



Programming and Problem Solving

SPPU | 2020
First Year Engineering



Python
Interpreter

Gaurav F. Jumnake

SCAN. STUDY. SCORE.



JOURNEY TO A SUCCESSFUL IN-SEM EXAM

Total Marks:30

UNIT-1

19
Topics

1st MILESTONE

1. Introduction to Problem Solving
2. Problem solving with computers
3. Difficulties with problem solving
4. Problem solving aspects
5. Problem Solving Strategies
6. Problem solving tools

2nd MILESTONE

7. Introduction to python
8. Features of python
9. Applications of python
10. Writing First Python Code

3rd MILESTONE

11. Python fundamental
12. Reserved words
13. Identifiers
14. Variables
15. Literal constants
16. Comments
17. Indentation
18. Operators
19. Data Types

UNIT-2

16
Topics

1st MILESTONE

1. Decision making statements
2. If Statement
3. If-else statement
4. If-elif-else statement
5. Nested if

2nd MILESTONE

6. Iterations
7. While loop
8. For loop
9. Python Loop control statements
10. Break statement
11. Continue statement
12. Pass statement
13. else - Statement with Loop

3rd MILESTONE

14. List
15. Tuple
16. Dictionary



Problem Solving, Programming and Python Programming

1. Introduction to Problem Solving

Problem

It is a situation which must be overcome by finding out appropriate applicable solutions OR by achieving expected goals.

Steps to Solve Problem

By using following approach we can solve any type of problem.

- (i) Identify the problem
 - (a). You need to be clear about what exactly the problem is.
 - (b). If problem is not identified properly, getting proper solution is impossible.
- (ii) Understand the problem
 - (a). What exactly are we trying to solve?
 - (b). Here problem should be defined properly so that it helps in understanding the problem more.
- (iii) List out alternative ways to solve the problem
 - (a). This step is very crucial because until and unless we explore all options, finding a correct solution is next to impossible.
 - (b). This step needs a lot of attention.
 - (c). If we explore all alternative possible chances of getting the problem solved is higher.
- (iv) Select the best way
 - (a). This is an important step.
 - (b). Selecting proper way from alternatives require a lot of study and need to study consequences as well.
 - (c). If we don't choose correct alternative, it may lead in incorrect solution of the problem.
- (v) Write down the procedure or steps to solve the problem
 - (a). Here we need to list out the steps which should be followed so that anyone can solve the problem.
 - (b). Be sure not to skip any step.
 - (c). This should be in detail and precise at the same time.
- (vi) Evaluate the solution
 - (a). Now this step is just like testing in development.
 - (b). We need to roll back and check whether we have solved the problem correctly or not.

Example: I am always late for college

- (i) Identification: getting late for college.
- (ii) Understanding: (List out probable reasons of or getting late) (Sleeping late, waking up late, Transportation, etc.)
- (iii) List out alternative ways – (Solutions)

- (a). Sleeping early at home.
 - (b). Waking up early.
 - (c). Waking alarm.
 - (d). Resolving travelling issue.
- (iv) Select the best way – Sleeping early in night.
- (v) List out the procedure
- (a). Having dinner as early as possible.
 - (b). Avoid use of mobile phones in night.
 - (c). Completing homework as early as possible.
 - (d). Then going early to bed.
- (vi) Evaluate the solution: Reached college on time.

Example: Not able to sleep

- (i) Identification: Not able to sleep.
- (ii) Understanding: (List out probable reasons of lack of sleep)
(erratic schedule, sleeping in the afternoon, etc.)
- (iii) List out Alternative
- (a). Fix bed time and wake up time.
 - (b). Limit screen time.
 - (c). Include meditation and breathing techniques.
 - (d). Fix a night time routine.
 - (e). Have an early dinner.
- (iv) Select the best way
- (a). Fix bed time and include meditation and breathing techniques.
- (v) List out procedure
- (a). After finishing dinner take a walk to ease in digestion.
 - (b). An hour before fixed bed time, get rid of all gadgets.
 - (c). Try to make your room darker and cooler.
 - (d). Include book reading for at least 15 minutes.
 - (e). Perform meditation and breathing techniques which will calm the mind and induce sleep.
- (vi) Evaluate the solution: Better sleep schedule.

Example: birthday celebration in covid times

- (i) Identification: Birthday in corona.
- (ii) Understanding: (List out probable reasons of lack of sleep)
(infection outside, lockdown, been at home for long)
- (iii) List out Alternative
- (a). Find dine in option.
 - (b). Go for shopping.

- (c). Find a place where limited number of people can be accommodated and go for day trip.
- (d). Go for a ride and order food at home.
- (e). Decorate and order food at home and have home quarantine birthday celebration.
- (iv) Select the best way
 - (a). Find a place where just your family can go for a day out and spend a day out so that everyone is safe and can enjoy as well. It should be a property owned and secluded from others.
- (v) List out procedure
 - (a). Find places where you can visit which are safe and will give you a break in these quarantine times.
 - (b). Check if that place is available.
 - (c). Use your vehicle for transport and carry all safety measures.
 - (d). Enjoy your day out with your family.
 - (e). Return back with beautiful memories of your birthday.
- (vi). Evaluate the solution: birthday celebrated!!!

Types of Problem

Problem can be categorized into two types depending on how the solution is found out or the approach which is followed to get that solution.

