JOIN

PUNE ENGINEERS
WHATSAPP CHANNEL

All Subject Notes:

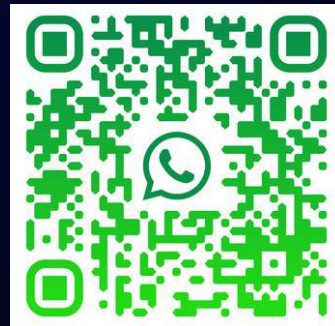**https://www.studymedia.in/fe/notes** ↗

JOIN COMMUNITY OF 30K+ ENGINEERS

CLICK HERE TO JOIN

SCAN ME

<div align="center">

Programming and Problem Solving

Unit- 1 Notes

Unit I: Problem Solving, Programming and Python Programming

</div>

1.1 General Problem Solving Concepts

1.1.1 Problem Solving in Everyday Life:

Q.1. Define Problem. What are steps in problem solving?

Definition of problem

Problem is defined as a situation or issue or condition which needs to be solve to achieve a specific goal.

Steps in problem solving

1. **Identify the problem** - You need to be clear about what exactly the problem is. If the problem is not identified then it cannot be solved. example- when you want to learn a language then first identify which language do you want to learn

2. **Understand the problem** - Here problem should be defined properly so that it helps in understanding the problem more. It helps to understand knowledge base of the problem. It helps in finding a target for what you want to achieve.

3. **Identify alternative ways to solve the problem** - This step is very crucial because until and unless we explore all options, finding a correct solution is next to possible. This involves the process to create a list of solutions.

4. **Select the best way to solve the problem** - Identify and evaluate pros and cons of each possible solution and then select the best solution. Selecting proper way from alternatives require a lot of study and need to study consequences as well.

5. **List instructions that enable you to solve the problem using selected solution** - Here we need to list out the steps which should

1

be followed so that anyone can solve the problem. This should be in

detail and precise at the same time.

6. **Evaluate the solution -** It mean to test the solution and check if the

result is satisfactory 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.

1.1.2 Types of problems:

Q. 2. Explain types of problems in detail?

Ans:

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.

2

(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.

1.1.3 Problem Solving with computers:

Q. 3. Write a note on Problem solving with computers.

Ans: Computer is a multi-purpose electronic machine which is used for storing, organizing and processing the data.

Computer is not able to find the solution by itself; rather the computer is a machine which follows the instruction given by programmer to solve any problem.

So all the efforts required to understand the problem and define procedure to solve that problem are taken by programmer (a human being).

Problem Definition:

Before a program is written for solving a problem, it is important to define the problem clearly.

Define problem: Problem is defined as a situation or issue or condition which needs to solve to achieve the goal.

For most of the software projects, the system analyst approach system users to collect user requirements and define the problem that the system aims to solve.

System analyst typically looks at following issues:

o What input is required for achieving expected output?

o Expected output of the problem.

o Current method of solving the problem.

o Can the problem or part of the problem be more effectively solved by a software solution?

o Computers are built to deal with algorithmic solutions, which are often difficult or very time consuming for humans.

3

Solution:

o   Solution means the instructions listed during problem solving – the instructions that must be followed to produce the best results.

o   The result may be: More efficient, faster, more understandable or reusable.

Results:

o   The result is outcome or the completed computer assisted answer.

o   May take any form: Printout, updated files, output to monitor speakers, etc.

Program:

o   Program is the set of instructions that make up the solution after they have been coded into a particular computer language.

1.1.4   Difficulties with problem solving:

Q.4. Explain the difficulties with Problem solving?

Ans:

o   Some people may not have got training regarding how to solve problems.

o   Some may not able to make a decision for fear it will not be the correct one.

o   There may be inaccuracy in defining the problem correctly or may not generate the sufficient list of alternatives.

o   While selecting best alternative, they may skip better alternative due to urgency.

o   The problem solving process is difficult. It takes practice as well as time to perfect, but in the long run the process proves to be of great benefit.

o   Illogical sequencing of instructions: In the process of problem solving on the computer, one of the most complicated jobs for the problem solver is writing the instructions. Example: Consider a task to find which number is the maximum from a set of three numbers. Near about everyone is immediately able to tell which is the largest but several of them are unable to explain the steps they followed to get the result. Computer is nothing but a machine that will perform only task that the user can explain.

4

1.1.5 **Problem Solving Aspects:**

Q.6. What is modularization? Explain different approaches to design an algorithm.

Ans:

Modularization:

For a complex problem, its algorithm is often divided into smaller units called modules. This process of dividing an algorithm into modules is called as modularization.

Top down design approach:

- A top-down design approach starts by dividing the complex algorithm into one or more functions.
- These functions can further be decomposed into one or more sub-functions, and this process of decomposition is iterated until the desired level of module complexity is achieved.
- Top-down design method is a form of stepwise refinement where we begin with the topmost module and incrementally add functions that it calls.
- Therefore, in a top-down approach, we start from an abstract design and then at each step, this design is refined into more concrete levels until a level is reached that requires no further refinement.

