number. In lexical analysis, the variable name 2num would be marked as an illegal identifier.

Lexical analysis: functions

- An error also arises in case a constant is assigned an illegal value such as of different data type. In such cases, an overflow or underflow error occurs.
- Keywords, constants and variables are replaced with unique symbols called tokens. Identifiers are replaced by pointer to the address of the location of variable. Keywords such as input, if, return and print are replaced by item codes.

Lexical analysis: symbol table

Stores detail about keywords, variables and constants used in the source code. For each entry in the symbol table, the following details are entered:

- Name of variable or keyword.
- Kind of item: Variable, keyword, constant, procedure, function or array.
- Data type of item: Float, int, char, string, date or Boolean.
- Address of the item or value of constant.
- Location information is required to access arrays. Array index must be within the bounds of the array. So, bounds of array is stored.
- Procedures and function require information about each parameter such as its data type.

Lexical analysis: symbol table

Consider the statement to find the area of rectangle in Python.

*Area = width * height*

The symbol table of this statement with run-time values for example is given.

Name of item	Kind of item	Data type	Run-time address or value
Area	variable	float	12.00
=	operator		
width	variable	float	3.00
height	variable	float	4.00
*	operator		

Lexical analysis: tokenisation

- After the creation of symbol table, each item is tokenised. The statement Area = width * height may be tokenised as 7 1 2 3 6.
- The lexical analyser fills in the name of the item and the run-time address or value column.
- The kind and type of item columns are filled by the syntax analyser in the next stage of compilation.

Syntax analysis

- A tool that determines whether the sequence of tokens obey the rules of a valid sentence in the language.
- Parsing: a task that determines whether each sentence is valid by applying a set of rules.
- Brackets in programs must be paired correctly. This is checked using stacks.
- Arithmetic operators are analysed and prioritised.

Syntax analysis

- The expressions are also converted into a form such that it can be easily converted to machine code.
- Syntax errors:
  - Unpaired brackets
  - Missing semicolons
  - Misspelt keywords

Semantic analysis

- Semantics represent the meaning of the language.
- Whereas, syntax represents the grammar of the language.
- Semantic errors:
  - Using undeclared variables (in some languages).
  - Type mismatch error: assigning a real value to an integer variable.
  - Using a real value to count the number of loops.

Semantic analysis

Consider this program:

```
Num = 10
Subject = 'Computing'
print(num)
print(subject)
print(num + subject)
```

```
10
Computing
Traceback (most recent call last):
  File "C:/Users/ User 1 /Desktop/KS3 computing/17 01.py", line 5, in <module>
    print(num1+subject)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>>
```

Code generation & Optimisation

- Final phase of compilation in which machine code is generated.
- Code optimisation techniques help in the reduction of execution time and resources used by a program.
- This is done by removing redundant instructions and changing the way a program runs but still producing the same results.

Code generation & Optimisation

- For example: assigning a value to a variable inside a loop is better when its placed outside a loop.
- This avoids the execution of the same statement numerous times.
- Due to this action, certain programs may produce incorrect results.
- Code optimisation also leads to an increase in compilation time in some cases.

Dynamic linked libraries

- Dynamic linked library is a shared library of sub-programs. These sub-programs are used by programmers in their programs whenever required.
- Some routines are used by programmers several times in their program. So, a library of such routines is developed and tested.
- When the statement for call of routine is translated, the code for the routine is taken from the library and executed.

Linkers

- When the main program is compiled, the compiled subroutines must be linked to the machine code. This is done by linker.
- When a subroutine is used, call and return statements transfer the control from the main program to a subroutine and vice versa.
- The function of linker is to provide machine addresses at the call and return statements.
- Ensures that the modules are linked together.

Loaders

- A relocating loader loads object codes anywhere in the memory.
- Some conditions are applicable for the functioning of relocating loader:
  - Program must not contain any absolute address
  - Program must be in a relocatable format.

2.

Activity

Activity-1

Duration: 15 minutes

1. What are the advantages and disadvantages of high-level and low-level languages?
2. What translation software is used to translate high-level and low-level languages to machine code? State the process involved with each translator in a diagram below.

Activity-2

Duration: 15 minutes

1. Analyse the Python code given. There are few errors. State any two things performed by:

a) Lexical analysis

b) Syntax analysis

```
1count=0
total=0
while (1count<5)
    value=int (input("Enter a number:“)
    1count=1count + 1
    total = total + value
print("The total is: ", total)
Average = total / 1count
print("The average is: %.2f" %average)
```

3.