- (i) Algorithmic solution
 - (a). These problems are solved by following some procedure or steps.
 - (b). For example: baking a cake. It can be done by following a series of instruction and we can expect results.
 - (c). Here following some predefined process needs to be done.
 - (d). Here solution is quite straight forward.
- (ii) Heuristic solution
 - (a). These types of problems solved using knowledge based i.e. by collecting information about the problem.
 - (b). For example: Expanding one's business.
 - (c). In this type of approach there is no straight forward defined path which we can follow to solve a problem.
 - (d). We need to build that path based on trial and error.
 - (e). In this approach experience and knowledge is very important.

2. Problem Solving With Computer

In case of computer problem solving is based on following three factors

- (i) Solution: Instructions to solve the problem.
- (ii) Result: A computer generated output or expected outcome.
- (iii) Program: These are actual computerized instructions which when processed generates output.

3. Difficulties in Problem Solving

- (i) Many problems to solve but don't know how
- (ii) Lack of proper understanding about problem
- (iii) A fear to make bold decisions
- (iv) Incomplete solution
- (v) Illogical sequencing of instructions
- (vi) Ability to write down instructions

4. Problem Solving Aspects

- (i) Problem definition phase: It tells us what must be done
- (ii) Getting started to solved problem
- (iii) Use of specific examples
- (iv) Working backward from solution

5. Top down approach

- ✚ This approach starts with a problem which are again divided into sub-problems until a solution is found.
- ✚ Here we begin with the root and add branches based on the complexity of the problem.
- ✚ It is like a tree structure.
- ✚ In this approach we start with an abstract design and then at each step, this design is more refined.
- ✚ Each sub problem gives much more details about solution.
- ✚ This approach was concept of function to form sub parts.
- ✚ We have program at the root and we add different functions to simplify the problems.
- ✚ These functions are called by the program itself. Hence it's called a top down approach
E.g. C language

6. Problem solving tools

Different tools used in problem solving are

- ✚ Algorithm
- ✚ Flowchart
- ✚ Pseudo code

6.1 Algorithm

- ✚ These are simple English like statements.
- ✚ These statement sometimes may contain symbols ,expressions or variables.
- ✚ These statements are written in logical sequence so that some task will get performed.
- ✚ Algorithm should be finite, unique and descriptive.
- ✚ Algorithmic steps must be problem specific and relevant.
- ✚ Algorithm starts with keyword "start" and ends with "stop".
- ✚ To every algorithmic step some numbers must be assigned in sequence.
- ✚ It should not be programming language dependent independent.

Example 1: Write a algorithm to perform addition of two variables

- (1) Start
- (2) Accept the input lets x and y
- (3) Perform addition i.e. $(c = x + y)$
- (4) Display the result i.e. c
- (5) Stop

Example 2: Write a algorithm to calculate area of circle

- (1) Start
- (2) Accept the input (i.e. r)
- (3) Calculate the area
 $area = 3.14 * r * r$
- (4) Print the area
- (5) Stop

Example 3: Write a algorithm to calculate roots of quadratic equation

- (1) Starts
- (2) Accept the input (i.e. a, b, c)
- (3) Calculate the roots
3.1) $x = b * b - 4 * a * c$
3.2) $y = x * * 0.5$
3.3) $R1 = (-b + y) / 2 * a$
3.4) $R2 = (-b - y) / 2 * a$
- (4) Print R1 and R2
- (5) Stop

Example 4: Write a algorithm to calculate gross salary of employee if dA is 80% of basic salary, HRA is 50% of basic salary and fixed TA of 10,000 Rs




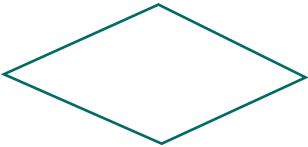

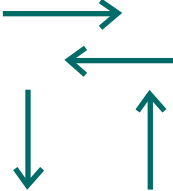

- (1) Start
- (2) Accept the input (i.e. Basic Salary [BS])
- (3) Calculate the salary
 $dA = (80 / 100) * BS$
 $HRA = (50 / 100) * BS$
 $TA = 10,000$
 $Sal = BS + DA + HRA + TA$
- (4) Print the value of Sal
- (5) Stop

Example 5: Write a algorithm to swap values of two variables

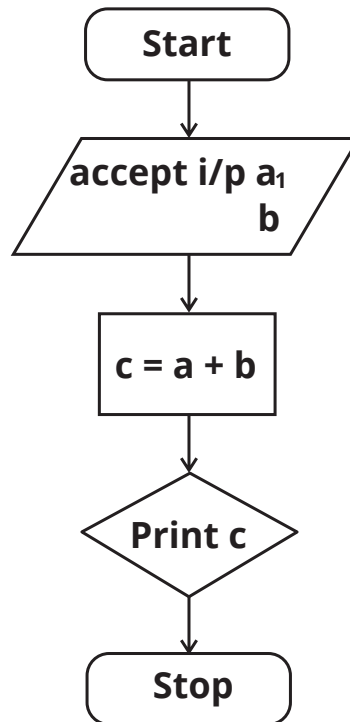
- (1) Start
- (2) Accept the input i.e. $x = 10, y = 8$
- (3) Calculate the swap variables
 $z = x$
 $x = y$
 $y = z$
- (4) Print new values of x, y
- (5) Stop