Diagram: Top Down Approach
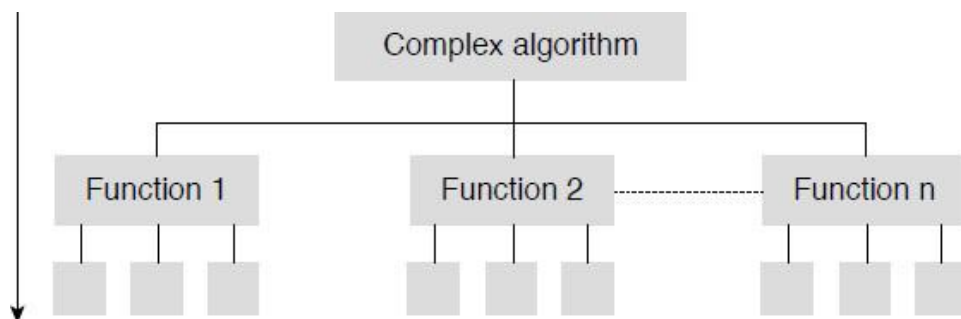
Bottom up design approach:

- It is just reverse of top down approach.
- In bottom up design, we start with designing the most basic or concrete modules and then proceed towards designing higher level modules.
- The higher level modules are implemented by using the operations performed by lower level modules.
- Thus, in this approach sub-modules are grouped together to form higher level module.
- All the higher level modules are clubbed together to form even higher level modules.
- This process is repeated until the design of the complete algorithm is obtained.

Difference between Top down and Bottom up approach:

Top-Down Approach	Bottom-Up Approach
Divides a problem into smaller units and then solve it.	Starts from solving small modules and adding them up together.
This approach contains redundant information.	Redundancy can easily be eliminated.
A well-established communication is not required.	Communication among steps is mandatory.
The individual modules are thoroughly analysed.	Works on the concept of data-hiding and encapsulation.
Structured programming languages such as C use top-down approach.	OOP languages like C++ and Java, etc. uses bottom-up mechanism.
Relation among modules is not always required.	The modules must be related for better communication and work flow.
Primarily used in code implementation, test case generation, debugging and module documentation.	Finds use primarily in testing.

1.1.6 Problem Solving Strategies:

Q.5. Explain the problem solving Strategies?

Ans.

Computer is a very powerful and versatile machine and can do any task given to it provided that the programmer has already fed correct instructions to direct what, how and when the steps have to be done to solve a particular problem. For this he should work to develop a step by step solution to problem. These steps can be given as:

- Clearly define the problem in very simple and precise language.
- Analyse the problem to find out different ways to solve problem and decide the best possible solution.
- Define the selected solution in a detailed step by step manner.
- Write the steps in a particular programming language so that it can be executed by the computer.

The entire software development process is divided into a number of phases where each phase performs a well defined task. Moreover, the outputof one phaseprovides the input for subsequent phase. The phases in the software development life cycle (SDLC) process are summarized as follows:
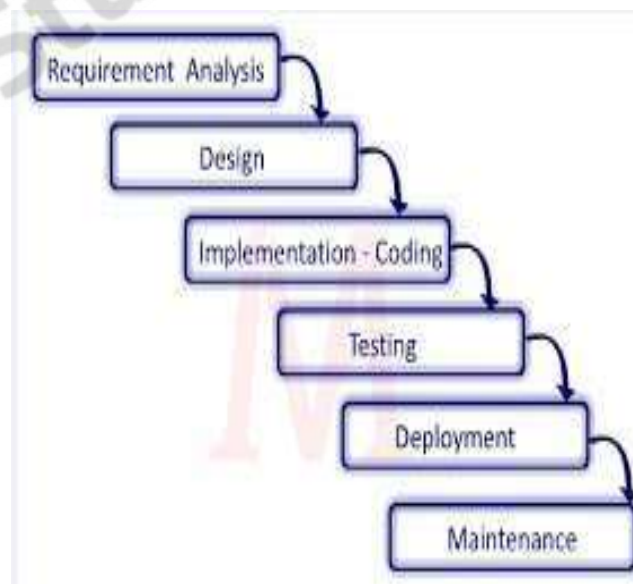

Fig: Software Development Life cycle (SDLC)

- Requirement Analysis: In this phase, the user's expectations are gathered to know why the software has to be built. The gathered requirements are analyzed to decide scope of software. The last activity in this phase includes documenting every identified requirement of the users in order to avoid any doubts regarding functionality of software. The functionality, capability, performance and availability of hardware and software components are all analyzed in this phase.

- Design: The requirements documented in previous phase act as an input to the design phase. In the design phase, core structure of the software is broken down into modules. The solution of the program is specified for each module in the form of algorithms or flowcharts.

- Implementation: In this phase, the designed algorithms are converted into program code using any of the high level languages. The particular choice of language will depend on the type of program, such as whether it is system or application program. This phase is also called as construction phase as the code of the software is generated in this phase. While constructing the code, the development team checks whether the software is compatible with available hardware that are mentioned in requirement specification document.