End of topic questions

End of topic questions

1. In what case does a programmer choose low-level languages over high-level language?
2. When does a programmer choose high-level languages over low-level language?
3. What are translators?
4. How is the function of a compiler different from an interpreter?
5. What is the function of an assembler?

End of topic questions

6. What are the advantages of using bytecodes?
7. What are the functions of lexical analyser?
8. State the type of errors that arise in lexical analysis. Also, specify when do these errors occur.
9. State examples of syntax errors and semantic errors.
10. What is dynamic link library?
11. What is the function of linker?

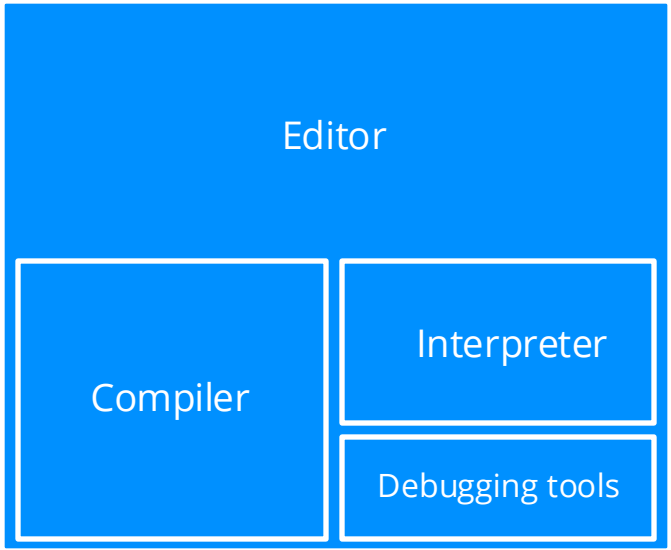
Integrated development environment

Content

- Debugging using IDE
- Types of errors
- Different types of test data
- Trace tables: finding and correcting errors in algorithms
- Software maintenance activities

Integrated Development Environment (IDE)

Integrated development environment consists of an editor with an interpreter and or compiler and debugging tools.

