

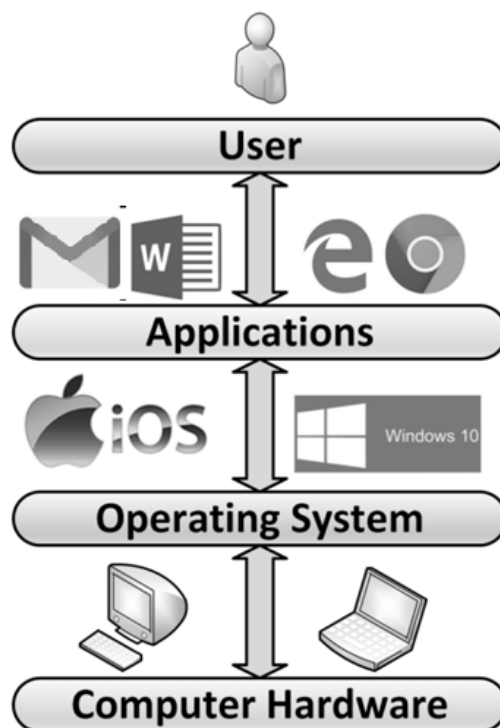
B.Sc.(Data Science) I sem
Problem Solving and Python Programming
Unit-I

1. Explain about Computer System.

A Computer is electronic devices that stores, manipulates and retrieves the data. We can also refer computer computes the information supplied to it and generates data.

A Computer System has following components

- ❖ **User (A person who uses the computer)** : A computer user, or end user, is any person who interacts with a computer system, software, or network service to perform tasks
- ❖ **Hardware:** Hardware are the physical components of the computer system. The hardware components consist of several parts like input devices, Central Processing Unit (CPU), primary storage, output devices and auxiliary storage devices.
- ❖ **Software:** Computer software is a collection of programs used to manage the entire file system of the computer. It is also necessary for the running of computer hardware. The working of the computer hardware depends on the computer software.



2. Explain the various Problem-Solving Skills.

Problem solving technique is a set of techniques that helps in providing logic for solving a problem. Problem Solving Techniques

Problem solving can be expressed in the form of

- ❖ **Algorithm:** Algorithm is an ordered sequence of finite, well defined, unambiguous instructions for completing a task. It is an English-like representation of the logic which is used to solve the problem. It is a step- by-step procedure for solving a task or a problem
- ❖ **Flowchart:** Flow chart is defined as graphical or diagrammatic representation of the logic for problem solving. The purpose of flowchart is making the logic of the program

clear in a visual representation. A flowchart is a picture of the separate steps of a process in sequential order.

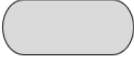


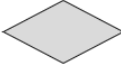

- ❖ **Pseudocode:** “Pseudo” means initiation. “Code” means the set of statements or instructions written in a programming language. Pseudo code consists of short, readable and formally styled English languages used for explaining an algorithm.

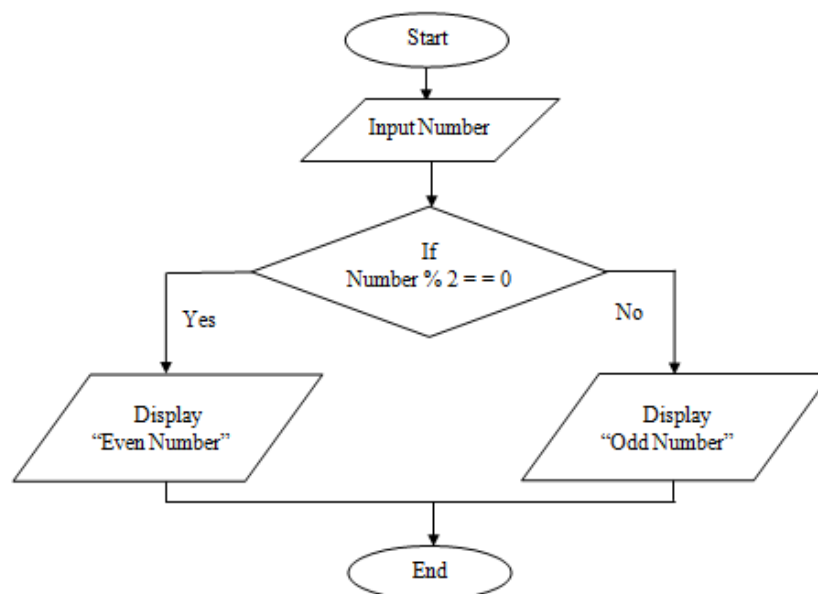
3. Write an Algorithm to find the Area of a Circle.