- Testing: In this phase all modules are tested together to ensure that the overall system works well as a whole product. In this phase the software is tested using a large number of varied inputs, also known as test data, to ensure the software is working as expected by user's requirements.

- Software development, training and support: After the code is tested and the software or the program has been approved by the users, it is installed or deployed in the production environment. In this phase it becomes very crucial to have training classes for users of software.

- Maintenance: Maintenance and enhancement are ongoing activities that are done to cope with newly discovered problems or new requirements. Such activities may take a long time to complete as the requirement may call for the addition of new code that does not fit original design or an extra piece of code, required to fix an unforeseen problem.

1.2 Program Design Tools

1. Algorithm
2. Flowchart
3. Pseudocode

Q.7. Discuss algorithm with example. Also enlist characteristics of good algorithm.

Ans:

Definition of algorithm

- An algorithm is precise, step-by-step set of instructions for solving a computational problem.
- It is a set of ordered instructions which are written in simple English language.
- In this type of problem solving the required input and expected outputs are identified and according to that the processing will be done.
- Algorithms helps the programmer to write actual logic of a problem on paper and validate it with the help of paper and pencil, and correct it if any fault is noticed.

Characteristics of algorithms:

Precision: The instructions should be written in a precise manner.

Uniqueness: The outputs of each step should be unambiguous, i.e., they should be unique and only depend on the input and the output of the preceding steps.

Finiteness: Not even a single instruction must be repeated infinitely.

9

Effectiveness: The algorithm should be designed in such a way that it should be the most effective among many different ways to solve a problem.

Input: The algorithm must receive an input. Algorithm may accept zero or more inputs.

Output: After the algorithm gets terminated, the desired result must be obtained.

Generality: The algorithm can be applied to various set of inputs.

Example: Algorithm to calculate area of circle.

    Algorithm: Area of circle
    Input: R=Radius
    Output:
    A=Area
    Steps:
    Step 1:BEGIN
    Step 2: Accept the R
    Step3: Find the square of R and store it in
    A=R*R Step4: Multiply R with 3.14 and
    store the result in A Step5 :Display the value
    of A
    Step6: END

Q.8. Explain control structures used in algorithms.
Ans:

Q. Explain control structures used in algorithm?

Algorithm has finite number of steps and some steps may involve in decision making and repetition. Broadly algorithm may apply three decision making structures

1. **Sequence:**
   - ✓ In sequence control structure each step of the algorithm is executed in the specific order.
   - ✓ Algorithm performs the steps in a purely sequential order.
   - ✓ Example:

            Step 1: Start
            Step 2: Input first number as A
            Step 3: Input Second number as B
            Step 4: Perform sum=A+B
            Step 5: Display sum
            Step 6: Stop

2. Decision:

- ✓ Decision control structures can be used for making decisions and branching of statements, where the outcome of the process depends on some condition.
- ✓ A condition in this context is any statement that may evaluate either to a TRUE or FALSE value.
- ✓ This form is commonly known as the if—else construct.
- ✓ Example:

> Step 1: Start
>
> Step 2: Input first number as A
>
> Step 3: Input Second number as B
>
> Step 4: IF A==B
>
> > Print Equal
> >
> > ELSE
> >
> > Print Not Equal
>
> Step 5: Stop

3. Repetition:

- ✓ These control structures are used for executing one or more steps for a number of times.
- ✓ It can be implemented using constructs such as for loop, while loop etc.
- ✓ Example:

> Step 1: Start
>
> Step 2: Initialize I=1, N=10
>
> Step 3: Repeat Steps 3 and 4 while I<=N
>
> Step 4: Print I
>
> Step 5: Set I=I+1
>
> Step: Stop

Q.9. Write an algorithm to swap two numbers.

Ans:

Algorithm: Swap two numbers

Input: Two numbers Output: Swapped numbers

Step 1 : Start

Step 2 : READ num1, num2

Step 3 : temp = num1

Step 4 : num1 = num2

Step 5 : num2 = temp

Step 6 : PRINT num1, num2

Step 7 : Stop

Q.10. **Write an algorithm to find largest number amongst three numbers.**

Ans: **Algorithm**

Algorithm: Greatest Number

Input: Three Numbers

Output: Greatest among three Numbers

Steps:

Step 1:BEGIN

Step2: Read values for NUM 1, NUM2 and NUM3

Step3: if NUM1>NUM2

          Check if Num1>NUM3,if true then display NUM 1 is Largest number

  Else check if NUM2>NUM1

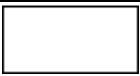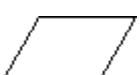   Check if Num2>NUM3 display NUM 2 is Largest number
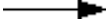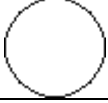
  Else check if NUM3>NUM1

   Check if Num3>NUM2 display NUM 3 is Largest number

Step4: END

Q.11. **Define term Flowchart. Explain Symbols used in flowchart.**

- A Flowchart is a graphical representation of a Algorithm.