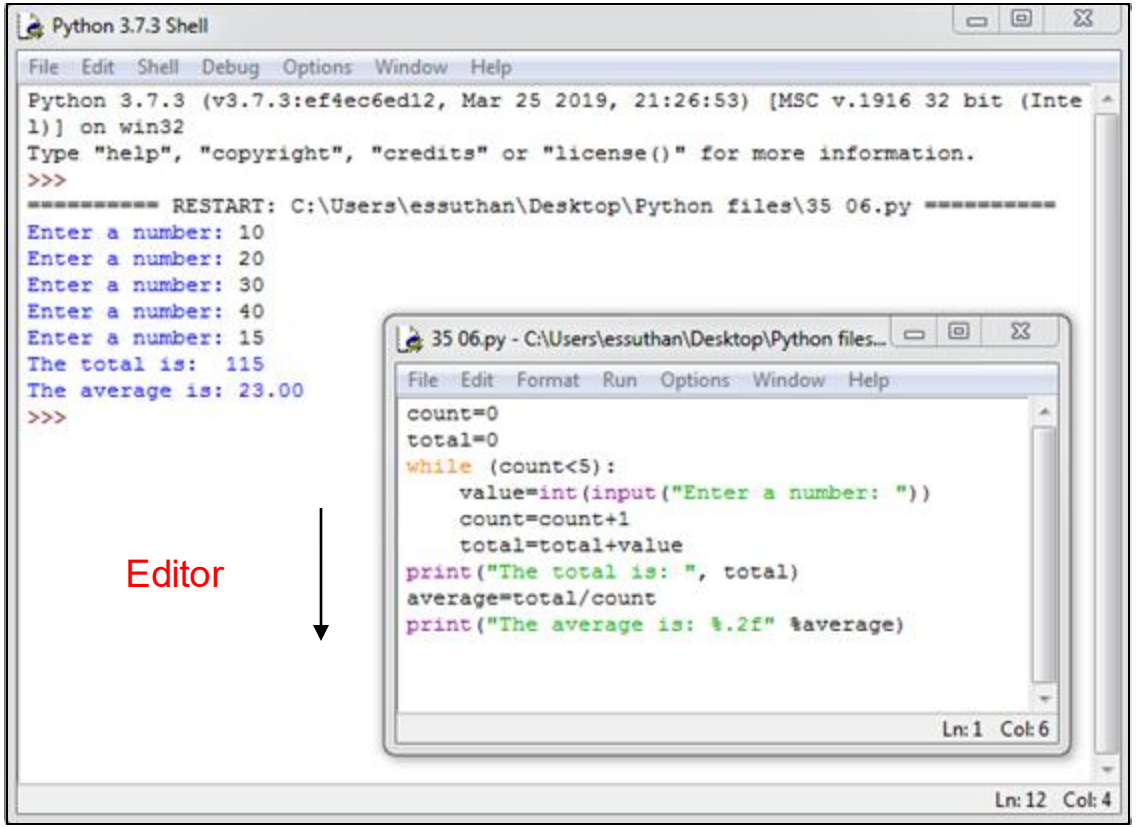


Integrated Development Environment (IDE): Editor

- Editor is a tool used by the programmer to write, edit and save codes.
- Editor looks like a word processor, but formatting options are not provided because formatting could contaminate the embedded codes.
- Editors also make the process of writing codes simpler. The codes written using editors are more readable.
- Conditional and iterative statements are identified, and the statements to be executed for a condition or in a loop are indented.
- Variables are presented in different colours.

IDE

- Idle used for Python programming.
- The coloured statements and indentations help us to understand the program.
- Program to find the total and average of 5 numbers entered by a user.

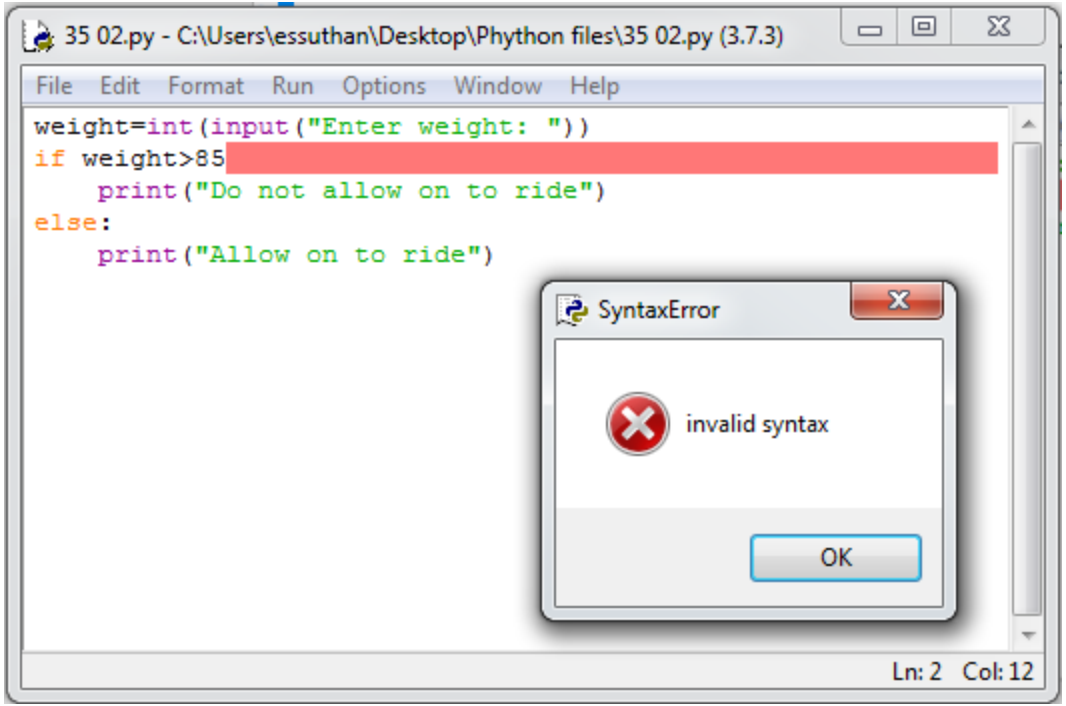


Integrated Development Environment (IDE)

- IDE is used in the development of high-level language as it improves the speed of program development.
- Microsoft Visual Studio is an example of an Integrated Development Environment that is used to develop apps for Android, iOS, Mac and Windows.
- IDE allows the user to enter the code, edit it, debug errors and then compile it. Some IDEs also contain an interpreter.
- The tools provided by an IDE to check and correct errors are called debugging tools.