Algorithm to find the Area of the circle

- 1) Start
- 2) Read the value of radius r
- 3) Calculate - $\text{Area} = 3.14 * r * r$
- 4) Print the Area of the circle
- 5) Stop

4. What are the various symbols in Flowchart? Draw a Flowchart to check whether the given number is Even or Odd.

Symbol	Name	Function
	Oval	Represents the start or end of a process
	Rectangle	Denotes a process or operation step
	Arrow	Indicates the flow between steps
	Diamond	Signifies a point requiring a yes/no
	Parallelogram	Used for input or output operations



5. Write a Pseudocode to add two numbers.

```
BEGIN
    INPUT a,b
    ADD c=a+b
    PRINT c
END
```

6. Write a Python Interactive Mode and Python Interpreter

Python Interactive Mode:

- Python provides an **interactive mode**, where you can type Python commands and get results **immediately**, line by line.
- You can start it simply by typing python in the command prompt.
- The prompt looks like this:

```
>>> print("Hello")
```

Python Interpreter:

Python Interpreter translates Python code into machine-understandable actions and executes the code line by line. (that's why Python is called an interpreted language). Write Python code in a .py file, the interpreter reads the code line by line, converts the code into bytecode and executes them so the computer can perform the tasks you described.

7. Explain Arithmetic Operators in Python.

Arithmetic operators are used to execute arithmetic operations such as addition, subtraction, division, multiplication etc.

<i>Operator</i>	<i>Description</i>	<i>Example</i>	<i>Result</i>
+	Addition	print(5 + 3)	8
-	Subtraction	print(5 - 3)	2
*	Multiplication	print(5 * 3)	15
/	Division	print(5 / 2)	2.5
//	Floor Division	print(5 // 2)	2
%	Modulus (remainder)	print(5 % 2)	1
**	Exponentiation (power)	print(5 ** 2)	25

8. Explain Comparison Operators or Relational Operators in Python.

Comparison operators are used to compare the values of two operands. The result of the comparison is always a Boolean value, which can be either True or False. The operands can be Numbers, Strings, or Boolean values.

Operator	Description	Example	Result
<code>==</code>	Equal to	<code>print(5== 3)</code>	False
<code>!=</code>	Not equal to	<code>print (5!= 3)</code>	True
<code>></code>	Greater than	<code>print(5 > 3)</code>	True
<code><</code>	Less than	<code>print(5 < 3)</code>	False
<code>>=</code>	Greater than or equal to	<code>print(5 >= 3)</code>	True
<code><=</code>	Less than or equal to	<code>print(5 <= 3)</code>	False

9. Explain Logical Operators in Python.

Logical operators are used to compare or negate the logical values of their operands and return a resulting logical value. The operands evaluated by these operators yield either True or False. Consequently, the output of a logical operator is always a Boolean value, which can be either True or False.

<i>Operator</i>	Description	Example	Result
<i>and</i>	Logical AND	<code>print(True and False)</code>	False
<i>or</i>	Logical OR	<code>print(True or False)</code>	True
<i>not</i>	Logical NOT	<code>print(not True)</code>	False

10. Explain various Data Types in Python.

Data Types

Data types specify the type of data like numbers and characters to be stored and manipulated within a program. Basic data types of Python are

- ❖ Numbers
- ❖ Boolean
- ❖ Strings
- ❖ None

Number : Integers, floating-point numbers and complex numbers are all part of the Python numbers category. In Python, they are represented by the `'int'`, `'float'` and `'complex'` classes, respectively. Integers can be of any length, limited only by the available memory. Floating-point numbers are precise up to 15 decimal places. The distinction between integers and floating-point numbers is marked by the presence of a decimal point; for example, 1 is an integer, while 1.0 is a floating-point number. Complex numbers are expressed in the form $(x + yj)$, where (x) is the real part and (y) is the imaginary part.

Boolean: Boolean is essential when you start working with conditional statements. A condition is essentially a yes-or-no question, and the response to that question is a Boolean value, either True or False. The Boolean values, True and False, are considered reserved words.

String: A string is a sequence of one or more characters, which can include letters, numbers and various other types of characters. Strings can also contain spaces. You can represent strings using either single quotes or double quotes and they are often referred to as string literals. For multiline strings, you can use triple quotes, either `'''` or `"""`. These strings are fixed values that you provide literally in your script, rather than variables.

None:

None is another special data type in Python. None is frequently used to represent the absence of a value.