- The sequence of steps in algorithm is maintained and represented in the flowchart by using some standard symbols such as rectangles and directed lines.

- As flowchart uses the pictorial representation of an algorithm, it becomes easier to understand what is going on.

- One of the most common ways to express algorithm with writing a program is a flowchart.

- Symbols used in flowchart:

Sr. No	Symbol	Name of Symbol	Use of Symbol
1		Start/End Terminal	Indicate start and end of the algorithm. Represented as rounded rectangle.
2		Action or Process	Indicate set of operations like assignments, Initialization calculations etc.
3		Decision	Indicates the decision making statements. It indicates a condition on which decision are made.
4		Input/Output	Indicate Input provided to algorithm and outputs processed by algorithm. It is represented by parallelogram.

12

5	———▶	Flow line	Indicates the flow of algorithm. Represented as arrow of line.
6	◯	On page Connector	To connect parts of algorithm. This symbol is used if the parts are on same page
7	▷	Off Page Connector	To connect parts of algorithm. This symbol is used if the parts are on different page

Q.12. Draw a flowchart to find greater number amongst two num0bers.
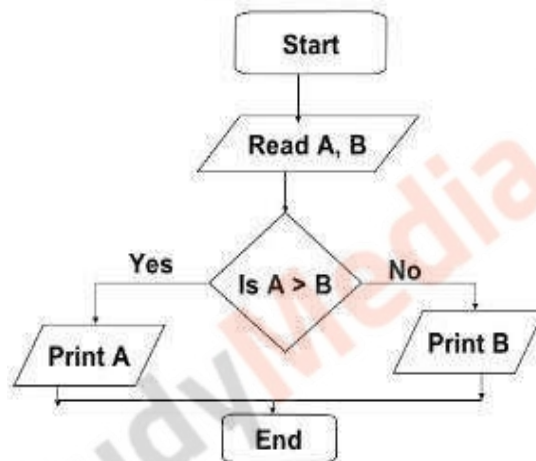
Q.13. Draw a flowchart to check whether a given number is even or odd.

13
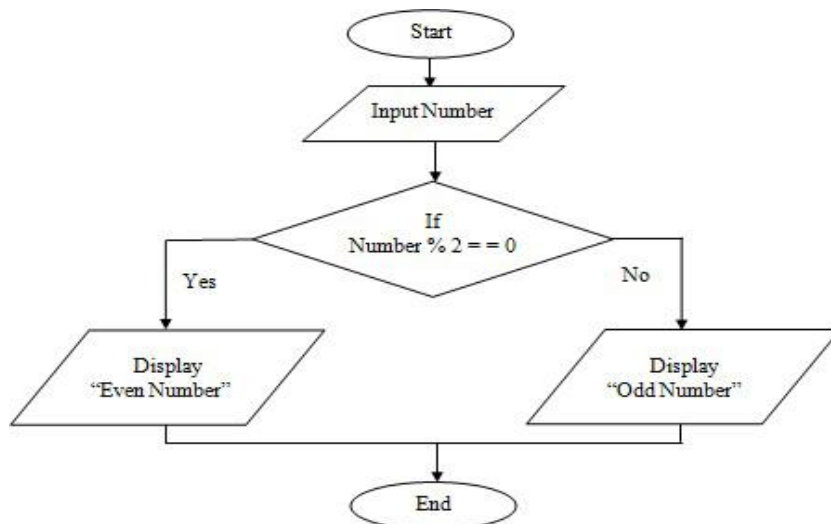
Q. 14. Explain various program design tools.

Ans:

Algorithm:

Definition: An algorithm is precise, step-by-step set of instructions for solving a computational problem.

Example:

Algorithm : To calculate area of circle
Input: R=Radius
Output: A=Area
Steps:
Step 1: Begin
Step 2: Accept R
Step3: Compute area= 3.14*R*R
Step4 :Display the value of area
Step5: End

Flowchart:

Definition: Flowchart is graphical representation of computation and provides visual description of solution/algorithm step by step.

Example:

Flowchart for calculating area of circle
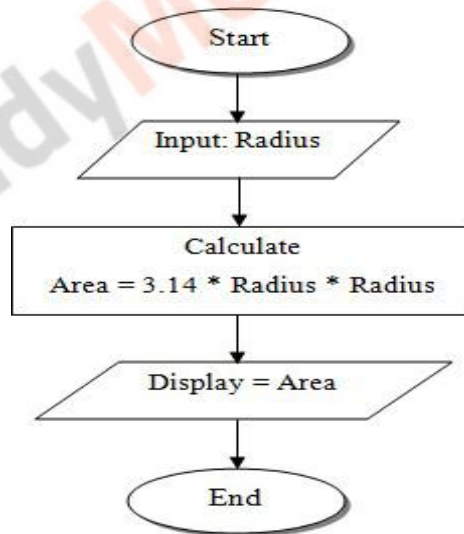
Pseudocode:

Definition: It is compact informal representation of algorithm. It is written in natural language with description of the details and compact mathematical notations.