Syntax error

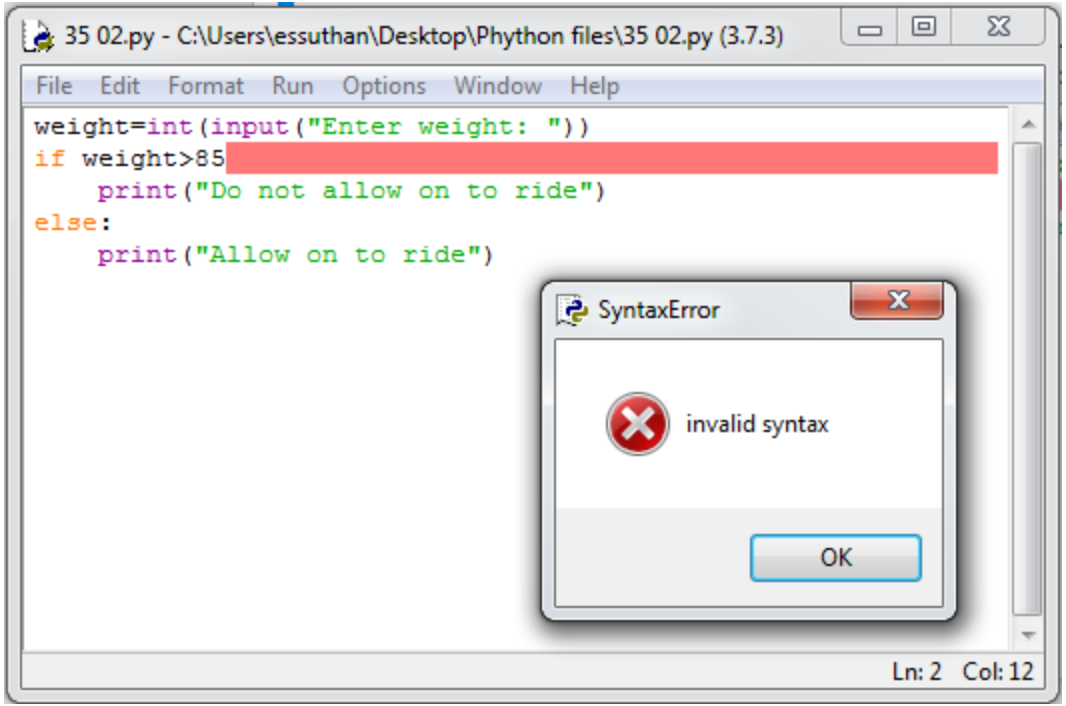
Syntax errors occur when the programming rules are not followed.



In line 2, the syntax error is identified. It states ‘invalid syntax’. This is because the colon at the end of the line is missing.

The correct statement is:

if weight>85:



Syntax error

- Some other examples of syntax errors in Python include:
  - ✓ incompatible variable types
  - ✓ incorrect variable names due to spelling
  - ✓ improper assignments. 3+5=k is an example of an improper assignment. It should be k=3+5.
- In other programming languages, the variable declaration is very important. Missing or improper variable declarations is a syntax error.

Logic errors

- When a program runs, if it produces incorrect or unexpected results, it is said to contain **logic errors.**
- An incorrect translation of problem statement or algorithm leads to logic error.
- Incorrect expressions and algorithms are examples of logic errors.

Run-time errors

- Run-time errors are said to occur when the program execution comes to an unexpected halt or crashes. For example, infinite loops.
- Dividing a number by zero results in a run-time error. Let us execute a program that divides two numbers.

```
Input (p)
Input (q)
answer=p/q
```

- If q=0, a ZeroDivisionError occurs in line 3. This is because a number cannot be divided by zero.