Category	Data Type	Description	Example
Numbers	int	Integer values (whole numbers).	42, -5, 0
	float	Floating-point numbers (decimal values).	3.14, -1.5
	complex	Complex numbers with real and imaginary parts.	3+4j, -2-5j
Boolean	bool	Logical values representing True or False.	True, False
Strings	str	Sequence of characters, enclosed in quotes.	"Hello", 'A'
None	NoneType	Represents the absence of a value or null.	None

11. Explain if..else statement with an example.

The if..else statement expands the functionality of the simple if statement by providing an alternative execution path when the initial condition evaluates to false. This construct allows programmers to handle multiple scenarios in a structured manner, ensuring that the code can respond appropriately regardless of the condition's outcome. The basic syntax of the if..else statement is as follows:

if condition:

 # code to execute if condition is true

else:

 # code to execute if condition is false

In this structure, the else block will only execute if the condition evaluates to false. This creates a clear dichotomy in the flow of execution, where one block of code is chosen over another based on the evaluation of the condition.

```
*book14.py - C:/Users/prasa/AppData/Local/Programs/Python/Python312/book14.py (3.12.0)*
File Edit Format Run Options Window Help
number = int(input("Enter a number"))
if number % 2 == 0:
    print(f"{number} is Even number")
else:
    print(f"{number} is Odd number")
```

12. Explain loops in Python

❖ while Loop

The while loop is a fundamental control flow statement in Python that allows for repeated execution of a block of code as long as a specified condition remains true. This loop is particularly useful in scenarios where the number of iterations is not predetermined, and the program needs to continue executing until a specific condition is met or until an external factor alters the flow.

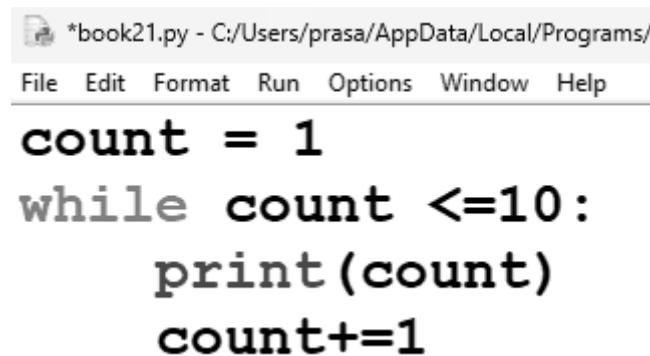
The syntax for a while loop is:

while condition:

 # code to execute as long as condition is true

In this structure, condition is a Boolean expression that is evaluated before each iteration. If the condition evaluates to True, the block of code within the loop is executed. Once the code block has been executed, the condition is re-evaluated. If it is still True, the loop continues; if it is False, the loop terminates, and the program proceeds to the next statement following the loop.

Program to print numbers from 1 to 10 using for loop

A screenshot of a Python IDE window titled '*book21.py - C:/Users/prasa/AppData/Local/Programs/'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code displayed is:

```
count = 1
while count <=10:
    print(count)
    count+=1
```

❖ for Loop

The for loop is a versatile control flow statement in Python that facilitates iteration over a sequence of elements. This capability allows programmers to traverse through collections such as lists, tuples, dictionaries, and strings with ease. The for loop is particularly powerful because it abstracts the complexity of managing an iteration index, thereby enhancing code readability and simplicity.

The basic syntax of a for loop in Python is as follows:

for element in *range(start , stop , step)::*

 # code to execute for each element


In this structure, element represents the current item in the iteration, and sequence is the collection being iterated over. For each iteration, the loop executes the block of code beneath it, processing each item in the sequence one at a time. Use range() function which is a built-in function at this stage as it is useful in demonstrating for loop. The range() function generates a sequence of numbers which can be iterated through using for loop.

Both start and step arguments are optional and the range argument value should always be an integer.

start = value indicates the beginning of the sequence. If the start argument is not specified, then the sequence of numbers start from zero by default.

Stop = Generates numbers up to this value but not including the number itself.

step = indicates the difference between every two consecutive numbers in the sequence. The step value can be both negative and positive but not zero.

 *book22.py - C:/Users/prasa/AppData/Local/Programs/Python/Python312/book22.py (3.12.0)*

File Edit Format Run Options Window Help

```
for i in range(10):
    print(f"{i}") # output: prints the values from 0 to 9
print("*****")
for i in range(1,5):
    print(f"{i}") # output: prints the values from 1 to 4
print("*****")
for i in range(1, 10, 3):
    print(f"{i}") # output: prints the values 1,4,7
```