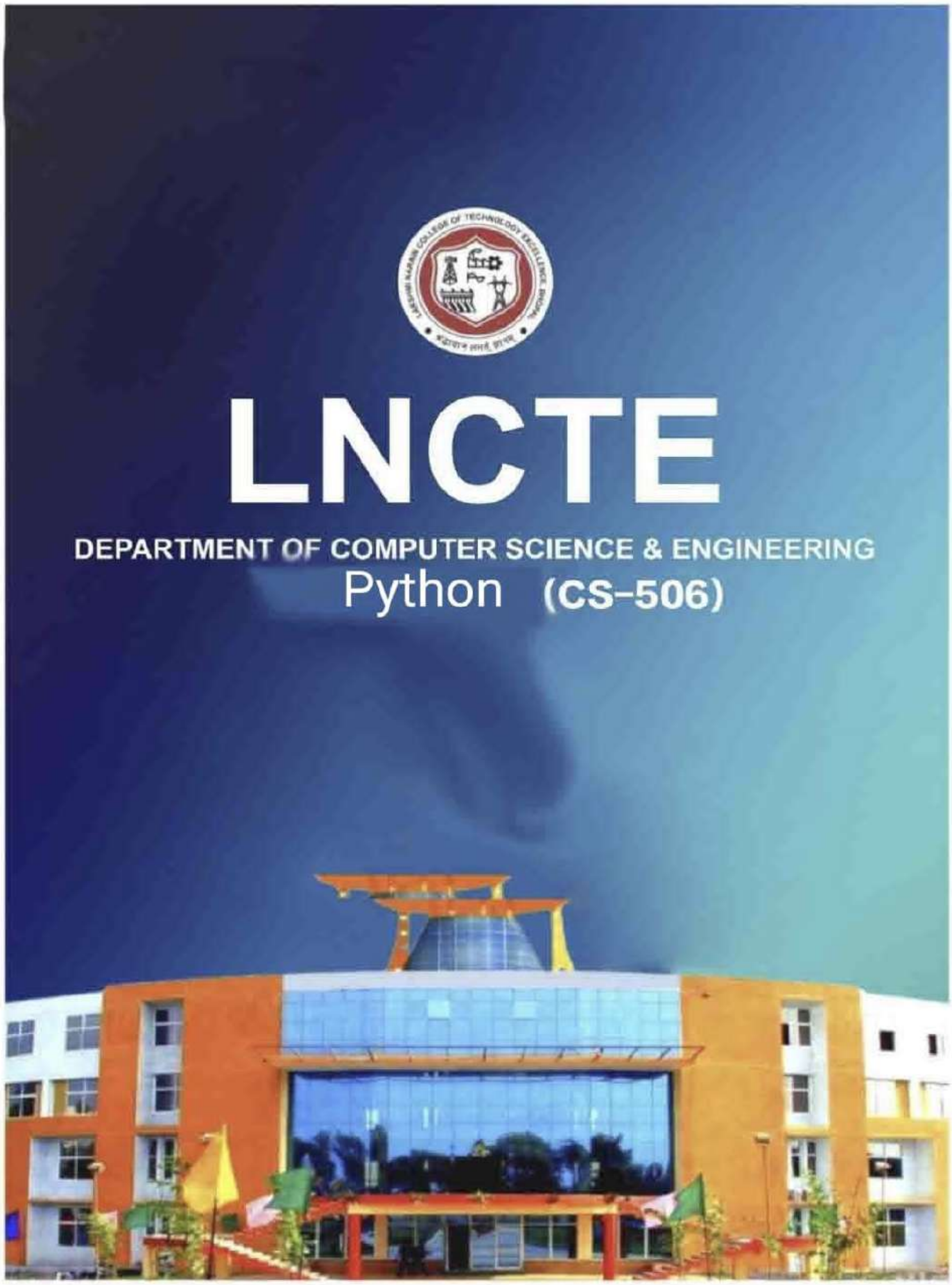


# LNCTE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Python (CS-506)

# Lakshmi Narain College of Technology Excellence

## Department of Computer Science and Engineering

**Lab Manual**

**Subject Name:** python

**Course Code:** CS 506

**Course:** B.Tech

**Session: 2023-24**

**Prepared By**

# TABLE OF CONTENT

## Vision and Mission of the Institute

### Vision of the institute

To become a pioneer institute in technical education and innovations to build competent technocrats and leaders for the nation.

### Mission of the institute

M1. To enhance the academic environment with innovative teaching learning processes and modern tools.

M2. To Practice and nurture high standards of human values, transparency and accountability.

M3. To collaborate with other academic and research institutes as well as industries in order to strengthen education and research.

M4. To uphold skill development for employability and entrepreneurship for interdisciplinary research and innovations.

## Vision and Mission of the Department

### Vision of the Department

To be a centre of excellence for providing quality technical education to develop future leaders with the aspects of research & computing, Software product development and entrepreneurship.

### Mission of the Department

| **Mission No.** | **Mission Statements** |
|---|---|
| **M1** | To offer academic program with state of art curriculum having flexibility for accommodating the latest developments in the areas of computer science and engineering |
| **M2** | To conduct research and development activities in contemporary and emerging areas of computer science & engineering. |
| **M3** | To inculcate moral values & entrepreneurial skills to produce professionals capable of providing socially relevant and sustainable solutions. |

## COURSE OUTCOMES: CS 506 – PYTHON

Students should be able to

| | |
|---|---|
| **CO506.1** | Recall basic concept of programming with Python. |
| **CO506.2** | Summerize premitive data structure in Python and application. |
| **CO506.3** | Solve statistical problems using python library. |
| **CO506.4** | Analyze real world data using visualization methods. |
| **CO506.5** | Create and evaluate models for real world problems applicable in recent machine |

## Course Articulation Matrix

| CO/PO | **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO10** | **PO11** | **PO12** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| CO506.1 | 3 | 2 | 3 | - | 2 | - | 2 | 1 | 2 | 2 | 2 | 3 |
| CO506.2 | 3 | 2 | 3 | - | 2 | - | 2 | 1 | 2 | 2 | 2 | 3 |
| CO506.3 | 3 | 3 | 3 | - | 2 | - | 2 | 1 | 1 | 2 | 2 | 3 |
| CO506.4 | 3 | 3 | 3 | - | 2 | - | 2 | 1 | 1 | 1 | 2 | 3 |
| CO506.5 | 3 | 2 | 2 | - | 2 | - | 2 | 1 | - | 1 | 1 | 3 |
| | **3** | **2.4** | **2.8** | **-** | **2** | **-** | **2** | **1** | **1.5** | **1.6** | **1.8** | **3** |

## Program Outcomes as defined by NBA (PO)

**Engineering Graduates will be able to:**

**1. Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

**2. Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

**3. Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## Program Specific Outcomes (PSO)

**A graduate of computer science and Engineering program will develop.**

**PSO 1:** An ability to apply technical Knowledge of computer science and engineering Fundamentals to become employable in industry.

**PSO 2:** An ability to develop programming skills using modern software tools and techniques.

**PSO 3:** An ability to develop real time projects for problem solving of domains such as Machine learning, Cyber security, Block chain And Big data.

**PSO 4:** An ability to grab research, higher studies and entrepreneurship opportunities Towards society with moral values and ethics.

## Program Educational Objectives (PEO):

**PEO-1:** Evolve as globally competent computer professionals, researchers and entrepreneurs possessing collaborative and leadership skills, for developing innovative solutions in multidisciplinary domains.

**PEO-2:** Excel as socially committed computer engineers having mutual respect, effective communication skills, high ethical values, and empathy for the needs of society.

**PEO-3:** involve in lifelong learning to faster the sustainable development in the emerging areas of technology.

**List of Program to be performed: -**

1) a) write a program in python to find simple interest?(CO1)

b) Find a square root of number by newton's method?(CO1)

c) Write a program to demonstrate different number data types in python (script.py)? (CO1)

d) Write a Python program to find the exponentiation of a number?(CO1)

2) a) Write a program remove multiple elements from a list in python? (CO2)

b) Write a Python Program to find the maximum from a list of numbers? (CO2)

c) Write a Python program to concatenate tuples to nested tuples? (CO2)

d) Write a Python Sort by Frequency of second element in Tuple List?(CO2)

3) a) Write a program to using different type of function update, remove, union,intersection(CO2)

b) Write a program iterate through all key value pair in a dictionary? Also explain its all type of function like :pop, pop item
, get? (C02)

4) a) Write a program to print given number is equal to 10,50,100?(CO3)

b) Write a python program to check whether an age is able to voting or not? (CO3)

c) Write a program to print marksheet? (CO3)

d) Program to print the largest of the three numbers. (CO3)

5) Write a Python program to perform binary, linear search? (CO3)

6) Write a Python program to perform selection, insertion sort? (CO3)

7) Write a Python program to perform merge, quick sort? (CO3)

8) a) Write a Python program to find first a prime numbers?(CO3)

b) Write a program to check whether a given number is Armstrong or not?(CO3)

c) Write a python program to multiply matrices?(CO3)

9) a) Write a python program to print calculator using class, object?(CO4)
b)Write a python program to print mark sheet using inheritance (using all type)(CO4)

10) Explain the try finally clause with suitable example? (CO4)

11) a) Write a Python program for command line arguments?(CO5)
b) Write a python program to print Date and Time?(CO5)

12) Write a Python program to find the most frequent words in a test read form a file?(CO5)

13) Write a Python program to simulate elliptical orbits in Pygame? (CO5)

14) Write a Python program to bouncing ball in Pygame?(CO5)

## What is python?

Python is an interpreter, object oriented, high level programming language with dynamic semantic Python syntax are easy compared other languages .

## Python History and Versions

Python laid its foundation in the late 1980s.The implementation of Python was started in December 1989 by Guido Van Rossum at CWI in Netherland.

In February 1991, Guido Van Rossum published the code (labeled version 0.9.0) to alt.sources.

In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce.

Python 2.0 added new features such as list comprehensions, garbage collection systems.