Run-time errors: Example of output

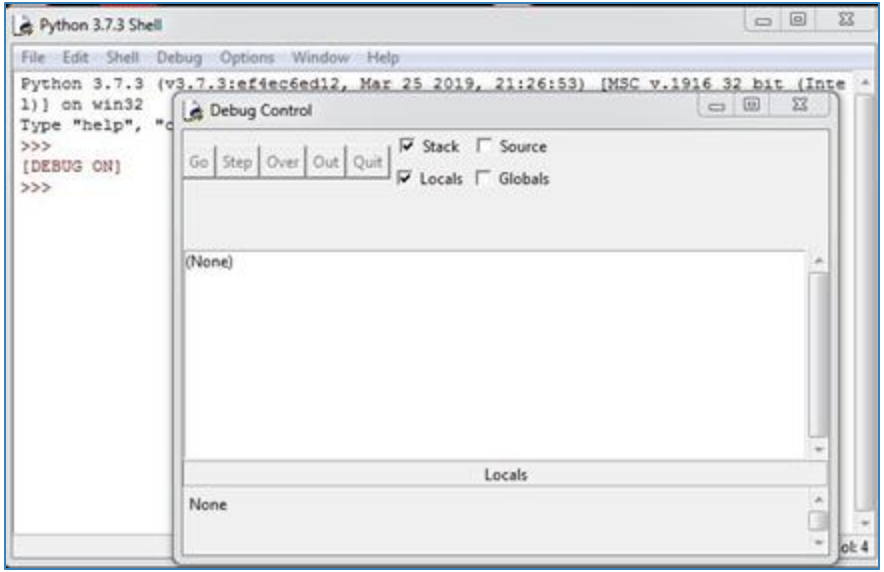
```
Traceback (most recent call last):
  File "C:\Users\A-Level Com\Desktop\Python files\40 03.py", line 3, in <module>
    answer=p/q
ZeroDivisionError: division by zero
>>>
```

Input (p)
Input (q)
answer=p/q

The syntax is correct but an error is found when the program runs. The position of the error is also stated.

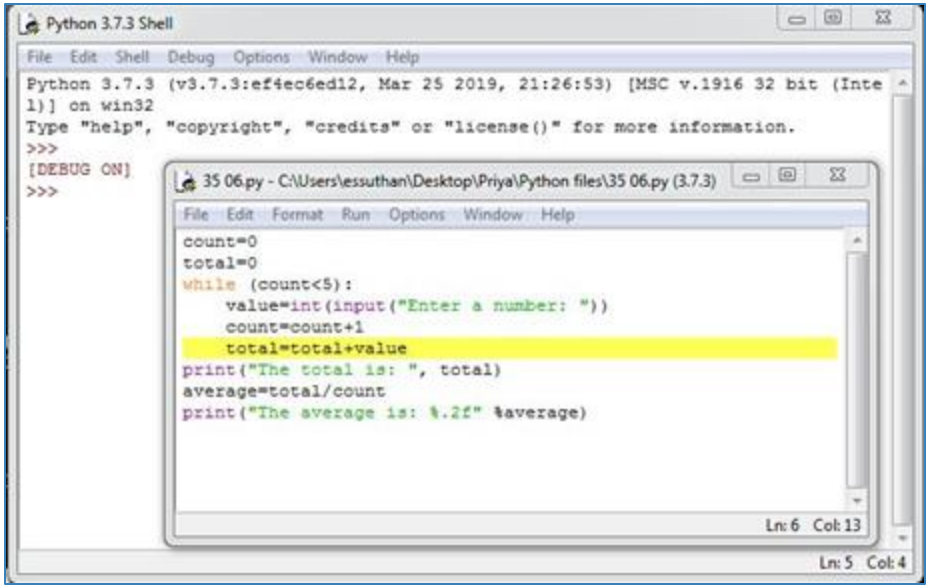
IDE: Debugger

- To check for logic errors, an IDE provides a debugger.
- Debugging is the process of finding and correcting errors in a program.
- Using this option, the value of a variable can be tracked every time it changes.
- The debugger is selected from the debug menu in the Python IDE.
- The statement [DEBUG ON] is printed on the shell.