6.2 Flow Chart

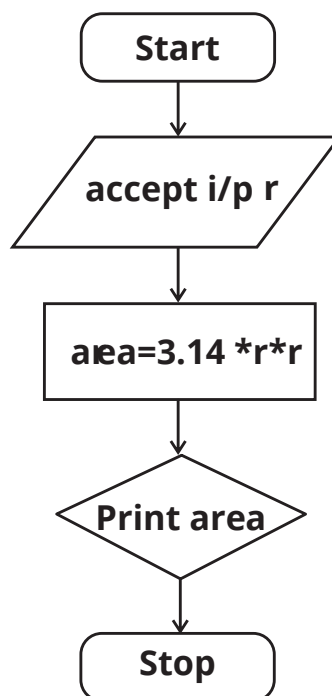
- ✚ It is an diagrammatic representation of Algorithm.
- ✚ Flow chart gives details about how the program and control flow.
By using this diagrammatic representation will help us to understand the problem more clearly.
- ✚ Flow chart help us to visualize complex and lengthy problems.
- ✚ Flow chart is represented using symbols as follows.

Symbol Name	Symbol	Description
1. Start / Stop		It is used to represent start and end of algorithm
2. Input / Output box		It is used to represent input output instructions i.e. accept, read, input, print, display
3. Process Box		Represents process i.e. calculation
4. Decision Box		It represents conditional statements i.e. statements with relational operators
5. Module call / Procedure Call		It represents sub algorithm, function call
6. Flow lines		They are used to connect components of flow chart
7. Connector		It is used to connect broken parts of flowchart

Example 1: Draw a flow chart for addition of two variables.



Example 2: Draw flow chart for area of circle.



6.3 Pseudo Code

- ☞ These are intermediate instructions which are constructed using algorithm and some of the instructions from actual program.
- ☞ Pseudo code gives details about the variables which are used, expressions and conditions.

Example 1: Write a algorithm and pseudo code to compare two variables.

Algorithm

- (1) Start
- (2) Accept the input
- (3) Compare the variables
if values of a and b are same then print
 "They are equal"
if not same then print
 "They are different"
- (4) Stop

Pseudo Code

- (1) Start
- (2) Accept input a, b
- (3) Compare variables
If $a = b$, then print
 "They are equal"
If $a \neq b$ then print
 "They are different"
- (4) Stop

Exercise - 1.1

Question Based on 1st Milestone

Click or scan using
FORTFLAG app
for detailed
solution.



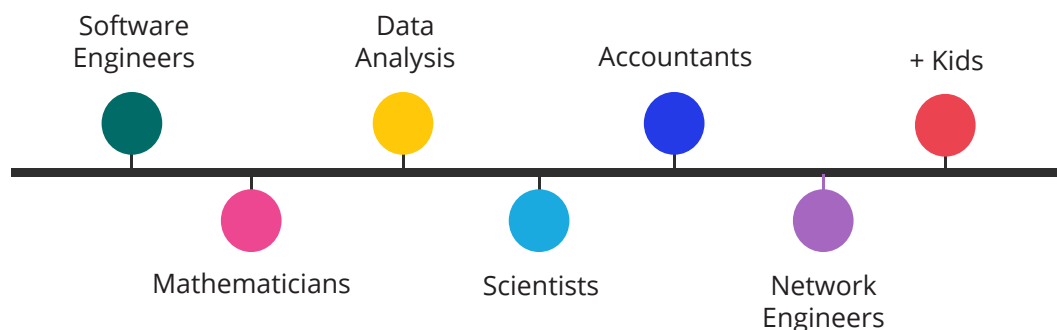
- Q 1. What are six problem solving step?
- Q 2. What is problem? List out six problem solving steps.
- Q 3. Explain six problem solving steps with example.
- Q 4. What is top down approach.
- Q 5. What are the difficulties in problem solving.
- Q 6. What is flowchart? Explain with suitable example.
- Q 7. What is algorithm? Explain with suitable example.
- Q 8. What is pseudo code? Explain with suitable example.
- Q 9. Write short note on
 - (a). Algorithm
 - (b). Flowchart
 - (c). Pseudo code

7. Introduction to Python

- Python is one of the programming language like C, C++, Java etc.
- It is interpreted, object oriented and high-level language.
- Python was started in late 1980s by Guido Van Rossum at CWI in Netherland.

- In February 1991, van Rossum published first labeled version 0.9.0.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages: ABC language, Modula-3.

Who are using Python?



16 Famous Company that uses PYTHON



8. Features of Python

(i) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language. As python community offer fast and effective support to the developers and programmers, it brings down the development time and cost. With lots of module available, web or application development become faster when user chooses Python.