On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify the 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 the Amoeba Operating System.

The following programming languages influence Python:

ABC language.

Modula-3

**First Python Program**

=>we will run a simple program to print **Hello World** on the console.

Python provides us the two ways to run a program:

- Using Interactive interpreter prompt
- Using a script file

Syntax: print ”hello world”

## =>literal constants:

Python Literals can be defined as data that is given in a variable or constant.Python supports the following literals:

## 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 to create a string.

**Example:** 1 "Aman" , '12345'

**Types of Strings:** There are two types of Strings supported in Python:

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

**Example:** text1='hello'

**b) Multi-line String -** A piece of text that is written in multiple lines is known as multiple lines string.

There are two ways to create multiline strings:

**1) Adding black slash at the end of each line.**

**Example:** text1='hello\

 user'

 **print**(text1)

 'hellouser'

**2) Using triple quotation marks:-Example:**

str2='''welcome
to
SSSIT'''
**print** str2
Output: welcome to SSSIT

II. Numeric literals:

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

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

Example - Numeric Literals

```
    x = 0b10100 #Binary Literals
     y = 100 #Decimal Literal
     z = 0o215 #Octal Literal
    u = 0x12d #Hexadecimal Literal

#Float Literal
float_1 = 100.5
float_2 = 1.5e2

#Complex Literal
a = 5+3.14j

print(x, y, z, u)
print(float_1, float_2)
print(a, a.imag, a.real)
```

```
    Output:20 100 141 301

    100.5 150.0

    (5+3.14j) 3.14 5.0
```

III. Boolean literals: A Boolean literal can have any of the two values: True or False.

Example - Boolean Literals

```
x = (1 == True)
        y = (2 == False)
        z = (3 == True)
        a = True + 10
        b = False + 10


        print("x is", x)
        print("y is", y)
        print("z is", z)
print("a:", a)
print("b:", b)
```

Output:

```
x is True
y is False
z is False
a: 11
b: 10
```

## IV. 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 the end of lists in Python.

Example - Special Literals

```
val1=10
val2=None
print(val1)
print(val2)
```

Output:

```
10
None
```

V. Literal Collections. Python provides the four types of literal collection such as List literals, Tuple literals, Dict literals, and Set literals.

## List:

- List contains items of different data types. Lists are mutable i.e., modifiable.
- The values stored in List are separated by comma(,) and enclosed within square brackets([]). We can store different types of data in a List.

Example - List literals

```
list=['John',678,20.4,'Peter']
list1=[456,'Andrew']
print(list)
print(list + list1)
```

Output:

```
['John', 678, 20.4, 'Peter']

['John', 678, 20.4, 'Peter', 456, 'Andrew']
```

## Dictionary:

- Python dictionary stores the data in the key-value pair.
- It is enclosed by curly-braces {} and each pair is separated by the commas(,).

Example

```
dict = {'name': 'Pater', 'Age':18,'Roll_nu':101}
print(dict)
```

Output: {'name': 'Pater', 'Age': 18, 'Roll_nu': 101}

## Tuple:

- Python tuple is a collection of different data-type. It is immutable which means it cannot be modified after creation.
- It is enclosed by the parentheses () and each element is separated by the comma(,).

Example

```
tup = (10,20,"Dev",[2,3,4])
print(tup)
```

Output: (10, 20, 'Dev', [2, 3, 4])

## Set:

- Python set is the collection of the unordered dataset.
- It is enclosed by the {} and each element is separated by the comma(,).

Example: - Set Literals

```
set = {'apple','grapes','guava','papaya'}
print(set)
```

Output: {'guava', 'apple', 'papaya', 'grapes'}

**Python Variables:** Variable is a name that is used to refer to memory location. Python variable is also known as an identifier and used to hold value.In Python, we don't need to specify the type of variable because Python is a infer language and smart enough to get variable type.Variable names can be a group of both the letters and digits, but they have to begin with a letter or an underscore.

It is recommended to use lowercase letters for the variable name. Rahul and rahul both are two different variables.

**Identifier Naming:** Variables are the example of identifiers. An Identifier is used to identify the literals used in the program. The rules to name an identifier are given below.

- The first character of the variable must be an alphabet or underscore ( _ ).
- All the characters except the first character may be an alphabet of lower-case(a-z), upper-case (A-Z), underscore, or digit (0-9).
- Identifier name must not contain any white-space, or special character (!, @, #, %, ^, &, *).
- Identifier name must not be similar to any keyword defined in the language.
- Identifier names are case sensitive; for example, my name, and MyName is not the same.
- Examples of valid identifiers: a123, _n, n_9, etc.
- Examples of invalid identifiers: 1a, n%4, n 9, etc.

Declaring Variable and Assigning Values Python does not bind us to declare a variable before using it in the application. It allows us to create a variable at the required time.We don't need to declare explicitly variable in Python. When we assign any value to the variable, that variable is declared automatically.

The equal (=) operator is used to assign value to a variable .Object References It is necessary to understand how the Python interpreter works when we declare a variable. The process of treating variables is somewhat different from many other programming languages.

Python is the highly object-oriented programming language; that's why every data item belongs to a specific type of class. Consider the following example.

```
1. print("John")
```

Output:John

The Python object creates an integer object and displays it to the console. In the above print statement, we have created a string object. Let's check the type of it using the Python built-in type() function.

```
1. type("John")
```

Output:

```
<class 'str'>
```

**Operator and Expressions:** The operator is a symbol that performs a certain operation between two operands, according to one definition. In a particular programming language, operators serve as the foundation upon which logic is constructed in a programme. The different operators that Python offers are listed here.

- Arithmetic operators
- Comparison operators
- Assignment Operators

- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

**Arithmetic Operators:**Arithmetic operations between two operands are carried out using arithmetic operators. It includes the exponent (**) operator as well as the + (addition), - (subtraction),* (multiplication), / (divide), % (reminder), and // (floor division)

| **Operator** | **Description** |
|---|---|
| **+ (Addition)** | It is used to add two operands. For example, if a = 10, b = 10 ->a+b = 20 |
| **- (Subtraction)** | It is used to subtract the second operand from the first operand. If the first operand is less than the second operand, the value results negative. For example, if a = 20, b = 5 -3.> a - b = 15 |
| **/ (divide)** | It returns the quotient after dividing the first operand by the second operand. For example, if a = 20, b = 10 -> a/b = 2.0 |
| ***** (Multiplication)** | It is used to multiply one operand with the other. For example, if a = 20, b = 4 -> a * b = 80 |
| **% (reminder)modulus** | It returns the reminder after dividing the first operand by the second operand. For example, if a = 20, b = 10 ->a%b = 0 |
| **** (Exponent)** | As it calculates the first operand's power to the second operand, it is an exponent operator. |
| **//(Floor division)** | It provides the quotient's floor value, which is obtained by dividing the two operands. |

**Comparison operator:**Comparison operators compare the values of the two operands and return a true or false Boolean value in accordance. The following table lists the comparison operators.

| **Operator** | **Description** |
|---|---|
| == | If the value of two operands is equal, then the condition becomes true. |

| | |
|---|---|
| != | If the value of two operands is not equal, then the condition becomes true. |
| <= | The condition is met if the first operand is smaller than or equal to the second operand. |
| >= | The condition is met if the first operand is greater than or equal to the second operand. |
| > | If the first operand is greater than the second operand, then the condition becomes true. |
| < | If the first operand is less than the second operand, then the condition becomes true. |

**Assignment Operators:**The right expression's value is assigned to the left operand using the assignment operators. The following table provides a description of the assignment operators.

| **Operator** | **Description** |
|---|---|
| = | It assigns the value of the right expression to the left operand. |
| += | By multiplying the value of the right operand by the value of the left operand, the left operand receives a changed value. For example, if a = 10, b = 20 -> a+ = b will be equal to a = a+ b and therefore, a = 30. |
| -= | It decreases the value of the left operand by the value of the right operand +and and assigns the modified value back to left operand. For example, if a = 20, b = 10 -> a-= b will be equal to a = a- b and therefore, a = 10. |
| *= | It multiplies the value of the left operand by the value of the right operand and assigns the modified value back to then the left operand. For example, if a = 10, b = 20 -> a*= b will be equal to a = a* b and therefore, a = 200. |
| %= | It divides the value of the left operand by the value of the right operand and assigns the reminder back to the left operand. For example, if a = 20, b = 10 -> a % = b will be equal to a = a % b and therefore, a = 0. |
| **= | a**=b will be equal to a=a**b, for example, if a = 4, b = 2, a**=b will assign 4**2 = 16 to a. |
| //= | A//=b will be equal to a = a// b, for example, if a = 4, b = 3, a//=b will assign 4//3 = 1 to a. |

**Bitwise Operators:** The two operands' values are processed bit by bit by the bitwise operators.

| **Operator** | **Description** |
|---|---|
| &(binary and) | A 1 is copied to the result if both bits in two operands at the same location are 1. If not, 0 is copied. |

| | |
|---|---|
| \| (binary or) | The resulting bit will be 0 if both the bits are zero; otherwise, the resulting bit will be 1. |
| ^ (binary xor) | If the two bits are different, the outcome bit will be 1, else it will be 0. |
| ~ (negation) | The operand's bits are calculated as their negations, so if one bit is 0, the next bit will be 1, and vice versa. |
| << (left shift) | The number of bits in the right operand is multiplied by the leftward shift of the value of the left operand. |
| >> (right shift) | The left operand is moved right by the number of bits present in the right operand. |

**Logical Operators:**The assessment of expressions to make decisions typically makes use of the logical operators. The following

logical operators are supported by Python.

| **Operator** | **Description** |
|---|---|
| And | The condition will also be true if the expression is true. If the two expressions a and b are the same, then a and b must both be true. |
| Or | The condition will be true if one of the phrases is true. If a and b are the two expressions, then a or b must be true in one of them. |
| Not | If an expression **a** is true, then not (a) will be false and vice versa. |

**Membership Operators:**The membership of a value inside a Python data structure can be verified using Python membership operators. The result is true if the value is in the data structure; otherwise, it returns false.