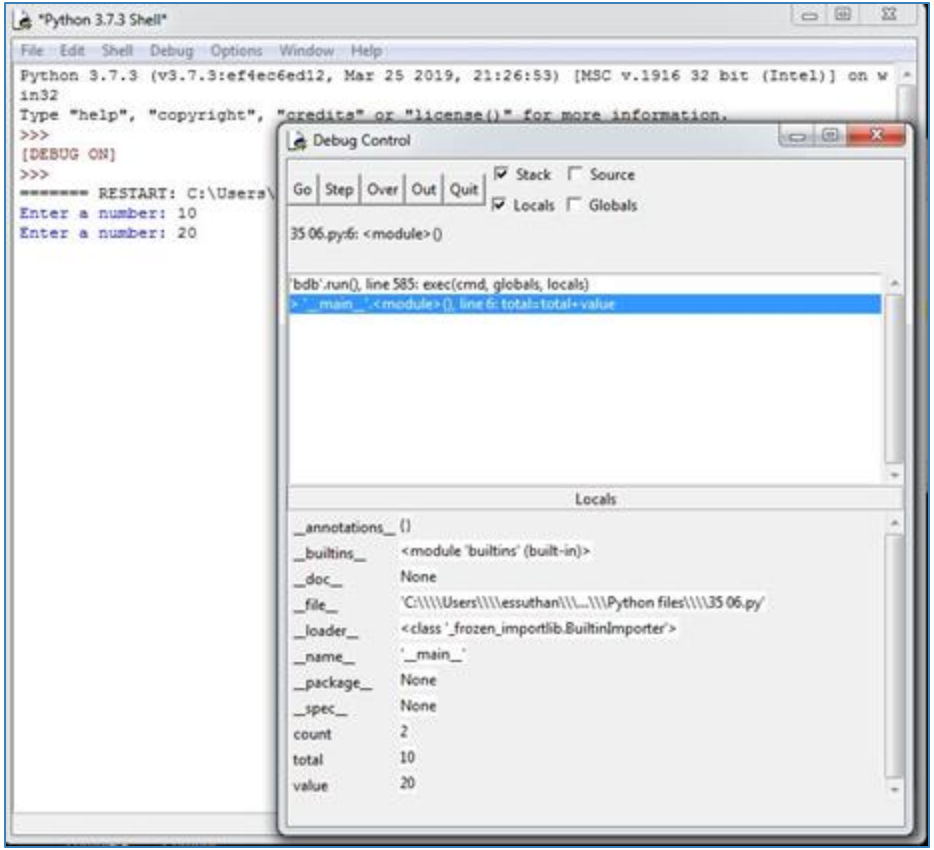
IDE: Breakpoint

- A breakpoint shall be set at any statement in the program by right-clicking and selecting “Set breakpoint”.
- The statement is now highlighted in yellow.
- A breakpoint is a marker that transfers the control from computer to human.
- The execution of the program stops at the breakpoint, and the values of variables can be checked.