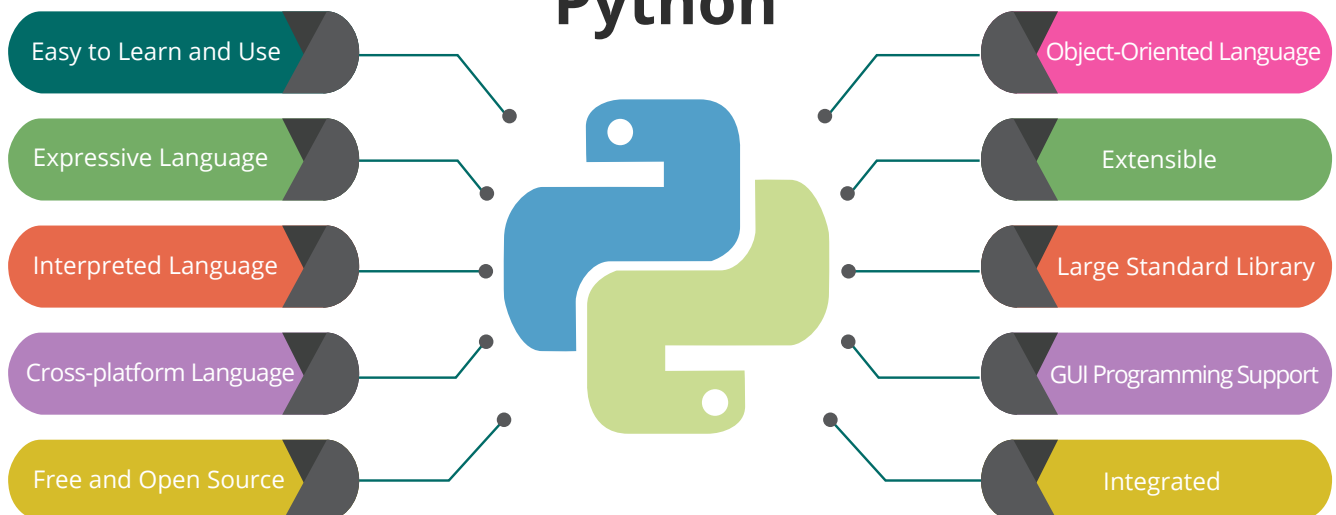
(ii) Expressive Language

Python language is more expressive means that it is more understandable and readable.

(iii) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

10 Reason to learn Python



(iv) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

(v) Free and Open Source

Python language is freely available at official web address. The source-code is also available. Therefore it is open source.

(vi) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

(vii) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

(viii) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development. so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit testing, web browsers etc.

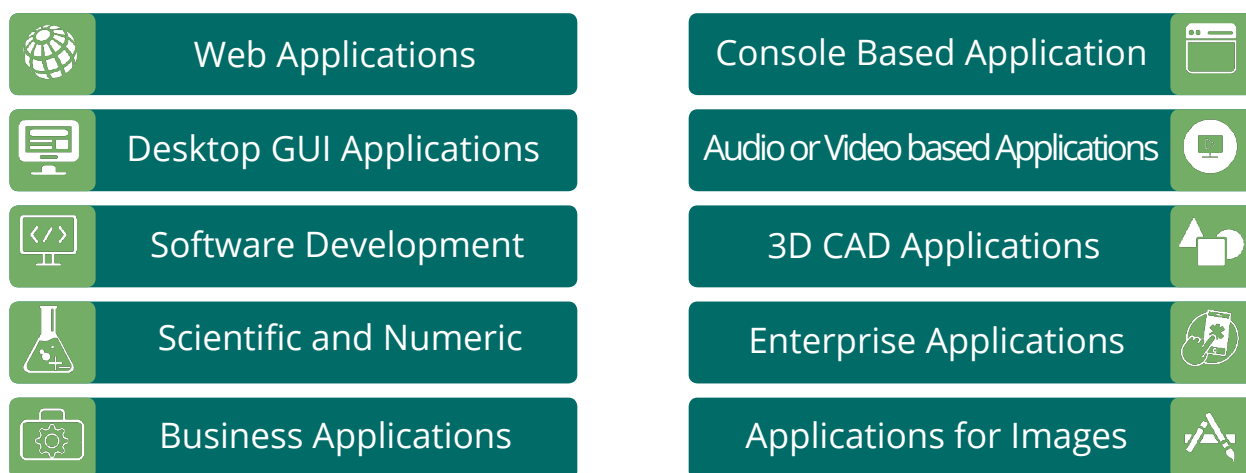
(ix) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

(x) Dynamically Typed Language

Python is dynamically-typed language. That means the type (for example- int, double, long etc) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

9. Applications of Python



(i) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautiful Soup, Feed parser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and develop web based applications. Some important developments are: Python WikiEngines, Pocoo, PythonBlog Software etc.

(ii) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

(iii) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

(iv) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

(v) Business Applications

Python is used to build Business applications like ERP and e-commerce systems. Python is a high level application platform.

(vi) Console Based Application

We can use Python to develop console based applications. For example: IPython.

(vii) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

(viii) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

(ix) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

(x) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek

10. Writing First Python Code

Now we know the importance of python and its application in various field, in this section, we will write our 1st python code. But before that please note following point.

- (i) Python cannot be learnt by reading book. Python must be learnt by writing and practicing code on computer.
- (ii) Student must attend all practical session in college and practice the program.
- (iii) Watch our video tutorial “How to install the Python on your computer”.

We will start with one of the simplest codes in Python syntax - the print Statement. Whenever we learn a new language, it is an age-old tradition to start by displaying the text “Hello World” on the screen.

Whatever we need to print is encapsulated in the parentheses following the print keyword. Let’s try printing “Hello World” on the Practice terminal. The text Hello World is bounded by quotation marks because it is a string or a group of characters. We will discuss about string in detail later.

●● INPUT CODE

```
1 print("Hello World")
```

●● OUTPUT

```
Hello World
```

Next, we'll print a few numbers. Each call to print moves the output to a new line:

●●● INPUT CODE

```

1 print (50)
2 print (1000)
3 print (3.142)

```

●●● OUTPUT

```

50
1000
3.142

```

We can even print multiple things in a single print command; we just have to separate them using commas.