| **Operator** | **Description** |
|---|---|
| In | If the first operand cannot be found in the second operand, it is evaluated to be true (list, tuple, or dictionary). |
| not in | If the first operand is not present in the second operand, the evaluation is true (list, tuple, or dictionary). |

**Identity Operators:**

| **Operator** | **Description** |
|---|---|
| Is | If the references on both sides point to the same object, it is determined to be true. |

| | |
|---|---|
| is not | If the references on both sides do not point at the same object, it is determined to be true. |

**Operator Precedence:** The order in which the operators are examined is crucial to understand since it tells us which operator needs to be considered first. Below a list of the Python operators' precedence tables.

| Operator | Description |
|---|---|
| ** | Overall other operators employed in the expression, the exponent operator is given precedence. |
| ~ + - | the minus, unary plus, and negation. |
| * / % // | the division of the floor, the modules, the division, and the multiplication. |
| + - | Binary plus, and minus |
| >><< | Left shift. and right shift |
| & | Binary and. |
| ^ \| | Binary xor, and or |
| <= <>>= | Comparison operators (less than, less than equal to, greater than, greater then equal to). |
| <> == != | Equality operators. |
| = %= /= //= -= += *= **= | Assignment operators |
| is is not | Identity operators |
| in not in | Membership operators |
| not or and | Logical operators |

**Experiment-1**

**a)Write a program in python to find  simple interest?**

```
P=float(input("enter the principle…"))

r=float(input("enter the rate…."))
```

```
t=float(input("enter the time….."))

s.i=p*r*t/100

print(s.i)

output:

enter the principle…1000

enter the rate…5

enter the time…6

s.i=300
```

**b) Find a square root of number by newton's method?**

```
num=int(input("Enter a number:"))
newtonSquareroot= 0.5*((0.5*num)+num/(0.5*num))
print (newtonSquareroot)

output: Enter a  number:2

1.5
```

**c) write a program to demonstrate different number data types in python(script.py)?**

```
num=5

print(num1,"is of type",type(num))

num=50.20

print(num1,"is of type",type(num))

num=1+2j

print(num1,"is of type",type(num))

output:

5 is of type<class 'int'>

50.20 is of type<class 'float'>

1+2j is of type<class 'complex'>
```

**d) Write a Python program to find the exponentiation of a number?**

```
Base=int(input("enter the number:"))
```

```
Exponent=int(input("enter exponent value:"))

c=base**exponent

Print( c)
```

**<u>Exercise Questions:</u>**
a) Write a python program to print a number is positive/negative .
b) Write a python program to find largest number among three numbers.

**<u>Viva question:</u>**
1) What is python? Why it's called interpreted language?

2) What is variable ,constants ,indentation in python?

**<u>Python List:</u>**A list in Python is used to store the sequence of various types of data. Python lists are mutable type its mean we can modify its element after it created. However, Python consists of six data-types that are capable to store the sequences, but the most common and reliable type is the list.

A list can be defined as a collection of values or items of different types. The items in the list are separated with the comma (,) and
enclosed with the square brackets [].
A list can be define as below

- L1 = ["John", 102, "USA"]
- L2 = [1, 2, 3, 4, 5, 6]

## Python List Operations

The concatenation (+) and repetition (*) operators work in the same way as they were working with the strings.
Let's see how the list responds to various operators.
1. Consider a Lists l1 = [1, 2, 3, 4], **and** l2 = [5, 6, 7, 8] to perform operation.

| **Operator** | **Description** | **Example** |
|---|---|---|
| Repetition | The repetition operator enables the list elements to be repeated multiple times. | L1*2 = [1, 2, 3, 4, 1, 2, 3, 4] |
| Concatenation | It concatenates the list mentioned on either side of the operator. | l1+l2 = [1, 2, 3, 4, 5, 6, 7, 8] |
| Membership | It returns true if a particular item exists in a particular list otherwise false. | print(2 in l1) prints True. |
| Iteration | The for loop is used to iterate over the list elements. | for i in l1:<br>   print(i)<br>**Output**<br>1,2,3,4 |
| Length | It is used to get the length of the list | len(l1) = 4 |

| **SN** | **Function** | **Description** | **Example** |
|---|---|---|---|
| 1 | cmp(list1, list2) | It compares the elements of both the lists. | This method is not used in the Python 3 and the above versions. |
| 2 | len(list) | It is used to calculate the length of the list. | L1 = [1,2,3,4,5,6,7,8]<br>print(len(L1))<br>8 |

| | | | |
|---|---|---|---|
| 3 | max(list) | It returns the maximum element of the list. | L1 = [12,34,26,48,72]<br>print(max(L1))<br>72 |
| 4 | min(list) | It returns the minimum element of the list. | L1 = [12,34,26,48,72]<br>print(min(L1))<br>12 |
| 5 | list(seq) | It converts any sequence to the list. | str = "Johnson"<br>s = list(str)<br>print(type(s))<br>&lt;class list&gt; |
| 6 | append() | Add single element to a list | L1=[10,20,30,40,50]<br>Append(l1(100)<br>Print(l1)<br>Output 10,20,30,40,50,100 |
| 7 | Extend() | Adds multiple element to a list | L1=[10,20,30,40,50]<br>extend(l1(100,200,300,400)<br>Print(l1)<br>Output 10,20,30,40,50,100,200,300,400 |

**Python Tuple:**Python Tuple is used to store the sequence of immutable Python objects. The tuple is similar to lists since the value of the items stored in the list can be changed, whereas the tuple is immutable, and the value of the items stored in the tuple cannot be changed.

**Creating a tuple:**A tuple can be written as the collection of comma-separated (,) values enclosed with the small () brackets. The parentheses are optional but it is good practice to use. A tuple can be defined as follows.

```
T1 = (101, "Peter", 22)
print(type(T1))
```

| **Operator** | **Description** | **Example** |
|---|---|---|
| Repetition | The repetition operator enables the tuple elements to be repeated multiple times. | T1*2 = (1, 2, 3, 4, 5, 1, 2, 3, 4, 5) |

| | | |
|---|---|---|
| Concatenation | It concatenates the tuple mentioned on either side of the operator. | T1+T2 = (1, 2, 3, 4, 5, 6, 7, 8, 9) |
| Membership | It returns true if a particular item exists in the tuple otherwise false | print (2 in T1) prints True. |
| Iteration | The for loop is used to iterate over the tuple elements. | for i in T1:<br>print(i)<br>**Output** [illegible] |
| Length | It is used to get the length of the tuple. | len(T1) = 5 |

**<u>Python set:</u>**A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.

Unlike other collections in Python, there is no index attached to the elements of the set, i.e., we cannot directly access any element of the set by the index. However, we can print them all together, or we can get the list of elements by looping through the set.

*Python Built-in set methods:*

Python contains the following methods to be used with the sets.

| SN | Method | Description |
|---|---|---|
| 1 | add(item) | It adds an item to the set. It has no effect if the item is already present in the set. |
| 2 | clear() | It deletes all the items from the set. |
| 3 | copy() | It returns a shallow copy of the set. |
| 4 | difference update(....) | It modifies this set by removing all the items that are also present in the specified sets. |
| 5 | discard(item) | It removes the specified item from the set. |
| 6 | intersection() | It returns a new set that contains only the common elements of both the sets. (all the sets if more than two are specified). |
| 7 | intersection_update(....) | It removes the items from the original set that are not present in both the sets (all the sets if more than one are specified). |
| 8 | Isdisjoint(....) | Return True if two sets have a null intersection. |
| 9 | Issubset(....) | Report whether another set contains this set. |
| 10 | Issuperset(....) | Report whether this set contains another set. |
| 11 | pop() | Remove and return an arbitrary set element that is the last element of the set. Raises KeyError if the set is empty. |
| 12 | remove(item) | Remove an element from a set; it must be a member. If the element is not a member, raise a KeyError. |
| 13 | symmetric_difference(....) | Remove an element from a set; it must be a member. If the element is not a member, raise a KeyError. |
| 14 | symmetric_difference_update(....) | Update a set with the symmetric difference of itself and another. |
| 15 | union(....) | Return the union of sets as a new set.(i.e. all elements that are in either set.) |
| 16 | update() | Update a set with the union of itself and others. |

**Python Dictionary** :Python Dictionary is used to store the data in a key-value pair format. The dictionary is the data type in Python, which can simulate the real-life data arrangement where some specific value exists for some particular key. It is the mutable data-structure. The dictionary is defined into element Keys and values.

Keys must be a single element
Value can be any type such as list, tuple, integer, etc.

In other words, we can say that a dictionary is the collection of key-value pairs where the value can be any Python object. In contrast, the keys are the immutable Python object, i.e., Numbers, string, or tuple.