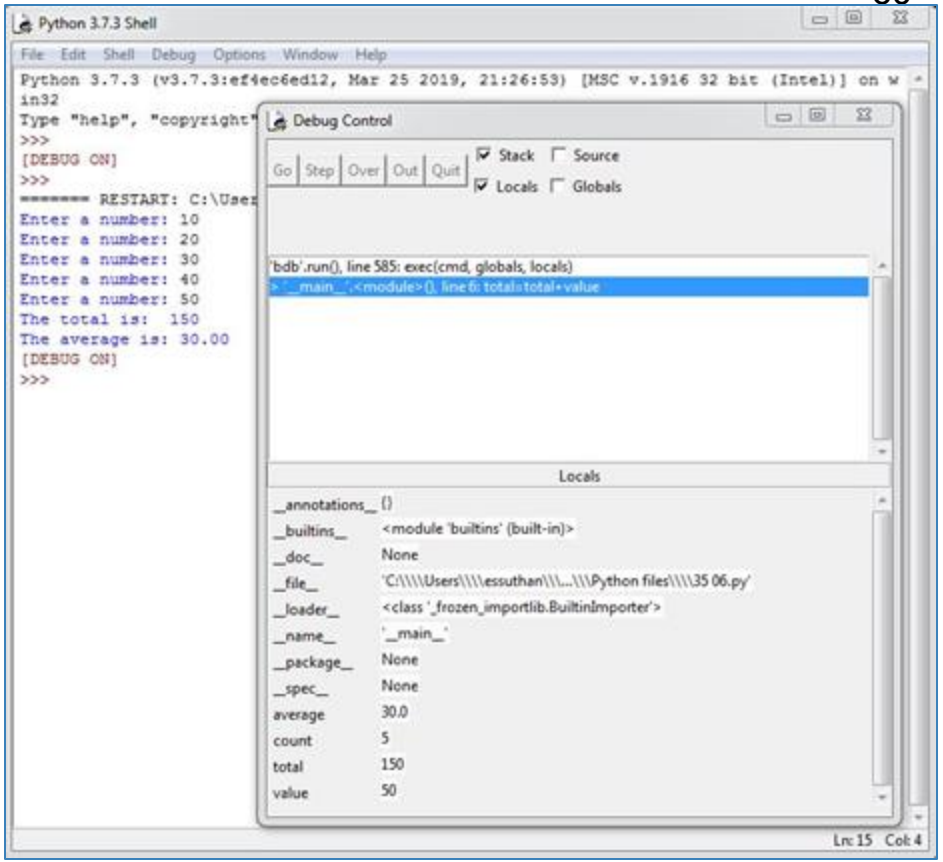
IDE: Breakpoint

- In some IDEs, you may set a watch on a variable to display every time it changes.
- You may also use the step-through option to execute a line at a time.
- Running the program with the breakpoint. Please note the value of variables shown for count=2.
- In case, any of your program produces incorrect results, use this option to track the changes in variables.



IDE: Breakpoint

- The screenshot of the Python shell and debug control after the complete execution of this program is given.



Testing programs

- A set of test data is applied to the algorithm to check its correctness. Types of test data are:
  a) Normal data
  b) Erroneous or abnormal data
  c) Boundary data

Normal data

- A set of test data used together with the expected results is called normal data. For example: For an algorithm calculating the average mark out of 100 scored by five students in a group,
- Normal test data: 76, 80, 63, 80, 90
- Expected result: 77.8
- Because the range is 0 to 100, the normal test data consists of numbers between them (1-99).

Erroneous or abnormal data

- Sets of values that a particular subsystem is expected to reject are called erroneous data. For example: For an algorithm where marks are only positive numbers,
- Erroneous data: -11, ten
- Result: invalid input values

Boundary data

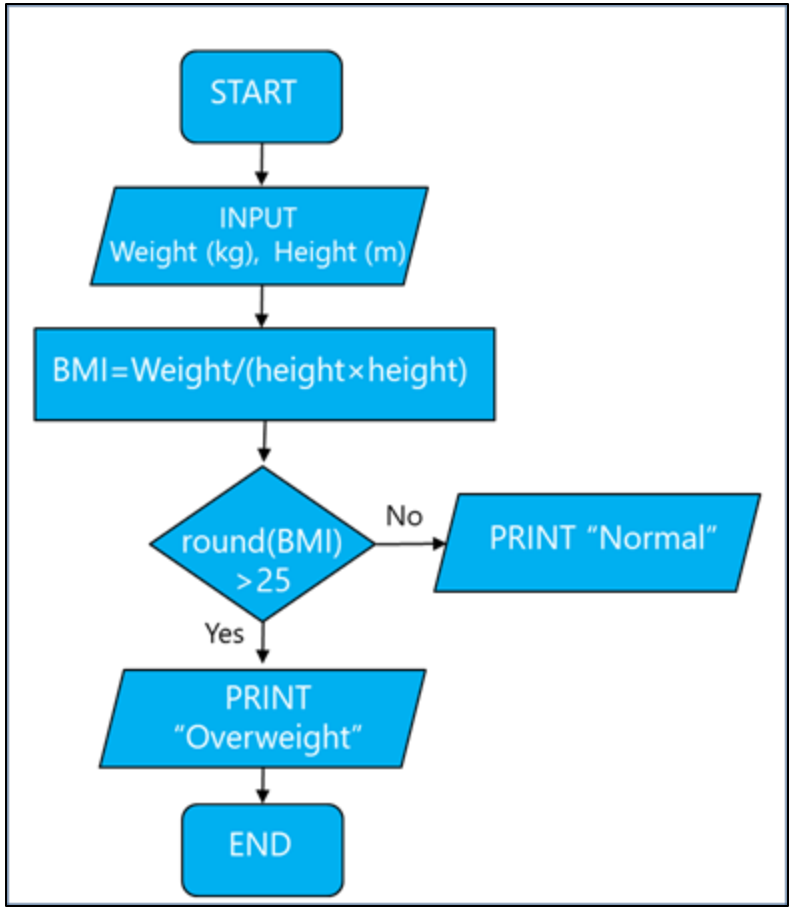
- The largest and smallest values that an algorithm can accept are called boundary data. For example, The minimum and maximum marks that can be scored are 0 and 100.
- Boundary data: 0, 100
- Result: valid input values

Trace table

- A trace table records the value of variables every time it changes in an algorithm.
- Using a trace table, the change of values of a variable due to each operation can be noted. This process is called a dry run.
- Trace tables are widely used to identify logic errors (if any) in an algorithm.