●●● INPUT CODE

```

1 print (50, 1000, 3.142, "hello world")

```

●●● OUTPUT

```

50 1000 3.142 hello world

```

Exercise - 1.2

Problem Based on 2nd milestone

Click or scan using
FORTFLAG app
for detailed
solution.



- Q 1. What are the features of python?
- Q 2. List out the applications of python
- Q 3. Write short note on history of python

11. Python Fundamental

In this section we will create the foundation for writing the python program. We will try to understand the fundamental of python programming language by understanding the function and role of each component.

12. Reserve Words

- ☞ These are also known as key words.
- ☞ These are the words whose meaning is predefined by Python interpreter and cannot be used like general purpose words.
- ☞ These are basic building blocks of a program used to construct instructions.
- ☞ Keywords cannot be used as variable name or function name or as an identifier.
- ☞ There are 33 reserve words available in python some of them are :
False, True, if, else, elif, While, For break, pass, continue, def, class, is, is not, return, in, not in, and, or, import, global, from

13. Identifiers

- ☞ These are naming conventions used in python.
- ☞ Identifier can be anything which is user defined.
- ☞ Identifiers always start with a character or underscore (_)
- ☞ A Python identifier is a name used to identify a variable, function, class, module or other object.
- ☞ An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- ☞ Use of blank space is not allowed in the identifiers instead we can use underscore.
- ☞ Identifier may contain digits, strings. But after underscore or character.
- ☞ Python does not allow punctuation characters such as @, \$, and % within identifiers.
- ☞ Python is a case sensitive programming language. This means, Variable and variable are not the same.
- ☞ Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- ☞ Starting an identifier with a single leading underscore indicates that the identifier is private.
- ☞ Starting an identifier with two leading underscores indicates a strongly private identifier. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

14. Variables

- ☞ These are the entities whose value is not fixed.
- ☞ Used for storing and manipulating data.
- ☞ Variables are nothing but identifiers so the rules used for identifier are also applicable here.
- ☞ In python variables need not to be declared and can be used directly.

Example: `a = 10`
 `name = "zeal"`

- ☞ Python variables can be reassigned, in that case previous content of variable will be replaced by new one.

- String variables can be combined together using (+) operator which is known as concatenation. It is possible only when both the items are of same type.

Example: `print("zeal" + "pune")`
`print("zeal" + 41)` ×

- A single value can be assigned to multiple variable.

Example: `a = b = c = 10`

- Multiple values can be assigned to multiple variables in single declaration `x, y, z = 4, 3.1 "Pune"`.

15. Literals Constants

- Literals are nothing but constants in python.
- These are the data items which are defined directly or through some variable.

Example: `a = 4`
`print (4 + 5)`
`print ("msg")`

- Depending on data literals can be following types

15.1 String literals

- String literals can be formed by enclosing a text in the quotes. We can use both single as well as double quotes for a String.

Example: `"Aman"` , `'12345'`

15.2 Types of Strings

There are two types of Strings supported in Python:

- Single line String- Strings that are terminated within a single line are known as Single line-Strings.

Example: `>>> text1='hello'`

- Multi line String- A piece of text that is spread along multiple lines is known as Multiple line String.

There are two ways to create Multiline Strings

(i) Adding back slash at the end of each line

Example:

```
1. >>> text1='hello\
2. user'
3. >>> text1
4. hellouser'
5. >>>
```

(ii) Using triple quotation marks

Example:

```
1. >>> str2="""welcome
2. to
3. SSSIT"""
4. >>> print str2
5. welcome
6. to
7. SSSIT
8. >>>
```

Numeric literals

Numeric Literals are immutable. Numeric literals can belong to following four different numerical types.

- ☛ **Int(signed integers)-** Numbers(can be both positive and negative) with no fractional part.
Example: 100
- ☛ **Long(long integers)-** Integers of unlimited size followed by lowercase or uppercase L.
Example: 87032845L
- ☛ **Float(floating point)-** Real numbers with both integer and fractional part eg: -26.2.
- ☛ **Complex(complex)-** In the form of $a+bj$ where a forms the real part and b forms the imaginary part of complex number. eg: $3.14j$

Boolean literals

- ☛ A Boolean literal can have any of the two values: True or False.

Special literals

- ☛ Python contains one special literal i.e., None.
None is used to specify to that field that is not created. It is also used for end of lists in Python.

Example:

```
1. >>> val1=10
2. >>> val2=None
3. >>> val1
4. 10
5. >>> val2
6. >>> print val2
7. None
8. >>>
```

Literal Collections Collections such as tuples, lists and Dictionary are used in Python.

List

- ☛ List contains items of different data types. Lists are mutable i.e., modifiable.
- ☛ The values stored in List are separated by commas(,) and enclosed within a square brackets([]). We can store different type of data in a List.
- ☛ Value stored in a List can be retrieved using the slice operator([] and [:]).
- ☛ The plus sign (+) is the list concatenation and asterisk(*) is the repetition operator.