**Built-in Dictionary functions:**

| SN | Function | Description |
|---|---|---|
| 1 | cmp(dict1, dict2) | It compares the items of both the dictionary and returns true if the first dictionary Values are greater than the second dictionary, otherwise it returns false. |
| 2 | len(dict) | It is used to calculate the length of the dictionary. |
| 3 | str(dict) | It converts the dictionary into the printable string representation. |
| 4 | type(variable) | It is used to print the type of the passed variable. |

The built-in python dictionary methods along with the description are given below.

| SN | Method | Description |
|---|---|---|
| 1 | dic.clear() | It is used to delete all the items of the dictionary. |
| 2 | dict.copy() | It returns a shallow copy of the dictionary. |
| 3 | dict.fromkeys(iterable, value = None, /) | Create a new dictionary from the iterable with the values equal to value. |
| 4 | dict.get(key, default = "None") | It is used to get the value specified for the passed key. |
| 5 | dict.has_key(key) | It returns true if the dictionary contains the specified key. |
| 6 | dict.items() | It returns all the key-value pairs as a tuple. |
| 7 | dict.keys() | It returns all the keys of the dictionary. |
| 8 | dict.setdefault(key,default= "None") | It is used to set the key to the default value if the key is not specified in the dictionary |
| 9 | dict.update(dict2) | It updates the dictionary by adding the key-value pair of dict2 to this dictionary. |
| 10 | dict.values() | It returns all the values of the dictionary. |
| 11 | len() | It is used to calculate the length of the dictionary. |

## Experiment-2

**a) Write a program remove multiple elements from a list in python?**

```
list=["Bran",11,22,33,"Stark",22,33,11]
list.remove(22)
 print(list)
output: [‘Bran’, 11, 33, ‘Stark’, 22, 33, 11]
```

**b) Write a Python Program to find the maximum from a list of numbers?**

```
list=[3,2,8,5,10,6]
max_number=max(list);
print(“max_number:”,max_number)
min_number=min(list);
print(“min_number:”,min_number)

output:max_number :10
min_number :2
```

**c) Write a Python program to concatenate tuples to nested tuples?**

```
my_tuple_1 = ( 7, 8, 3, 4, 3, 2),
my_tuple_2 = (9, 6, 8, 2, 1, 4),

print ("The first tuple is : " )
print(my_tuple_1)
print ("The second tuple is : " )
print(my_tuple_2)

my_result = my_tuple_1 + my_tuple_2

print("The tuple after concatenation is : " )
print(my_result)
```

output:

```
The first tuple is :((7, 8, 3, 4, 3, 2),)
The second tuple is :((9, 6, 8, 2, 1, 4),)
The tuple after concatenation is :((7, 8, 3, 4, 3, 2), (9, 6, 8, 2, 1, 4))
```

**d) Write a Python Sort by Frequency of second element in Tuple List?**

```
t = (('a', 53), ('b', 37), ('c', 23), ('d', 1), ('e', 18))

t = tuple(sorted(list(t), key=lambda x: x[1]))

print(t)
```

output:

```
(('d', 1), ('e', 18), ('c', 23), ('b', 37), ('a', 53))
```

**<u>Exercise Questions:</u>**

a) Write a Python Program to find the maximum from a list of numbers?
b) Write a python program to find the index, and count the element in the list and tuple?

c) Write a python program to find sum of the element in the list?

**Viva question:**

1) Difference between list and tuple?
2) What is indexing And slicing in list?
3) What is difference between Append and Extend?

## Experiment-3

### a) Write a program to using different type of function update, remove, union,intersection?

```
A ={0, 2, 4, 6, 8};
B ={1, 2, 3, 4, 5};

# union
print("Union :", A | B)

# intersection
print("Intersection :", A & B)

# difference
print("Difference :", A -B)

# symmetric difference
print("Symmetric difference :", A ^ B)
```

Output:

```
('Union :', set([0, 1, 2, 3, 4, 5, 6, 8]))

('Intersection :', set([2, 4]))

('Difference :', set([8, 0, 6]))

('Symmetric difference :', set([0, 1, 3, 5, 6, 8]))
```

**b)Write a program iterate through all key value pair in a dictionary? Also explain its all type of function like :pop, pop item , get?**

```
my_dict = {"name": "John", "age": 30, "gender": "Male"}
  for key, value in my_dict.items():
print(key, ":", value)
```

Output:

```
name : John
age : 30
gender : Male
```

```
inventory = {'shirts': 25, 'paints': 220, 'shock': 525, 'tshirts': 217}
element = inventory.pop('shoes')
print(element)
```

Output:

```
KeyError: 'shoes'
```

```
inventory = {'shirts': 25, 'paints': 220, 'shocks': 525, 'tshirts': 217}
# Displaying result
print(inventory)
p = inventory.popitem()
print("Removed",p)
print(inventory)
```

Output:

```
{'shirts': 25, 'paints': 220, 'shocks': 525, 'tshirts': 217}
Removed ('tshirts', 217)
{'shirts': 25, 'paints': 220, 'shocks': 525}
```

## Exercise Questions:

a) Write a python program to check whether a given key already exist in a dictionary?
b) Write a python program to remove an elements from a given set and also find the elements in a given set that are not in another set?

## Viva question:

1) Difference between set and dictionary?
2) What is frozen set.?

**Python If-else statements**:Decision making is the most important aspect of almost all the programming languages. As the name implies, decision making allows us to run a particular block of code for a particular decision. Here, the decisions are made on the validity of the particular conditions. Condition checking is the backbone of decision making.

| **Statement** | **Description** |
|---|---|
| If Statement | The if statement is used to test a specific condition. If the condition is true, a block of code (if-block) will be executed. |

| | |
|---|---|
| If - else Statement | The if-else statement is similar to if statement except the fact that, it also provides the block of the code for the false case of the condition to be checked. If the condition provided in the if statement is false, then the else statement will be executed. |
| Nested if Statement | Nested if statements enable us to use if ? else statement inside an outer if statement. |

**if statement:** The if statement is used to test a particular condition and if the condition is true, it executes a block of code known as if-block. The condition of if statement can be any valid logical expression which can be either evaluated to true or false.

The syntax of the if-else statement is given below.

```
if expression:
    statement
```

**if-else statement:** The if-else statement provides an else block combined with the if statement which is executed in the false case of the condition. If the condition is true, then the if-block is executed. Otherwise, the else-block is executed.

Syntax:

```
if condition:
#block of statements
else:
#another block of statements (else-block)
```

**Elif statement**:enables us to check multiple conditions and execute the specific block of statements depending upon the true condition among them. We can have any number of elif statements in our program depending upon ourneed. However, using elif is optional.Theelif statement works like an if-else-if ladder statement in C. It must be succeeded by an if statement.The syntax of the elif statement is given below.

```
if expression 1:
   # block of statements
elif expression 2:
   # block of statements
elif expression 3:
# block of statements
else:
# block of statements
```

## Experiment-4

### a) Write a program to print given number is equal to 10,50,100?

```
number = int(input("Enter the number?"))
```

```
if number==10:
print("number is equals to 10")
elif number==50:
print("number is equal to 50");
elif number==100:
print("number is equal to 100");
else:
print("number is not equal to 10, 50 or 100");
```

Output:

```
Enter the number?15
number is not equal to 10, 50 or 100
```

**b)Write a python program to check whether an age is able to voting or not?**

```
age =int(input("Enter your age: "))
ifage >=18:
print("Person is allowed to vote")
else:
print("Person is not allowed to vote")
```

output

```
enter your age:32
Person is allowed to vote
```

**c)Write a program to print marksheet?**

```
marks = int(input("Enter the marks? "))
if marks > 85 and marks <= 100:
  print("Congrats ! you scored grade A ...")
elif marks > 60 and marks <= 85:
  print("You scored grade B + ...")
elif marks > 40 and marks <= 60:

  print("You scored grade B ...")
elif (marks > 30 and marks <= 40):
  print("You scored grade C ...")
else:
  print("Sorry you are fail ?")
```

Output:

```
Enter the marks? 89
Congrats ! you scored grade A ...
```

**d)Program to print the largest of the three numbers?**

```
a = int (input("Enter a: "));
b = int (input("Enter b: "));
c = int (input("Enter c: "));
if a>b and a>c:
    print ("From the above three numbers given a is largest");
if b>a and b>c:
    print ("From the above three numbers given b is largest");
if c>a and c>b:
 print ("From the above three numbers given c is largest");
```

Output:

```
Enter a: 100
Enter b: 120
Enter c: 130
From the above three numbers given c is largest
```

## Exercise Questions:

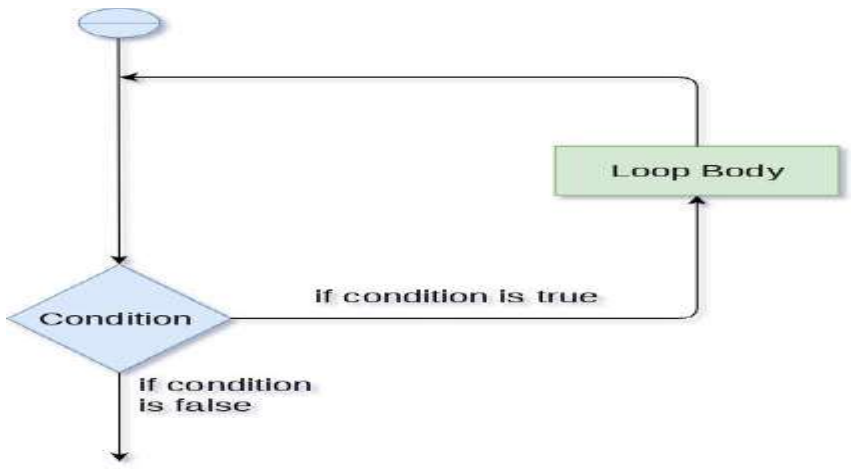
a) Write a Python Program to check whether a number is even or not?

b) Write a python program to Simple Python program to understand elif statement ?

## Viva question:

1) What is elif statement?

**Python Loops**:The flow of the programs written in any programming language is sequential by default. Sometimes we may need to alter the flow of the program. The execution of a specific code may need to be repeated several numbers of times.For this purpose, The programming languages provide various types of loops which are capable of repeating some specific code several numbers of times. Consider the following diagram to understand the working of a loop statement.



## Why we use loops in python?

The looping simplifies the complex problems into the easy ones. It enables us to alter the flow of the program so that instead of writing the same code again and again, we can repeat the same code for a finite number of times. For example, if we need to print the first 10 natural numbers then, instead of using the print statement 10 times, we can print inside a loop which runs up to 10 iterations.

Advantages of loops:

There are the following advantages of loops in Python.

1. It provides code re-usability.
2. Using loops, we do not need to write the same code again and again.
3. Using loops, we can traverse over the elements of data structures (array or linked lists).

There are the following loop statements in Python.