Example:

Student is Pass or Fail
If student's grade is greater than or equal
to 60 Print "Passes"

14

JOIN BIGGEST SPPU ENGINEERING GROUP

1

PuneEngineers Channel

https://whatsapp.com/channel/0029Va7akvF8V0theLGDwl21

2

PuneEngineers Channel

https://t.me/PuneEngineers

3

SPPU Engineering Group

https://t.me/SPPUBTECH

OVER 30,000+

PUNE ENGINEERS

SPPU ENGINEERING @SPPUBTECH

JOIN NOW

Else
Print "Failed"

1.3 Features of Python

Q.15. Explain various features of python.
Ans.

1. **Simple:** Python is simple and small language. Reading a program written in python feels like reading English.

2. **Easy to Learn**: Python programming is clearly defined and easily readable.

3. **Versatile:** Python supports wide variety range of applications ranging from text processing to Web applications

4. **Free and Open Source**: Python software is freely available on internet

5. **High Level Language**: Programmers don't have to worry about low level details like managing memory used by program etc.

6. **Interactive**: Programs in python works in interactive mode

7. **Portable (Platform Independent):** Python is portable language. Program can work similarly on different platform like window, Linux, Macintosh, Solaris etc.

8. **Object Oriented:** Python supports Object oriented style of programming like C++, Java languages. It support features like C class, Objects, Inheritance etc.

9. **Interpreted:** Python is processed at run time by the interpreter. So no need to compile program before executing it.

10. **Scalability:** Python support modular programming, so programmer can add module at any stage of the program

11. **Secure:** Python programming environment is secure from tempering

1.4 History and Future of Python

- Python is a general purpose interpreted interactive object oriented and high level programming language.

- It was first introduced in 1991 by Guido van Rossum, a Dutch computer programmer.

- The language places strong emphasis on code reliability and simplicity so that the

programmers can develop applicationsrapidly

- Python is multi-paradigm programming language, which allows user to code in several different programming styles.

- Python supports cross platform development and is available through open source. Python is widely used for scripting in Game menu applications effectively

History of Python:

- Python is created by **Guido Van Rossum in the 1980s.**

- Rossum published the first version of Python code (0.9.0) in February **1991 at the CWI (Centrum Wiskunde & Informatics) in the Netherlands , Amsterdam.**

- Python is derived from **ABC programming language, which is a general-purpose programming language that had been developed at the CWI.**

- Rossum chose the name "**Python", since he was a big fan of Monty Python's Flying Circus.**

- Python is now maintained by a core development team at the institute, although Rossum still holds a vital role in directing its progress.

Future of Python:

- Python's userbase is vast and growing – it's not going away any time soon.

- Utilized by the likes of Nokia, Google, and even NASA for it's easy syntax, it looks to have a bright future ahead of it supported by a huge community of OS developers.

- Its support of multiple programming paradigms, including object-oriented Python programming, functional Python programming, and parallel programming models makes it a highly adaptive choice – and its uptake keeps growing.

1.5 Writing and Executing Python Programs

- Open IDLE.

- Go to **File** > **New**.

- Write Python code in the file.

- Then save the file with .py extension. For example, hello.py, example.py etc.
  You can give any name to the file. However, the file name should end with **.py**

- Run the program.

1.6 Literal Constants

The data that is provided in the variable are known as literals in Python. Python supports the following literals:-

Types of literals with example

Python support the following literals:

4.1)   Numbers:

Numbers as a name suggest, refers to a numeric value.

You can use three types of numbers in a python program. These include integers floating point, and complex numbers.

- int- Number like 5 or other whole numbers are represented as integers.

- float- numbers like 3.23, 8.73 are termed as floating point numbers.

- Complex- numbers like a+bj form (like 3+2j) are complex numbers

Remember that commas are never used in numeric literals or numeric values.

Although there is no limit to the size of integers that can be represented in python, floating point numbers do have a limited range and limited precision. In python you can have a floating point numbers in a range 10-308 to 10308 with 16 to 17 digits of precision. In fact, large floating point numbers are efficiently represented in scientific notation. For ex, 5.0012304*106 (6 digits of precision) can be written as 5.0012304e+6

Built-in format( ) function

Any floating point value may contain an arbitrary number of decimal places, so it is always recommended to use the built-in format( ) function to produce a string version of a number with a specific number of decimal places. Observe the difference between the following outputs.

17

# Without using format( )	# using format( )
>>> float(16/(float(3)))	>>> format(float(16/(float(3))),
5.333333333333333	'.2f')
	'5.33'

Here .2f in the format() function round the result to two decimal places of accuracy in the string produced.

For very large (or very small) values, 'e' can be used as a format specifier.