Example:

```

1. >>> list=['aman',678,20.4,'saurav']
2. >>> list1=[456,'rahul']
3. >>> list
4. ['aman', 678, 20.4, 'saurav']
5. >>> list[1:3]
6. [678, 20.4]
7. >>> list+list1
8. ['aman', 678, 20.4, 'saurav', 456, 'rahul']
9. >>> list1*2
10. [456, 'rahul', 456, 'rahul']
11. >>>

```

16. Comments

- ☛ These are non-executable statements written in a program.
- ☛ They are for users or programmers understanding.
- ☛ It gives information about that statement or a program.
- ☛ Comments are optional and can be declared anywhere within a program.
- ☛ In python the comments can be declared as
 - (i) Single line comment vlt is declared using hash (#) e.g. #This is a comment, too.
 - (ii) Multiline comments

Declared using pair of triple quote

```
'''      '''
```

Example:

Programming in python is fun.

Sample program to show use of comments in python"

```
a=10 #10 is assigned to variable a
```

```
b=20
```

```
print(a+b) #print addition of a and b
```

17. Indentation

- ☞ Most of the programming languages use curly braces ({ }) to show the block of statement.
- ☞ But python uses indentations to show block of statements or sub-statements.
- ☞ Indentation highlights block of statement.
- ☞ All statement which are part of some other statement are written in a single column with some space from the left.
- ☞ Generally one tab or four spaces are included.

Example:

INPUT CODE

```
1. a=int(input("Enter value of a"))
2. b=int(input("Enter value of b"))
3. if a > b:
4.     print("a is greater than b")
5. else:
6.     print("b is greater than a")
```

OUTPUT

```
Enter the number4
Enter the number5
5 is greater than 4
```

Properly indented program

INPUT CODE

```
1 if a > b:
2     print("a is greater than b")
3 else:
4     print("b is greater than a")
```

Without indentation gives error

OUTPUT

```
File "main.py", line 4
    print(a, "is greater than", b)
    ^
IndentationError: expected an indented block
```

18. Operators

- ☞ Operators are used to process data.
- ☞ Depending on the use the operators in python are as follows

(i) Arithmetic Operators

+	Addition	
-	subtraction	
*	Multiplication	
/	Division	
**	Exponent	Calculate rest to power of variable i.e.xy x2
//	Floor division	Returns rounded off result
%	Reminder operator	Returns the reminder of division

●●● INPUT CODE

```

1  a = int(input("Enter the number"))
2  b = int(input("Enter the number"))
3  add = a + b
4  print("Addition is =", add)
5  sub = a - b
6  print("Subtraction is =", sub)
7  mul = a * b
8  print("Multiplication is =", mul)
9  div = a / b
10 print("Divison is =", div)
11 div1 = a // b
12 print("Floor division is =", div1)
13 rem = a % b
14 print("Remainder is =", rem)
15 expo = a ** b
16 print("a to the Power b is =", expo)

```

●●● OUTPUT

```

Enter the number25
Enter the number15
Addition is = 40
Subtraction is = 10
Multiplication is = 375
Divison is = 1.6666666666666667
Floor division is = 1
Remainder is = 10
a to the Power b is = 931322574615478515625

```

(ii) Relational operators

Note:- When two operands are compared evaluated as either true or false. If True the binary value is 1, if false the binary value is 0.

True → 1

False → 0

Example:

>	Greater than
<	Less than
>=	Greater than equal to
<=	Less than equal to
==	Equal to
!=	Not equal to

INPUT CODE

```
1. x = int (input ("Enter the number"))
2. y = int (input ("Enter the number"))
3. print ("x > y is", x > y)
4. print ("x < y is", x < y)
5. print ("x >= y is", x >= y)
6. print ("x <= y is", x <= y)
7. print ("x == y is", x == y)
8. print ("x != y is", x != y)
```

OUTPUT

```
Enter the number20
Enter the number30
x > y is False
x < y is True
x >= y is False
x <= y is True
x == y is False
x != y is True
```

(iii) Logical operators

Note:- non zero numbers are evaluated as true while zero as false.
Operates number as whole no binary conversion is required.

Example:

and	Logical and	If both sides of expression are true then returns true
or	Logical or	If any of the side is true then returns true
not	Logical not	Gives negation of given output

INPUT CODE

```
x = True
y = False
print ("x and y is", x and y)
print ("x or y is", x or y)
print ("not y is", not y)
```

OUTPUT

```
x and y is False
x or y is True
not y is True
```

(iv) Bitwise operators

a = 4 (whose binary is)= 0100

b = 10 = 1010

&	Bitwise and	Decimal numbers are converted into binary and then 'and' operations is performed on that binary data a & b = 0100 & 1010 = 0000 (zero)
 	Bitwise or	Decimal numbers are converted into binary and then 'or' operations is performed on that binary data
~	Bitwise not	Decimal numbers are converted into binary and then 'not' operations is performed on that binary data ~ b = ~ 1010 = 1110101 (-14)
^	Bitwise xor	Decimal numbers are converted into binary and then 'xor' operations is performed on that binary data a ^ b = 0100 ^ 1010 = 1110 (14)
<<	Left shift	Shifts bit of given binary number to left by those many number of bits and insert zero at empty location 4 << 2 => 0100 => 1000 (1st shift) => 0000 (2nd shift)
>>	Right shift	Shifts bit of given binary number to right by those many number of bits and insert zero at empty location 4 >> 2 => 0100 => 0010 (1st shift) => 0001 (2nd shift)



Bitwise operators process individual bits of data.



Numbers are converted into binary form and then processed by bitwise operators.