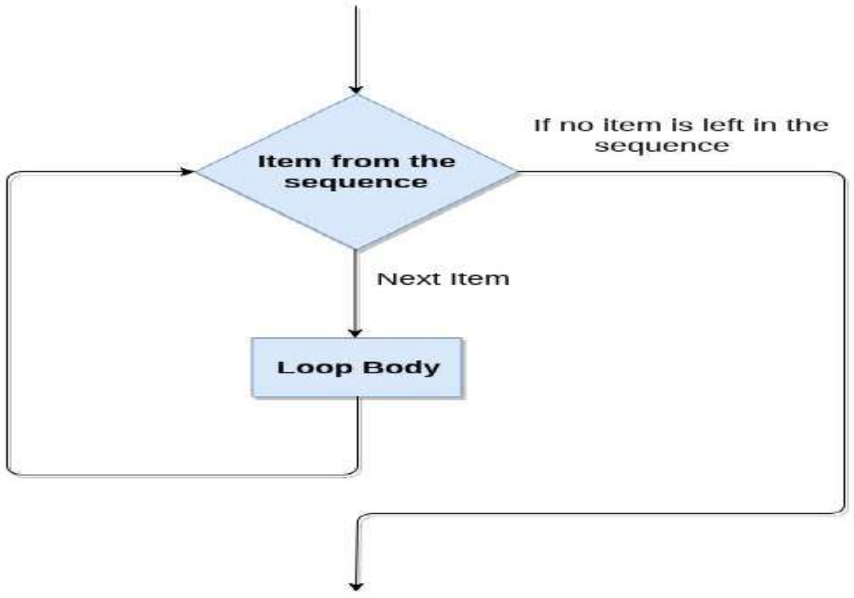
| **Loop Statement** | **Description** |
|---|---|
| for loop | The for loop is used in the case where we need to execute some part of the code until the given condition is satisfied. The for loop is also called as a per-tested loop. It is better to use for loop if the number of iteration is known in advance. |
| while loop | The while loop is to be used in the scenario where we don't know the number of iterations in advance. The block of statements is executed in the while loop until the condition specified in the while loop is satisfied. It is also called a pre-tested loop. |
| do-while loop | The do-while loop continues until a given condition satisfies. It is also called post tested loop. It is used when it is necessary to execute the loop at least once (mostly menu driven programs). |

**<u>Python for loop</u>**:The for **loop in Python** is used to iterate the statements or a part of the program several times. It is frequently used to traverse the data structures like list, tuple, or dictionary.

The syntax of for loop in python is given below.

```
For var in sequence:
   statement(s)
```

<u>The for loop flowchart:</u>



For loop Using Sequence

Ex: Iterating string using for loop

```
str = "Python"
for i in str:
print(i)
```

output:

```
P
y
t
h
o
n
```

## For loop Using range() function

**The range() function**

The **range()** function is used to generate the sequence of the numbers. If we pass the range(10), it will generate the numbers from 0 to 9. The syntax of the range() function is given below.

**Syntax:**

range(start,stop,step size)

The start represents the beginning of the iteration.

The stop represents that the loop will iterate till stop-1. The **range(1,5)** will generate numbers 1 to 4 iterations. It is optional.

The step size is used to skip the specific numbers from the iteration. It is optional to use. By default, the step size is 1. It is optional.
Consider the following examples:

Example-1: Program to print numbers in sequence.

```
for i in range(10):
print(i,end = ' ')
Output:
0 1 2 3 4 5 6 7 8 9
```

**<u>Nested for loop in python</u>:** Python allows us to nest any number of for loops inside a for loop. The inner loop is executed n number of times for every iteration of the outer loop. The syntax is given below.

Syntax

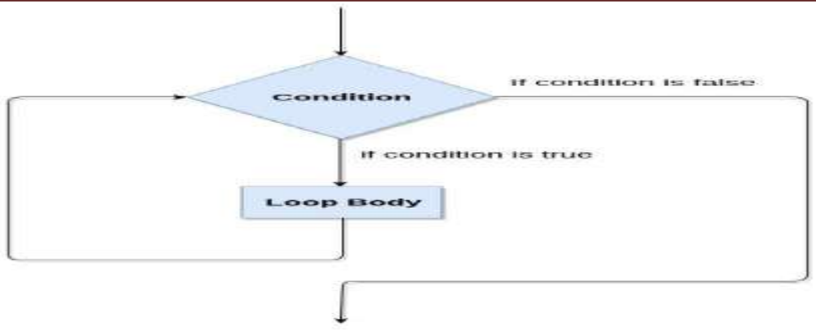
```
for iterating_var1 in sequence:  #outer loop
    for iterating_var2 in sequence:  #inner loop
        #block of statements
#Other statements
```

```
Ex:# User input for number of rows
      rows = int(input("Enter the rows:"))
      # Outer loop will print number of rows
      for i in range(0,rows+1):
      # Inner loop will print number of Astrisk
      for j in range(i):
      print("*",end = '')
      print()
     Output: Enter the rows: 5

           *
           **
           ***
           ****
           *****
```

**<u>Python While loop:</u>**The Python while loop allows a part of the code to be executed until the given condition returns false. It is also known as a pre-tested loop. It can be viewed as a repeating if statement. When we don't know the number of iterations then the while loop is most effective to use.

```
Syntax: while expression:
           statements
```

While loop Flowchart



**<u>Python Do While Loop</u> :**Python doesn't have do-while loop. But we can create a program like this.The do while loop is used to check condition after executing the statement. It is like while loop but it is executed at least once.

General Do While Loop Syntax

```
do {
    //statement
} while (condition);
```

**<u>Python break statement</u>**:The break is a keyword in python which is used to bring the program control out of the loop. The break statement breaks the loops one by one, i.e., in the case of nested loops, it breaks the inner loop first and then proceeds to outer loops. In other words, we can say that break is used to abort the current execution of the program and the control goes to the next line after the loop.
The break is commonly used in the cases where we need to break the loop for a given condition.
The syntax of the break is given below.

```
#loop statements
break;
```

Ex:

```
str = "python"
 for i in str:
   if i == 'o':
       break
   print(i);
```

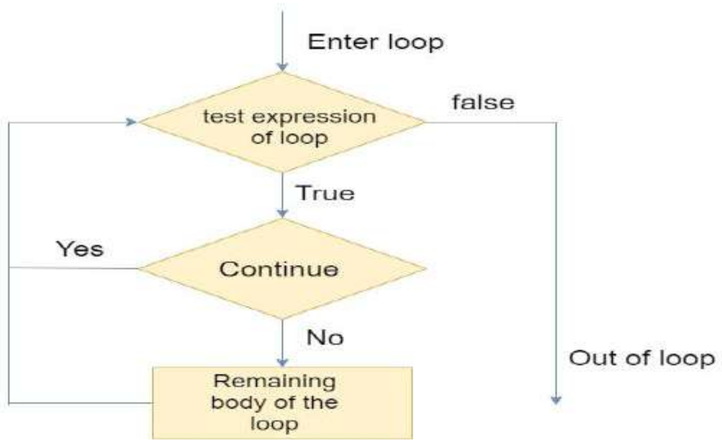
Output:

```
p
y
t
h
```

**Python continue Statement**:The continue statement in Python is used to bring the program control to the beginning of the loop. The continue statement skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition.The continue statement in Python is used to bring the program control to the beginning of the loop. The continue statement skips the remaining lines of code inside the loop and start with the next iteration. It is mainly used for a particular condition inside the loop so that we can skip some specific code for a particular condition.

```
Syntax: #loop statements
            continue
        #the code to be skipped
```

Flow Diagram



Example 1

```
i = 0
while(i < 10):
  i = i+1
if(i == 5):
    continue
  print(i)
```

Output:

```
1
2
3
4
6
7
8
```

```
9
10
```

**Python Pass:**In Python, the pass keyword is used to execute nothing; it means, when we don't want to execute code, the pass can be used to execute empty. It is the same as the name refers to. It just makes the control to pass by without executing any code. If we want to bypass any code pass statement can be used.

It is beneficial when a statement is required syntactically, but we want or we don't want to execute or execute it later. The difference between the comments and pass is that, comments are entirely ignored by the Python interpreter, where the pass statement is not ignored.

Suppose we have a loop, and we do not want to execute right this moment, but we will execute in the future. Here we can use the pa

Ex:

```
for i in [1,2,3,4,5]:
   if(i==4):
      pass
print("This is pass block", i)
   print(i)
```

Output:

```
1
2
3
This is pass block 4
4
5
```

## Excersice-5

**a) Write a Python program to perform binary, linear search?**

**Linear search:**

```
nums= [4,9,15,21,25,28,35,38,40,45]
item = 38
value_found= False

for i in range(len(nums)):

   if nums[i] == item:
value_found = True
print("Item found at index ", i)
      break
```

```
if value_found == False:
print("Item not found")
```

Output:

```
Item found at index  7
```

**Binary search:**

```
nums = []
print("Enter 10 Numbers (in ascending order):")
for i in range(10):
nums.insert(i, int(input()))
print("Enter a Number to Search:")
search = int(input())
first = 0
last = 9
middle = (first+last)/2
middle = int(middle)
while first <= last:
   if nums[middle]<search:
      first = middle+1
elifnums[middle]==search:
print("The Number Found at Position:")
      print(middle+1)
      break
   else:
      last = middle-1
   middle = (first+last)/2
   middle = int(middle)
if first>last:
print("The Number is not Found in the List")
```

output:

```
Enter 10 Numbers (in ascending order):
10
20
30
40
50
60
70
80
90
100
Enter a Number to Search:50
The Number Found at Position:5
```

## Excersice-6

**1)Write a Python program to perform selection, insertion sort?**

**Selection sort:**

```
nums = []
print("Enter 10 Elements for the List: ")
for i in range(10):
nums.append(int(input()))

for i in range(9):
chk = 0
  small = nums[i]
  for j in range(i+1, 10):
    if small >nums[j]:
      small = nums[j]
chk = chk + 1
      index = j
  if chk != 0:
    temp = nums[i]
nums[i] = small
nums[index] = temp

print("\nSorted List is: ")
for i in range(10):
  print(nums[i])
```

output:

```
Enter 10 Elements for the List:10,8,15,12,16,5,9,7,4,17
Sorted List is:4,5,8,9,10,12,16,17
```