The format() function can also be used to format floating point numbers in scientific notation. Look at the result of the expression given below:

```
>>> format(3**50, '.5e')
'7.17898e+23'
```

The result is formatted in scientific notation with five decimal places

```
>>> format(123456,',')
'123,456'
```

Simple operations on numbers

Python can carry out simple operations on numbers. To perform calculation, simply enter the numbers and the type of operations that need to be performed on them directly into the python console, and it will print the answer, as shown in the following examples.

>>> 10+7	>>> 5 * 3.0	>>> 12*10	>>> 152.78 // 3.0
17	15.0	120	50.0

4.2)   <u>String Literals:</u>

A string is a group of characters. If you want to use text in python you have to use string.

Single quotes, double quotes or triple quotes are used to define String literals.

18

There are two kinds of strings supported in Python, single line and multiline string literals.

Example:

```
str1 = 'Amit'
str2 = "Ricky Ponting"
str3 = '''Welcome to python '''
```

String Literal Concatenation:

Python concatenates two string literals that are placed side by side.

```
>>> print('Beautiful Weather' '....' 'Seems it would rain')
Beautiful Weather....Seems it would rain
```

Escape Sequences:

Some characters **(like " ' \ )** cannot be directly included in string. Such characters must be escaped by placing a backslash before them.

For example,

```
>>> print( 'What's your name?' )
SyntaxError: invalid syntax
```

We got this error, as python got confused where the string starts and ends. So, we need to clearly specify that this single quote does not indicate the end of the string . This indication can be given with the help of escape sequence.

```
For example,
>>> print('What\'s your name?')
What's your name?
```

You can use the escape sequence for the newline character (\n). Character following the \n are moved to the next line.

```
>>> print("Today is 15th August. \n India became independent on this day.")
Today is 15th August.
 India became independent on this day.
```

Another useful escape sequence is \t which inserts tab in a string.

```
>>> print("Hello..\t Welcome to python")
Hello..     Welcome to python
```

Note that when specifying a string, if a single backslash (\ ) at the end of the line is added , then it indicates that the string is continued in the next line, but no new line is added otherwise.

For example,

```
>>> print("I have studied many programming languages. \
But my favorite language is python")
I have studied many programming languages. But my favorite language is python
```

The different types of escape sequences used in Python

Escape sequence	Purpose	Example	Output
\\	Print Backslash	print("\\")	\
\'	Print single-quote	print(" \' ")	'
\"	Print double-quote	print(" \" ")	"
\a	Rings Bell	print("\a")	Bell rings
\n	Print a newline	print("Hello\n World")	Hello World
\t	Print tab	print("Hello \t World")	Hello World

20

1.7 Variables and Identifiers

<u>Variables:</u>

- ✓ Variable are just part of computer memory where information is stored.
- ✓ To identify these memory easily, each variable is given an appropriate name.
- ✓ Every variable is assigned a name which can be used to refer to the value later in the program.
- ✓ Further it can be defined as 'Variables are reserved memory locations that stores values'.

- ✓ Rules for an Variable
    - o An Variable can only have alphanumeric characters (a-z , A-Z , 0-9) and underscore(_).
    - o The first character of an identifier can only contain alphabet (a-z , A-Z) or underscore (_).
    - o Keywords are not allowed to be used as Variable.
    - o No special characters, such as semicolon, period, whitespaces, slash or comma are permitted to be used in or as Identifier.
    - o Variable are also case sensitive. For example name and Name are two different Variable.

```
counter = 100          # An integer assignment

miles   = 1000.0       # A floating point

name    = "John"       # A string


print counter

print miles

print name
```

<u>Identifiers:</u>

Q. What is identifier? Explain rules of identifiers.
Identifiers
Identifier is a collection of alphanumeric character used to give name to the programming elements like variables, arrays, functions, etc.

Rules for constructing identifiers.

- Each identifier starts with an alphabet or underscore and then followed by any number of alphabets, digits or underscore.

- Identifier should not start with digit

- Identifier is the unique name given to the element of a program.

- Identifiers are case sensitive, so the max and MAX considered as two different identifiers.

- An Identifier should not contain special symbols, comma, or blank space.

- Identifiers are user-defined words so they should not be same as of the keywords otherwise compiler will generate an error.

1.8 Data Types

Q. Explain different data types supported by python?

Ans:

1. **Numbers**
   - They are defined as int, float and complex class in Python.
     - int: It consists of all integer numbers. Integers can be either positive or negative. Buil in size of integers is unlimited. Ex: -5, 0, 890 etc
     - float: Float point numbers are written as decimal numbers separated by decimal point. It is expressed in the form of integer number and fractional part sepaprated by decimal point. Ex: -5.2, 11.325 etc.
     - complex: Complex numbers are expressed in the form of a+bi, where a is real part and b is imaginary part. Ex: 3+2i etc

2. **String**
   - String is sequence of Unicode characters.
   - We can use single quotes or double quotes to represent strings.
   - Multi-line strings can be denoted using triple quotes, ''' or """.