●●● INPUT CODE

```

1 x = int(input("Enter the first number"))
2 y = int(input("Enter the second number"))
3 print("x & y is", x & y)
4 print("x | y is", x | y)
5 print("x ^ y is", x ^ y)
6 print("~ x is", ~ x)
7 print("x << y is", x << y)
8 print("x >> y is", x >> y)

```

●●● OUTPUT

```

Enter the first number 4
Enter the second number 10
x & y is 0
x | y is 14
x ^ y is 14
~ x is -5
x << y is 4096
x >> y is 0

```

(v) Assignment operators

=	Assigns values which is on right side to left side .	a = b , a = 10
+=	It performs addition with both operands and stores the result in a operand present on left of expression.	a += b is equivalent to a = a + b
-=	It performs subtraction with both operands and stores the result in a operand present on left of expression.	a -= b is equivalent to a = a - b
*=	It performs multiplication with both operands and stores the result in a operand present on left of expression.	a *= b is equivalent to a = a * b
/=	It performs division with both operands and stores the result in a operand present on left of expression.	a /= b is equivalent to a = a / b
%=	It performs division with both operands and stores the remainder in a operand present on left of expression.	a %= b is equivalent to a = a % b
**=	It will calculate power and will stores the result in a operand present on left of expression.	a **= b is equivalent to a = a * b
//=	It performs division with both operands and stores the rounded off result in a operand present on left of expression.	a //= b is equivalent to a = a // b

(v) Assignment operators

=, +=, -=, *=, /=, %=

Example:

- a = b
- a += b a = a + b
- a -= b a = a - b
- a *= b a = a * b
- a /= b a = a / b
- a %= b a = a % b

INPUT CODE

```
1. x = int(input("Enter the first number"))
2. y = int(input("Enter the second number"))
3. x += y
4. print ("x += y is", x)
5. x -= y
6. print ("x -= y is", x)
7. x *= y
8. print ("x *= y is", x)
9. x /= y
10. print ("x /= y is", x)
11. x //= y
12. print ("x //= y is", x)
13. x %= y
14. print ("x %= y is", x)
15. x **= y
16. print ("x **= y is", x)
```

OUTPUT

```
Enter the first number10
Enter the second number4
x += y is 14
x -= y is 10
x *= y is 40
x /= y is 10.0
x //= y is 2.0
x %= y is 2.0
x **= y is 16.0
```

(vi) Membership operators

These operators are used to check presence of any value or data in given range.

(a) in – returns true if that item exists in a list or a range.

(b) not in – returns true if that item does not exists in given list or a range.

Example:**INPUT CODE**

```
1 msg = "Welcome to programming with python"
2 data = [1,2,3,4,5]
3 print ('to' in msg)
4 print ('u' in msg)
5 print ('u' not in msg)
6 print (3 in data)
7 print (3 not in data)
```

●●● OUTPUT

```
True
False
True
True
False
```

(vii) Identify operator

- ☞ These operators check if given items are identical and equal.
- ☞ In python following identity operators are used
 - (a) is – returns true if given elements are equal and identical (identical means both objects are having same reference location or memory location).
 - (b) is not – returns true if given elements are neither equal nor identical.

Example:

●●● INPUT CODE

```
1 x1 = 5           #nonmutable
2 y1 = 5           #nonmutable
3 x2 = "hello"     #nonmutable
4 y2 = "hello"     #nonmutable
5 x3 = [1,2,3]     #mutable
6 y3 = [1,2,3]     #mutable
7 print(x1 is y1)
8 print(x2 is y2)
9 print(x3 is y3)
10 print(x1 is not y1)
11 print(x2 is not y2)
12 print(x3 is not y3)
```

●●● OUTPUT

```
True
True
False
False
False
True
```

Operator precedence

- ☞ If an expression consisting of multiple operators then result of such expression is calculated using operator precedence.
- ☞ It gives hierarchy in which these operators will be processed.

Processing sequence	Operator & Description
1	** Exponentiation (raise to the power)
2	~ + - Complement, unary plus and minus
3	* / % // Multiply, divide, modulo and floor division
4	+ - Addition and subtraction
5	>><< Right and left bitwise shift
6	& Bitwise 'AND'
7	^ Bitwise exclusive 'OR' and regular 'OR'
8	<= <>>= Comparison operators
9	<> == != Equality operators
10	= %= /= //= -= += *= **= Assignment operators
11	is is not Identity operators
12	in not in Membership operators
13	not or and Logical operators

19. Data Types

- It gives the information about data to interpreter if required.
- Generally python do not require any variable to be declared as per the data assign to variables interpreter automatically decides data type of variables.

(i) Numerical

- Number stores numeric values. Python creates Number objects when a number is assigned to a variable. For example;

```
a = 3 , b = 5 #a and b are number objects
```

- Python supports 4 types of numeric data.
 - int (signed integers like 10, 2, 29, etc.)
 - long (long integers used for a higher range of values like 908090800L, -0x1929292L, etc.)
 - float (float is used to store floating point numbers like 1.9, 9.902, 15.2, etc.)
 - complex (complex numbers like 2.14j, 2.0 + 2.3j, etc.)
- Python allows us to use a lower-case L to be used with long integers. However, we must always use an upper-case L to avoid confusion.
- A complex number contains an ordered pair, i.e., $x + iy$ where x and y denote the real and imaginary parts respectively).

(ii) String

- The string can be defined as the sequence of characters represented in the quotation marks. In python, we can use single, double, or triple quotes to define a string.
- String handling in python is a straightforward task since there are various inbuilt functions and operators provided.