**Insertion sort:**

```
arr = []
print("Enter 10 Elements: ")
for i in range(10):
arr.append(int(input()))

for i in range(1, 10):
elem = arr[i]
 if elem<arr[i-1]:
  for j in range(i+1):
   if elem<arr[j]:
    index = j
for k in range(i, j, -1):
arr[k] = arr[k-1]
    break
 else:
  continue

arr[index] = elem

print(arr)
```

output:

```
Enter 10 Elements for the List:10,8,15,12,16,5,9,7,4,17
Sorted List is:4,5,8,9,10,12,16,17
```

## Excersice-7

**1)Write a Python program to perform merge, quick sort?**

```
listOne = []
listTwo = []

print("Enter 6 Elements for First List: ")
for i in range(6):
listOne.append(input())
print("\nEnter 6 Elements for Second List: ")
for i in range(6):
listTwo.append(input())

listThree = []
for i in range(6):
listThree.append(listOne[i])
for i in range(6):
listThree.append(listTwo[i])

print("\nThe New (Merged) List is: ")
print(listThree)
```

output:

```
Enter 6 Elements for First List:1,2,3,4,5
Enter 6 Elements for Second List:6,7,8,9,10
The New (Merged) List is:1,2,3,4,5,6,7,8,9,10
```

**Quick sort:**

```
def quicksort(arr):

   if len(arr) <= 1:

    return arr

   else:

      pivot = arr[0]

      left = [x for x in arr[1:] if x < pivot]

      right = [x for x in arr[1:] if x >= pivot]

      return quicksort(left) + [pivot] + quicksort(right)

arr = [5, 1, 8, 3, 2]
```

```
print("original array:", arr)

sorted_arr = quicksort(arr)

print("Sorted array:", sorted_arr)
```

Output:

```
original array: [5, 1, 8, 3, 2]
Sorted array: [1, 2, 3, 5, 8]
```

## Excersice-8

**8) a) Write a Python program to find first a prime numbers?**

```
num =11
# If given number is greater than 1
ifnum> 1:
   # Iterate from 2 to n / 2
   fori inrange(2, int(num/2)+1):
      # If num is divisible by any number between
      # 2 and n / 2, it is not prime
      if(num %i) ==0:
         print(num, "is not a prime number")
         break
   else:
      print(num, "is a prime number")
else:
   print(num, "is not a prime number")
```

Output

```
11 is a prime number
```

**b) write a program to check whether a given number is Armstrong or not?**

```
Num = int(input("Please Enter the Value : "))

Sum = 0
Times = 0

Temp = Num
while Temp > 0:
   Times = Times + 1
   Temp = Temp // 10

Temp = Num
for n in range(1, Temp + 1):
    Reminder = Temp % 10
    Sum = Sum + (Reminder ** Times)
    Temp //= 10

if Num == Sum:
print("%d is Armstrong Number." %Num)
else:
print("%d is Not." %Num)
```

output:

```
Please Enter the Value:153
is Armstrong Number
```

**c) Write a python program to multiply matrices?**

```
A = [[5, 4, 3],
    [2, 4, 6],
    [4, 7, 9]]
B = [[3, 2, 4],
    [4, 3, 6],
    [2, 7, 5]]
multiResult = [[0, 0, 0],
              [0, 0, 0],
              [0, 0, 0]]
for m in range(len(A)):
  for n in range(len(B[0])):
       for o in range(len(B)):
           multiResult[m][n] += A[m][o] * B[o][n] # Storing multiplication result in empty matrix
print("The multiplication result of matrix A and B is: ")
for res in multiResult:
  print(res)
```

Output:

```
The multiplication result of matrix A and B is:
[37, 43, 59]
[34, 58, 62]
[58, 92, 103]
```

**Exercise:**

1)python program that accepts a string and calculates the number of digits and letters.
2) Write a Python program that iterates the integers from 1 to 25.
3) Python program to check the validity of password input by users.
4)Python program to convert the month name to a number of days.

**Viva questions:**

1)what is range function?

2)Difference between break ,pass, continue?

**Python OOPs Concepts:**Like other general-purpose programming languages, Python is also an object-oriented language since its beginning. It allows us to develop applications using an Object-Oriented approach. In Python, we can easily create and use classes and objects.An object-oriented paradigm is to design the program using classes and objects. The object is related to real-word entities such as book, house, pencil, etc. The oops concept focuses on writing the reusable code. It is a widespread technique to solve the problem by creating objects.

Major principles of object-oriented programming system are given below.

- Class
- Object
- Method
- Inheritance
- Polymorphism
- Data Abstraction
- Encapsulation

**Class:** class is a user-defined data type that contains both the data itself and the methods that may be used to manipulate it. In a sense, classes serve as a template to create objects. They provide the characteristics and operations that the objects will employ.

Syntax

```
class ClassName:
        #statement_suite
```

**Object:**An object is a particular instance of a class with unique characteristics and functions. After a class has been established, you may make objects based on it. By using the class constructor, you may create an object of a class in Python. The object's attributes are initialised in the constructor, which is a special procedure with the name __init__.
Syntax

```
    # Declare an object of a class
    object_name = Class_Name(arguments)
```

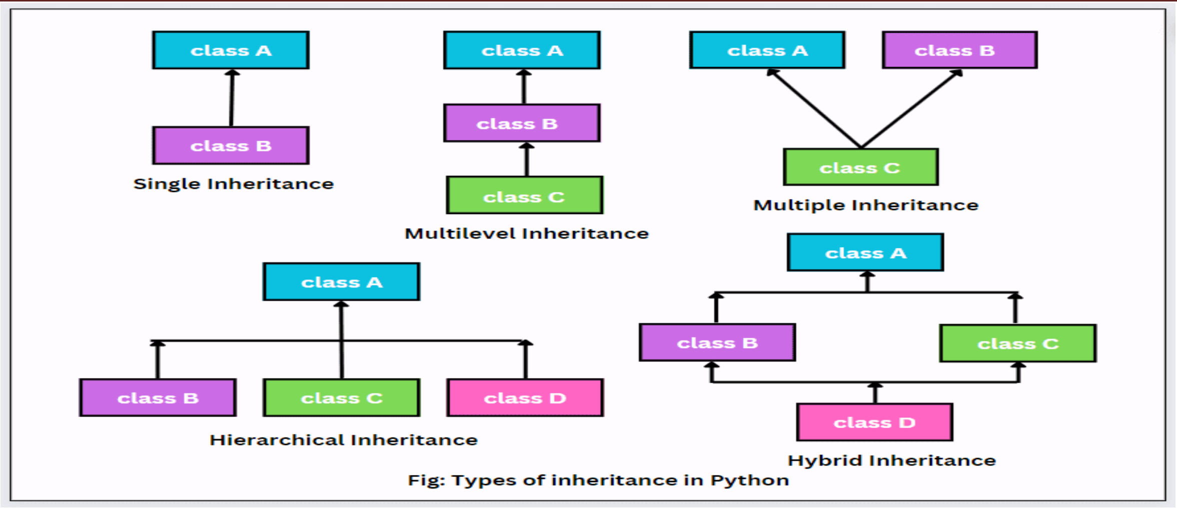
**methods**: Python methods are **functions** that are associated with an object or a class in Python programming. They are used to perform specific tasks or operations on the data stored within objects or classes.

**Python Inheritance** :Inheritance is an important aspect of the object-oriented paradigm. Inheritance provides code reusability to the program because we can use an existing class to create a new class instead of creating it from scratch.

In inheritance, the child class acquires the properties and can access all the data members and functions defined in the parent class. A child class can also provide its specific implementation to the functions of the parent class. In this section of the tutorial, we will discuss inheritance in detail. In python, a derived class can inherit base class by just mentioning the base in the bracket after the derived class name. Consider the following syntax to inherit a base class into the derived class.

Syntax: **class** derived-**class**(base **class**):

<**class**-suite>



**Polymorphism:** The word polymorphism means having many forms. In programming, polymorphism means the same function name (but different signatures) being used for different types. The key difference is the data types and number of arguments used in function.

**inbuilt polymorphism:**

```
print(len("geeks")) //for a string
print(len([10,22,30]))//for a list
output:
5
3
```

**User defined polymorphism:**

```
def add(x ,y, z=0):
  return x+y+z
print(add(2,3))
print(add(2,3,4))
Output:5
        9
```

**Polymorphism with Inheritance:**

In Python, Polymorphism lets us define methods in the child class that have the same name as the methods in the parent class. In inheritance, the child class inherits the methods from the parent class. However, it is possible to modify a method in a child class that it has inherited from the parent class.

**Data Abstraction** :Abstraction is used to hide the internal functionality of the function from the users. The users only interact with the basic implementation of the function, but inner working is hidden. User is familiar with that **"what function does"** but they don't know **"how it does."**

In simple words, we all use the smartphone and very much familiar with its functions such as camera, voice-recorder, call-dialing, etc., but we don't know how these operations are happening in the background. Let's take another example - When we use the TV remote to increase the volume. We don't know how pressing a key increases the volume of the TV. We only know to press the "+" button to increase the volume.

That is exactly the abstraction that works in the object-oriented concept.

**Syntax**

```
from abc import ABC
class ClassName(ABC):
```

**Encapsulation:**

Encapsulation is one of the fundamental concepts in object-oriented programming (OOP). It describes the idea of wrapping data and the methods that work on data within one unit. This puts restrictions on accessing variables and methods directly and can prevent the accidental modification of data. To prevent accidental change, an object's variable can only be changed by an object's method. Those types of variables are known as **private variables.**

**Python Exception:**An exception can be defined as an unusual condition in a program resulting in the interruption in the flow of the program. Whenever an exception occurs, the program stops the execution, and thus the further code is not executed. Therefore, an exception is the run-time errors that are unable to handle to Python script. An exception is a Python object that represents an error.

Python provides a way to handle the exception so that the code can be executed without any interruption. If we do not handle the exception, the interpreter doesn't execute all the code that exists after the exception.