Trace table

- Let us consider the algorithm given here, which calculates and analyses BMI. Weight is entered in kg and height is entered in m.
- The healthy range of BMI is 18.5 to 24.9. A BMI of 25.0 or more is overweight.



Trace table

- A trace table consists of columns representing each variable and a column for output.
- Let us consider this test data (weight, height): (80, 1.70), (70, 1.65), (72, 1.65), (65, 1.55), (55, 1.50), (75, 1.65), (70, 1.75) and (52, 1.44).

Weight (kg)	Height (m)	round(BMI)	Result
80	1.70	28	Overweight
78	1.65	29	Overweight
72	1.65	26	Overweight
65	1.55	27	Overweight
55	1.50	24	Normal
75	1.65	28	Overweight
70	1.75	23	Normal
52	1.44	25	Normal

- A deliberate error was made at the conditional statement.
- The last test data (51.8, 1.44) calculates the BMI to be 25, but the result is normal.
- This data helps us to identify an error in the algorithm.
- In the algorithm, the conditional statement should be BMI ≥ 25.0 instead of BMI > 25.
- In this algorithm, further conditional statements can be included for additional conditions such as obese (BMI ≥ 30.0).

Weight (kg)	Height (m)	round(BMI)	Result
80	1.70	28	Overweight
78	1.65	29	Overweight
72	1.65	26	Overweight
65	1.55	27	Overweight
55	1.50	24	Normal
75	1.65	28	Overweight
70	1.75	23	Normal
52	1.44	25	Normal

Let’s review some concepts

Integrated Development Environment

A tool used to develop programs of a high-level language. It consists of an editor with an interpreter and/or compiler and debugging tools.

Testing

A part of software development to make sure that the system is working as expected. Types of test data:

- Normal data
- Erroneous or abnormal data
- Boundary data

Syntax errors

Programming rules are not followed.

Logic errors

Incorrect or unexpected results

Run-time errors

Unexpected halt or crash

Dry run

The process of recording the value of the variable in an algorithm every time it changes.

Debugger

Setting breakpoints and tracing the change in variables at every step.

Trace table

A trace table records the value of the results of variables at each step of an algorithm. Using a trace table, the change of values of a variable due to each operation can be noted.

Software maintenance

Corrective maintenance, adaptive maintenance & perfective maintenance.

2.

Activity-1

Duration: 10 minutes

1. A Python program is given. Test data is: 1, 2, 3, 4, 5, 6, -1. Complete the trace table.

a	b	c	d
...	...	...	...

```
a=0
b=0
c=0
d=int(input('Enter a number: '))
while (d>0):
    a=a+d
    c=c+1
    d=int(input('Enter a number: '))
b=a/c
print(b)
```

Activity-2

Duration: 10 minutes

1. Complete the trace table for the algorithm. The input value is 7.

```
input a
for b in range(1,a)
    c=a-b
    d=5*c-b
end for
print(b)
```

a	b	c	d
...	...	...	...

3.

End of topic questions

1. What are run-time errors, logic errors and syntax errors? Give examples for both. Explain how are these identified and corrected in a program.
2. A Python program is given. The test data is: 1, 2, 3, 11, -1. Create and complete the trace table for this test data.

```
a=1
b=0
c=0
d=int(input('Enter a number: '))
while (d>0):
    a=a*d
    c=c+1
    d=int(input('Enter a number: '))
b=a%c
print(b)
```

4. The algorithm calculates the total and average of numbers entered by the user.

   Create a trace table and dry run this algorithm using test data for count=6. Choose your own values for number.

5. What are the three main software maintenance activities? How are they different from each other?

```
print ('How many numbers?')
input count
total ← 0
j← count
while (count >0) do
   print('Enter a positive
   number')
   input number
   count =count -1
   total=total + number
endwhile
average=total/j
print(total, average)
```

Let’s review some concepts

Translators

A utility program that translates the program written by programmers and codes in assembly language into binary form, which is understandable by the computer.

Compiler

Translates a high-level language code into machine code.

Interpreter

Executes one statement of high-level language at a time and then proceeds to the next.

Assembler

A computer program that translates a program written in an assembly language into machine code.

Stages of compilation

Lexical analysis

Syntax & semantic analysis

Code generation & optimisation

Dynamic Link Libraries

Dynamic linked library is a shared library of sub-programs. These sub-programs are used by programmers in their programs whenever required.

Linkers & Loaders