- Strings are declared using

```
' _ _ _ ' , " _ _ " , " " " _ _ _ " " "
```

```
e.g.; a = 'G'
```

```
print (a)
```

```
b = "Hello"
```

```
print (b)
```

```
C = " " " " Hello,
```

```
Everyone. How are
```

```
You today? " " "
```

```
print (C)
```

- Strings can be combined together using (+) plus operator (concatenation)

```
e.g.; a = 'Pune'
```

```
b = "Hello"
```

```
print (a + b)
```

- Strings can be sliced by using slicing operation to slice given data following syntax is used `var_name [start : end]`

- ☞ To slicing start and end are optional if required they can be used or otherwise will carry default values starting index as zero and end can be the complete length of data.

Example: a = 'Pune is beautiful city'

```
print(a[:])
print(a[:4])
print(a[2:])
print(a[3:8])
```

(iii) List

- ☞ It is a single variable which can store different types of elements.
- ☞ It is similar to array except that it can store different elements.
- ☞ Lists are declared using [] (square brackets) and elements are separated using , (comma)

Example: a = []

```
print(a)
data = [78, 4.5, 't', 'python', 123456789]
print(data)
```

- ☞ The plus and star operator is also used on list same as that of string.

Example: L1 = [1, 2, 3, 4]

```
L2 = [4, 5, 6]
print(L1 + L2)
print(L1 * 3)
```

- ☞ Slicing is also executed / performed on list data type.

Example: L1 = [1, 2, 3, 4, 5, 6]

```
print(L1[1:3])
```

- ☞ List is mutable

Example: L1 = [1, 2, 3, 4, 5]

```
print(L1)
L1[2] = 100
print(L1)
```

Result: [1, 2, 3, 4, 5]

[1, 2, 100, 4, 5]

(iv) Tuple

- ☞ Tuples are same as that of list except tuples are non-mutable.

- Tuples are also used to store different type of elements.
- It is an default data type used by python.
- Tuples are declared using rounded brackets ()

Example: L1 (1, 2, 3, 4, 5)

```
print (L1)
```

- As tuple is non-mutable the contents of tuple a read only, we cannot change or edit items of tuple.

Example:

```
L1 (1, 2, 3, 4, 5)
```

```
print (L1)
```

```
L1 [2] = 100
```

```
print (L1)
```

Concatenation, repetition and slicing is also applicable in case of tuple data type.

(v) Dictionary

- It is an ordered set of key and value.
- It is like an hash table or associative array which stores keys with specific element.
- Here key and value can be of any type i.e. integer, float, character or string.
- Dictionary is defined using curly braces { } and pairs are separated by comma (,) key, value pair is formed using colon (:)

Example:

```
data={1 : 'one', 'two' : 2, 4.5 : 'decimal', 'name' : 'Pune'}
```

```
print (data)
```

- Dictionary data type is mutable.

Example:

```
data = {1 : 'one', 'two' : 2, 4.5 : 'decimal', 'name' : 'Pune'}
```

```
data ['two'] = 8
```

```
print (data)
```

- Concatenation, repetition and slicing are also applicable in case of dictionary data type.

Data type conversion

- Depending on how data is assigned or stored by interpreter following conversions are used.

(a) Implicit

In this method data type is decided by interpreter depending on the value assigned.

Example: `b = 10`

In above e.g. given data is a rounded number i.e. 10 so interpreter will store it as an integer in the name of b.

(b) Explicit

In this method data is converted externally then assigned to the variable.

`a = float(5)`

In above e.g. data is converted externally i.e. 5 into float number then stored into variable a.

Exercise - 1.3

Question Based on 3rd Milestone

Click or scan using
FORTFLAG app
for detailed
solution.



- Q 1. What is comment? Explain different types of comments.
- Q 2. What is operator? List out different type of operators available in python.
- Q 3. What is slicing? What will be output of following instructions.
- `Str="Programming with python is fun"`
- (a). `print(str [: 6])`
 - (b). `print(str [2 : 10])`
 - (c). `print(str [1 :])`
 - (d). `print(str [-3 : 10])`
- Q 4. What is literal? Describe different types of literal with suitable example.
- Q 5. Explain following terms with suitable example.
- (a). Comment
 - (b). Reserve words
 - (c). Indentation
 - (d). Literals
- Q 6. Describe following data types with example.
- (a). Tuple
 - (b). Dictionary
- Q 7. Describe Identity operator along with example.
- Q 8. Describe Membership operators along with example.

- Q 9. Write short note on Data types in python.
- Q 10. Write python program to calculate roots of quadratic equation.
- Q 11. Write python program to calculate compound interest i.e.

$$A = P \left(1 + \frac{r}{n} \right)^{nt}$$

A = final amount

P = initial principal balance

r = interest rate

n = number of times interest applied per time period

t = number of time periods elapsed

- Q 12. What is algorithm? Write algorithm to convert no of seconds into hours.
- Q 13. Write a python program to convert no of seconds into hours.
- Q 14. Write a python to program to calculate amount of money in piggybank which is in the form of coins of Rs 10, Rs 5, Rs 2, Rs 1.
- Q 15. Write a algorithm and draw flowchart to calculate average of five numbers.
- Q 16. Write python program to swap values of two variables.
- Q 17. Write python program to accept first name and last name as input from user and display the message as Hello firstname lastname, How are you today.