Python has many **built-in exceptions** that enable our program to run without interruption and give the output. These exceptions are given below:

Common Exceptions

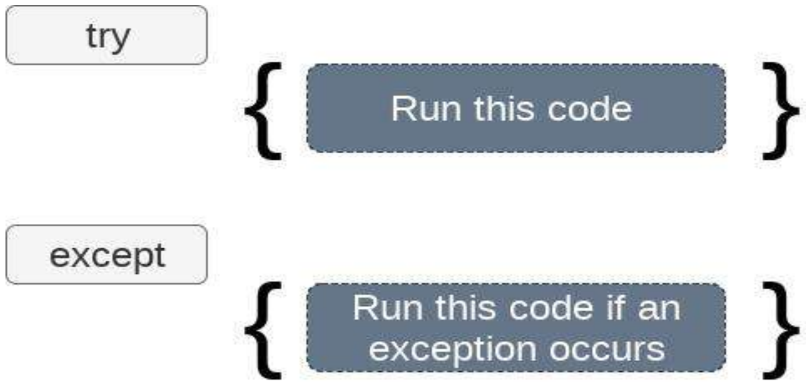
Python provides the number of built-in exceptions, but here we are describing the common standard exceptions. A list of common exceptions that can be thrown from a standard Python program is given below.

1. **ZeroDivisionError:** Occurs when a number is divided by zero.
2. **NameError:** It occurs when a name is not found. It may be local or global.
3. **IndentationError:** If incorrect indentation is given.
4. **IOError:** It occurs when Input Output operation fails.
5. **EOFError:** It occurs when the end of the file is reached, and yet operations are being performed.

Exception handling in python

## The try-expect statement

If the Python program contains suspicious code that may throw the exception, we must place that code in the **try** block. The **try** block must be followed with the **except** statement, which contains a block of code that will be executed if there is some exception in the try block.

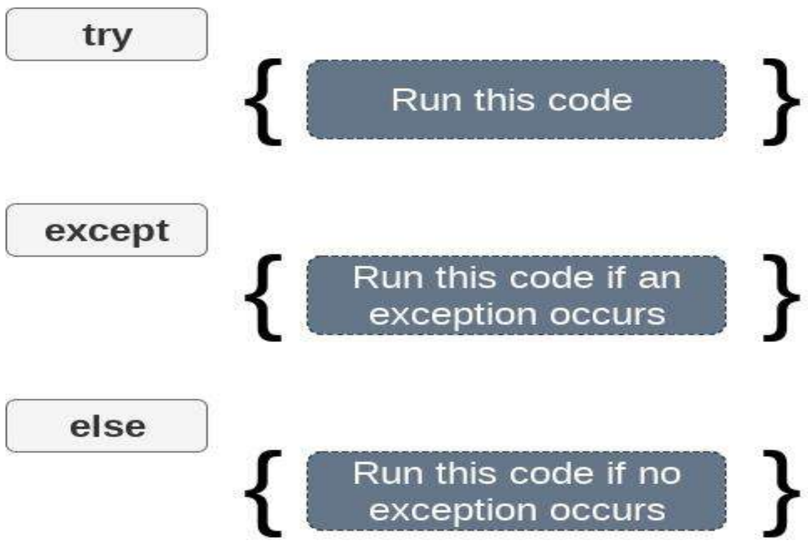


**Syntax**

```
try:
   #block of code

except Exception1:
   #block of code

except Exception2:
   #block of code

#other code
```

The syntax to use the else statement with the try-except statement is given below.

```
try:
   #block of code

except Exception1:
   #block of code

else:
   #this code executes if no except block is executed
```


try
Run this code
except
Run this code if an exception occurs
else
Run this code if no exception occurs

The Python allows us to declare the multiple exceptions with the except clause. Declaring multiple exceptions is useful in the cases where a try block throws multiple exceptions. The syntax is given below.

**Syntax**

```
1.  try:
2.      #block of code
3.
4.  except (<Exception 1>,<Exception 2>,<Exception 3>,...<Exception n>)
5.      #block of code
6.
7.  else:
8.      #block of code
```

**<u>The try...finally block:</u>**

Python provides the optional **finally** statement, which is used with the **try** statement. It is executed no matter what exception occurs and used to release the external resource. The finally block provides a guarantee of the execution.
We can use the finally block with the try block in which we can pace the necessary code, which must be executed before the try statement throws an exception.
The syntax to use the finally block is given below.
**Syntax**

```
1.  try:
2.      # block of code
3.      # this may throw an exception
4.  finally:
5.      # block of code
```

6. # this will always be executed

## Experiment-9

**a) Write a python program to print calculator using class, object?**

```
def add(x, y):
   return x + y
def sub(x, y):
   return x - y
def multiply(x, y):
   return x * y
def divide(x, y):
   return x / y
def menu():
   opt = '1. Add \n2. Sub \n3. Multiply \n4. Divide'
   print(opt)
menu()

choice = int(input('Please select one of the following : '))

num1 = int(input('Enter your First name : '))
num2 = int(input('Enter your Second name : '))
output:
'Please select one of the following :3
Enter your First name :12
Enter your Second name :4
48
```

**b) Write a python program to print mark sheet using inheritance (using all type)?**

```
class Student:
        def __init__(self):
                self.student_details = {}

        @property
        def input_student_details(self):          # input student details
                roll_number = input("Enter Roll Number : ")
                name = input("Enter Student Name : ")
                marks = float(input("Enter Marks : "))
                self.student_details[roll_number] = {'name': name, 'marks': marks}
                print("Marks Added successfully")

        def update_student_marks(self):          # update student marks
                roll_number = input("Enter Roll Number of the Student to Update Marks : ")
                if roll_number in self.student_details:
                        new_marks = float(input("Enter New Marks : "))
                        self.student_details[roll_number]['marks'] = new_marks
                        print("Marks updated successfully")
                else:
                        print("Student not found")
        def print_student_details(self):          # print student details
                roll_number = input("Enter Roll Number of the student to view details : ")
                if roll_number in self.student_details:
                        student = self.student_details[roll_number]
                        print("Roll Number :", roll_number)
                        print("Student Name :", student['name'])
                        print("Marks :", student['marks'])
                else:
                        print("Student not found")

obj = Student()

while True:
print("\n ********* Student Management System *********  ")
print("\n1. Input Student Details")
print("2. Update Student Marks")
print("3. Print Student Details")
print("4. Exit")

    choice = input("\nEnter your Choice (1 to 4): ")

    if choice == '1':
obj.input_student_details
elif choice == '2':
obj.update_student_marks()
elif choice == '3':
obj.print_student_details()
elif choice == '4':
print("Exiting the program")
        break
    else:
print("Invalid Choice. Please Try Again !!!")
```

Output

```
 ********* Student Management System *********

1. Input Student Details
2. Update Student Marks
3. Print Student Details
4. Exit

Enter your Choice (1 to 4): 1
Enter Roll Number : ST001
Enter Student Name : Siva
Enter Marks : 430
Marks Added successfully
```

## Experiment-10

**Explain the try finally clause with suitable example?**

```
try:
 a=int(input("enter the first no:- "))
 b=int(input("enter the second no:- "))
 c=a/b
 print("Div=",c)
except ZeroDivisionError as e:
print("b never be zero")
 print(e)
except ValueError as e1:
print("Value should be an integer:")
 print(e1)
except Exception:
print("exception found:")
print("Hello...........")
////////////////////////////////////////////////////

try:
 a=int(input("enter the first no:- "))
 b=int(input("enter the second no:- "))
 c=a/b
 print("Div=",c)
finally:
print("Good bye")
```

**Exercise:**

1)Write a Python program to create a class representing a bank. Include methods for managing customer accounts and transactions?

2) Write a Python program to create a class representing a shopping cart. Include methods for adding and removing items and calculating the total price.

**Viva question**

1)what is inheritance explain all types of inheritance?

2)what is difference between static and class method?

**Python File Handling:** The file-handling implementation is slightly lengthy or complicated in the other programming language, but it is easier and shorter in Python. The file handling plays an important role when the data needs to be stored permanently into the file. A file is a named location on disk to store related information. We can access the stored information (non-volatile) after the program termination.

In Python, files are treated in two modes as text or binary. The file may be in the text or binary format, and each line of a file is ended with the special character.
Hence, a file operation can be done in the following order.

- Open a file
- Read or write - Performing operation
- Close the file

| SN | Access mode | Description |
|---|---|---|
| 1 | R | It opens the file to read-only mode. The file pointer exists at the beginning. The file is by default open in this mode if no access mode is passed. |
| 2 | rb | It opens the file to read-only in binary format. The file pointer exists at the beginning of the file. |
| 3 | r+ | It opens the file to read and write both. The file pointer exists at the beginning of the file. |
| 4 | rb+ | It opens the file to read and write both in binary format. The file pointer exists at the beginning of the file. |

| | | |
|---|---|---|
| 5 | W | It opens the file to write only. It overwrites the file if previously exists or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file. |
| 6 | wb | It opens the file to write only in binary format. It overwrites the file if it exists previously or creates a new one if no file exists. The file pointer exists at the beginning of the file. |
| 7 | w+ | It opens the file to write and read both. It is different from r+ in the sense that it overwrites the previous file if one exists whereas r+ doesn't overwrite the previously written file. It creates a new file if no file exists. The file pointer exists at the beginning of the file. |
| 8 | wb+ | It opens the file to write and read both in binary format. The file pointer exists at the beginning of the file. |
| 9 | A | It opens the file in the append mode. The file pointer exists at the end of the previously written file if exists any. It creates a new file if no file exists with the same name. |
| 10 | ab | It opens the file in the append mode in binary format. The pointer exists at the end of the previously written file. It creates a new file in binary format if no file exists with the same name. |
| 11 | a+ | It opens a file to append and read both. The file pointer remains at the end of the file if a file exists. It creates a new file if no file exists with the same name. |

| | | |
|---|---|---|
| 12 | ab+ | It opens a file to append and read both in binary format. The file pointer remains<br><br>at the end of the file. |