     >>>s = "This is a string"
     >>>s = '''a multiline

3. **Python List**
   - List is an ordered sequence of items.
   - List represents collection of data elements (items).
   - These data items can be of different data types.
   - Items in list are separated by commas are enclosed within brackets [ ].
   - Ex: a = [1, 2.2, 'python']

4. Python Tuple

- <u>Tuple</u> is an ordered sequence of items same as list.The only difference is that tuples are immutable. Tuples once created cannot be modified.
- It is defined within parentheses () where items are separated by commas.
- Ex: t = (5,'program', 1+3j)

5. Dictionary

- Dictionary is an unordered collection of items in key:value pair of any type. In dictionary values are separated by comma inside braces { }. ·
- In dictionary, key should be unique.
- Ex: a = {'name':'Bob','Age': 21}

1.9 Input and Output operation

input() and print() functions are widely used for standard input and output operations respectively.

1. input()

To allow flexibility we might want to take the input from the user. In Python, we have the input() function to allow this.

Syntax:

```
input (expression)
```

2. print()

We use the print() function to output data to the standard output device (screen). **print()** evaluates the expression before printing it on the monitor. Print statement outputs an entire (complete) line and then goes to next line for subsequent output (s). To print more than one item on a single line, comma (,) may be used.

Syntax:

```
print (expression/constant/variable)
```

23

Example:

x = input("Enter any value: ")

printing value

print("Entered value is: ", x)

1.10 Comments

Q. Explain comments in python.

Ans:
- · The comments are ignored by the interpreter and are used to add additional information about the program or its statements.
- · They can be written anywhere in the program.
- · Two way comments can be given in Pyhton
  1. One line comments using (#)
  2. Multi line comments using (''' ''')
- · Comments in Python begins with ' #' character.
- · A comment may appear at the beginning of a line or can be placed after whitespace or code, but not within a string literal.

        Example:1
                #This is first Python Program
                 print("Hello World") # prints Hello World

        Example: 2
                ''' program is written for addition
                    Of two numberd '''
                    x=10
                    y=20;
                        Print(x+y)

1.11 Reserved Words

Q. Explain keywords (reserved words) in python.

- • In python there are certain words which have predefined meaning. They are called as reserved words.
- • Reserved words are also called as keywords.
- • Reserved words cannot be used for naming an identifier.
- • Example:
      for, int, while etc.

and	del	for	is	raise

assert	elif	from	lambda	return
break	else	global	not	try
class	except	if	or	while
continue	exec	import	pass	with
def	finally	in	print	yield

1.12 Indentation

Q. Write a note on indentation.
Ans.

- White space at the beginning of the line is called indentation.
- These white spaces or the indentation are very important in python.
- In python program, the leading white space and tabs at the beginning of the logical line determines the indentation level of logical line.
- All statements inside block should be at same indentation level.
- Python checks the indentation level very strictly and gives an error if indentation is not correct.

<u>Use of indentation:</u>

- In most of the programming languages, indentation has no effect on program logic. It is used to align statements to make the code readable.
- However, in python indentation is used to associate and group statements.

<u>Example:</u>

Program to find number is even or odd

```python
num = int(input("Enter a number: "))
if (num % 2) == 0:
    print("Number is even")
else:
    print("Number is Odd")
```

1.12 Operators and Expressions

Q. What is operator? Enlist and explain types of operators in python.
Ans.

Python operators are symbols that are used to perform mathematical or logical manipulations. Operands are the values or variables with which the operator is applied to, and values of operands can manipulate by using the operators.

25

Types of Operators with explanation

Python language supports the following types of operators.

- **Arithmetic Operators**
  +, -, *, /, %, /,**

  <u>Example:</u>
  x = 15
  y = 4
  print('x + y =',x+y)
  print('x - y =',x-y)
  print('x * y =',x*y)
  print('x / y =',x/y)
  print('x // y =',x//y)
  print('x ** y =',x**y)
  <u>Output:</u>
  x + y = 19
  x - y = 11
  x * y = 60
  x / y = 3.75
  x // y = 3
  x ** y = 50625

- **Comparison (Relational) Operators**
  <, >, <= ,>=, !=, <, >
  When two operands are compared evaluated as either true of false. If True the
  binary value is 1 , if false the binary value is 0.
  True : 1
  False : 0

  <u>Example:</u>
  x = 10
  y = 12
  print('x > y is',x>y)
  print('x < y is',x<y)
  print('x == y is',x==y)
  print('x != y is',x!=y)
  print('x >= y is',x>=y)
  print('x <= y is',x<=y)
  <u>Output:</u>
  x > y is False
  x < y is True
  x == y is False
  x != y is True
  x >= y is False
  x <= y is True

26

- **Assignment Operators**
  = ,+=, -=, *=, /= , //=, %=
  <u>Example:</u>

Operator	Example	Equivatent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5

- **Logical Operators**
  and , or, not
  non zero numbers are evaluated as true while zero as false.
  Operates number as whole no binary conversion is required.
  <u>Example:</u>
  x = True
  y = False
  print('x and y is',x and y)
  print('x or y is',x or y)
  print('not x is',not x)
  <u>Output:</u>
  x and y is False
  x or y is True
  not x is False

- **Bitwise operators**
  &, |, ^, ~, >>, <<
  Bitwise operators process individual bits of data.
  Numbers are converted into binary form and then processed by bitwise operators.