## Experiment-11

**a) Write a Python program for command line arguments?**

```
import sys
count={}
with open(sys.argv[1],'r') as f:
for line in f:
for word in line.split():
if word not in count:
count[word]=1
else:
count[word]+=1
print(word,count[word])
f.close()
```

Output:

```
python main.py –i input.txt
words 23
lines 1
```

**b) Write a python program to print Date and Time?**

```
importdatetime

# using now() to get current time
current_time =datetime.datetime.now()

# Printing value of now.
print("Time now at greenwich meridian is:", current_time)
```

output:Time now at greenwich meridian is: 2022-06-20 16:06:13.176788

## Experiment-12

**Write a Python program to find the most frequent words in a test read form a file?**

```
count = 0;
word = "";
maxCount = 0;
```

```
words = [];

#Opens a file in read mode
file = open("data.txt", "r")

#Gets each line till end of file is reached
for line in file:
    #Splits each line into words
    string = line.lower().replace(',','').replace('.','').split(" ");
    #Adding all words generated in previous step into words
    for s in string:
        words.append(s);

#Determine the most repeated word in a file
for i in range(0, len(words)):
    count = 1;
    #Count each word in the file and store it in variable count
    for j in range(i+1, len(words)):
        if(words[i] == words[j]):
            count = count + 1;

    #If maxCount is less than count then store value of count in maxCount
    #and corresponding word to variable word
    if(count > maxCount):
        maxCount = count;
        word = words[i];

print("Most repeated word: " + word);
file.close();
```

Output: Most repeated word: computer

Data.txt file

A computer program is a collection of instructions that performs specific task when executed by a computer.

Computer requires programs to function.

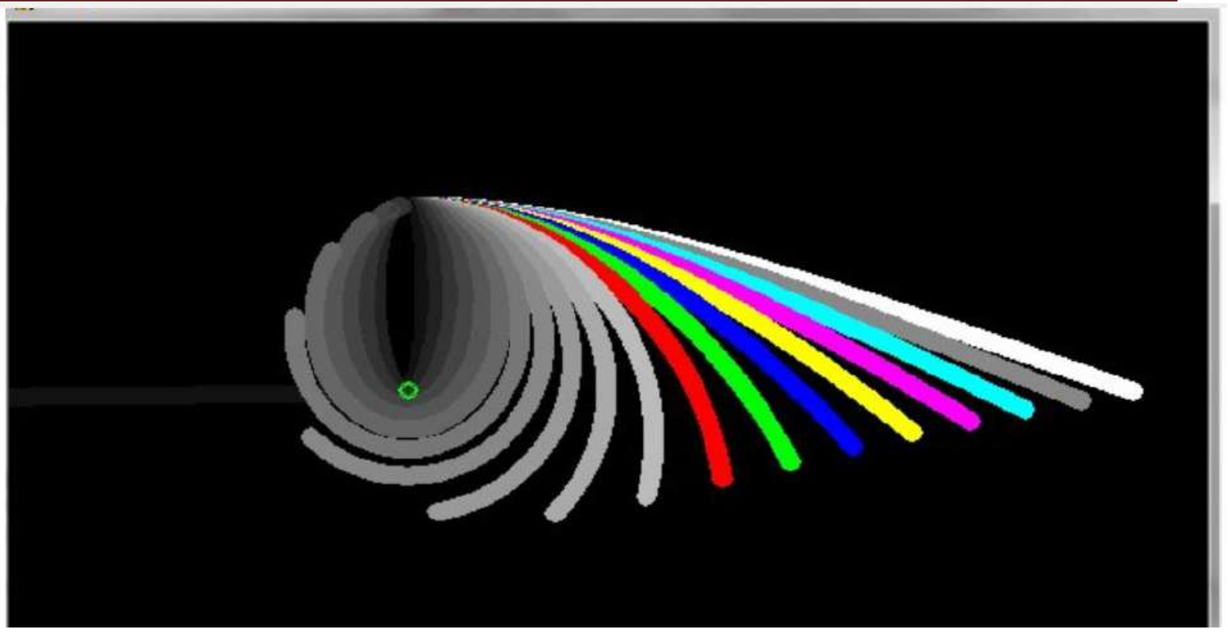
Computer program is usually written by a computer programmer in programming language.

## Experiment-13

**Write a Python program to simulate elliptical orbits in Pygame?**

```
import math
import random
import pygame
class Particle ():
def __init__ (self, x, y, colour=0x000000):
self.x = x
self.y = y
self.vx = 0
self.vy = 0
self.colour = colour
def apply_gravity (self, target):
dsqd = (self.x - target.x) ** 2 + (self.y - target.y) ** 2
#g = G*m/dsqd * normalized (self - target)
if dsqd == 0:
return
self.vx += -1 / dsqd * (self.x - target.x) / dsqd ** 0.5
self.vy += -1 / dsqd * (self.y - target.y) / dsqd ** 0.5
def update (self):
self.x += self.vx
self.y += self.vy
pygame.init()
window = pygame.display.set_mode ((600, 400))
main_surface = pygame.Surface ((600, 400))
colours = [0x000000, 0x111111, 0x222222, 0x333333, 0x444444, 0x555555, 0x666666,
0x777777, 0x888888, 0x999999, 0xaaaaaa, 0xbbbbbb] + [0xFF0000, 0x00FF00, 0x0000FF,
0xFFFF00, 0xFF00FF, 0x00FFFF, 0x888888, 0xFFFFFF, 0x808000, 0x008080, 0x800080,
0x800000]
#colours = [0xFF0000, 0x00FF00, 0x0000FF, 0xFFFF00, 0xFF00FF, 0x00FFFF,
0x888888,0xFFFFFF, 0x808000, 0x008080, 0x800080, 0x800000]
particles = [Particle (200, 100, colours [i]) for i in range (20)]
earth = Particle (200, 200)
for i, p in enumerate (particles):
p.vx = i / 100
while (True):
# main_surface.fill(0x000000)
pygame.draw.circle (main_surface, 0x00FF00, (earth.x, earth.y), 5, 2)
for p in particles:
p.apply_gravity (earth)
p.update ()
pygame.draw.circle (main_surface, p.colour, (int (p.x), int (p.y)), 5, 2)
window.blit(main_surface, (0, 0))
pygame.display.flip()
```

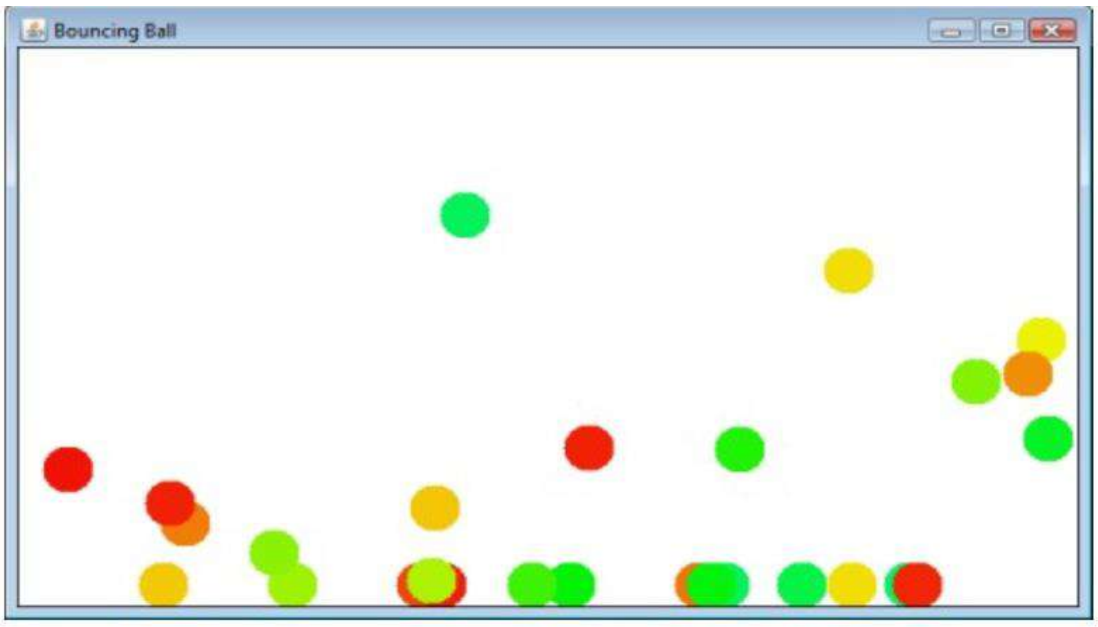
Output:



## Experiment-14

**Write a Python program to bouncing ball in Pygame?**

```
import sys
import pygame
pygame.init()
size = width, height = 320, 240
speed = [2, 2]
black = 0, 0, 0
screen = pygame.display.set_mode(size)
ball = pygame.image.load('C:\\Users\\admin\\Desktop//ball.jpg')
ballrect = ball.get_rect()
while 1:
for event in pygame.event.get():
if event.type == pygame.QUIT: sys.exit()
ballrect = ballrect.move(speed)
if ballrect.left< 0 or ballrect.right> width:
speed[0] = -speed[0]
if ballrect.top< 0 or ballrect.bottom> height:
speed[1] = -speed[1]
screen.fill(black)
screen.blit(ball, ballrect)
pygame.display.flip()
```



**Excersice:**

1)write a python program print marksheet using file handling concept?

2)write a python program multi-threading?

**Viva question**

1)what is module and package?

2)what is multi-processing and multi-threading?

## VISION OF THE DEPARTMENT

To be a centre of excellence for providing quality technical education to develop future leaders with the aspects of research &computing, Software product development and entrepreneurship.

## MISSION OF THE DEPARTMENT

**M1:** To offer academic programme with state of art curriculum having flexibility for accommodating the latest developments in the areas of Computer science engineering

**M2:** To conduct research and development activities in contemporary and emerging areas of computer science & engineering.

**M3:** To inculcate moral values & entrepreneurial skills to produce professionals capable of providing socially relevant and sustainable solutions.

# Python (CS-506)