  <u>Example:</u>

&	Bitwise AND	x& y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)

27

^	Bitwise XOR	x ^ y = 14 `(0000 1110)`
>>	Bitwise right shift	x>> 2 = 2 `(0000 0010)`
<<	Bitwise left shift	x<< 2 = 40 `(0010 1000)`

- **Identity Operators**
  is, is not
  These operators checks 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:
  x1 = 5
  y1 = 5
  x2 = 'Hello'
  y2 = 'Hello'
  x3 = [1,2,3]
  y3 = [1,2,3]
  print(x1 is not y1)
  print(x2 is y2)
  print(x3 is y3)
  Output:
  False
  True
  False

- **Membership operators**
  in, not in
  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:
  x = 'Hello world'
  y = {1:'a',2:'b'}
  print('H' in x)
  print('hello' not in x)
  print(1 in y)
  print('a' in y)

28

Output:
True
True
True
False

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisions, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

1.13 Expressions in Python

- ✓ An expression is any valid combination of symbols like variables, constants and operators that represents a value.
- ✓ In python, an expression must have at least one operand and can have one or more operators.

- ✓ An Example of valid expression:
    - o A+B*C-5
    - o X=A/B
    - o Y=A^B

- ✓ An example of invalid expression:
    - o A+
    - o A*

- ✓ Python supports different types of expressions can be classified as:
    1. Based on the position of operators in an expression:
        a. Infix expression:
            - ✓ Example: a=**b-c**
        b. Prefix expression:
            - ✓ Example: a=**+bc**
        c. Postfix expression:
            - ✓ Example: a=**bc+**

    2. Based on the data type of the result obtained on evaluating an expression:
        a. Constant expression:
            - ✓ Example: 8+2-3
        b. Integral expression:
            - ✓ Example: a=10
        c. Relational expression:
            - ✓ Example: c=a>b
        d. Logical expression:
            - ✓ Example: a>b and a>c

1.14 Application of Object –Oriented Programming (OOP)

Object-Oriented Programming is a programming paradigm that uses objects and classes to structure code. Python fully supports OOP, and its key features include:

1. *Classes and Objects*
   A **class** is a blueprint for creating objects, while an **object** is an instance of a class.

   Example: A Dog class can have attributes like name and methods like bark().

2. Encapsulation

   Bundling data (attributes) and methods (functions) that operate on the data into a single unit (class).

   Protects data from outside interference by restricting access (using private or protected members).

3. *Inheritance*
   Allows a class (child) to inherit attributes and methods from another class (parent).

   Promotes code reusability and hierarchical classification.

   Example: A Cat class can inherit from an Animal class.

4. *Polymorphism*
   Allows objects of different classes to be treated as objects of a common superclass.

   Enables methods to behave differently based on the object calling them.

   Example: A draw() method can behave differently for Circle and Square objects.

5. *Abstraction*
   Hides complex implementation details and exposes only essential features.

   Example: Using abstract classes or interfaces to define a template for subclasses.

1.15 Features of Python

Python is a versatile language used in a wide range of applications, including:

1. *Web Development*
   - Frameworks like **Django** and **Flask** are used to build scalable and secure web applications.

2. *Data Science and Analytics*
   - Libraries like **NumPy**, **Pandas**, and **Matplotlib** are used for data manipulation, analysis, and visualization.

3. *Machine Learning and AI*
   - Libraries like **TensorFlow**, **PyTorch**, and **Scikit-learn** are used for building machine learning models and AI applications.

4. *Scientific Computing*
   - Python is used in scientific research for simulations, modeling, and data analysis.

5. *Automation and Scripting*
   - Python is widely used for automating repetitive tasks, such as file handling, web scraping, and system administration.

6. *Game Development*
   - Libraries like **Pygame** are used to develop 2D games.

7. *Desktop Applications*
   - Frameworks like **Tkinter**, **PyQt**, and **Kivy** are used to create cross-platform desktop applications.

8. *Internet of Things (IoT)*
   - Python is used to program IoT devices and manage data from sensors.

9. *Cybersecurity*
   - Python is used for penetration testing, network scanning, and developing security tools.

10. *Finance and Trading*
    - Python is used for algorithmic trading, risk management, and financial analysis.

11. *Education*
    - Python is often the first programming language taught due to its simplicity and readability.

JOIN BIGGEST SPPU ENGINEERING GROUP

1

PuneEngineers Channel

https://whatsapp.com/channel/0029Va7akvF8V0theLGDwI21

2

PuneEngineers Channel

https://t.me/PuneEngineers

3

SPPU Engineering Group

https://t.me/SPPUBTECH

OVER 30,000+

PUNE ENGINEERS

SPPU ENGINEERING @SPPUBTECH

JOIN NOW