Python

1.Write a Python program to Show Multilevel Inheritance.

Algorithm

1. **Start**
2. **Define class Mca**
   - **Method:** Define `course1()` that prints "PG Course: MCA"
3. **Define class Bca that inherits from Mca**
   - **Method:** Define `course2()` that prints "UG Course: BCA"
4. **Define class Bsc that inherits from Bca**
   - **Method:** Define `course3()` that prints "Second UG Course: BSc"
5. **Create an instance of class Bsc**
   - Store it in variable `c`
6. **Call the method `course1()` on object `c`**
7. **Call the method `course2()` on object `c`**
8. **Call the method `course3()` on object `c`**
9. **Stop**

Flowchart Structure

Here’s how the flowchart flows from one shape to another:

1. **Oval** (Start)
   - Label: "Start"
2. **Rectangle** (Process)
   - Label: "Define class Mca with course1()"
3. **Rectangle** (Process)
   - Label: "Define class Bca inheriting from Mca with course2()"
4. **Rectangle** (Process)
   - Label: "Define class Bsc inheriting from Bca with course3()"
5. **Rectangle** (Process)
   - Label: "Create object 'c' of class Bsc"
6. **Rectangle** (Process)
   - Label: "Call course1() method on object 'c'"
7. **Rectangle** (Process)
   - Label: "Call course2() method on object 'c'"
8. **Rectangle** (Process)
   - Label: "Call course3() method on object 'c'"

9. **Oval** (End)
    - Label: "Stop"

2.Write a Python program to display Calendar by providing the Year entered by user.

Algorithm

1. **Start**
2. **Import the calendar module**
3. **Prompt the user to enter a year**
    - Read the input and store it in variable `y`
4. **Prompt the user to enter a month**
    - Read the input and store it in variable `m`
5. **Display the calendar for the specified month and year**
    - Use `calendar.month(y, m)` to print the calendar
6. **Stop**

Flowchart Structure

Here’s how the flowchart flows from one shape to another:

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Import calendar module"
3. **Rectangle** (Input)
    - Label: "Prompt user to enter year (y)"
4. **Rectangle** (Input)
    - Label: "Prompt user to enter month (m)"
5. **Rectangle** (Process)
    - Label: "Display calendar for month m of year y"
6. **Oval** (End)
    - Label: "Stop"

3.. Write a Program in Python to Show Method Overiding.

Algorithm

1. **Start**
2. **Define class Demo1**
    - **Method:** Define `Display()` that prints "I am from Parent Class"
3. **Define class Demo2 that inherits from Demo1**
    - **Method:** Override `Display()` to print "I am from Child Class"

4. **Create an instance of class Demo2**
    - Store it in variable `a`
5. **Call the `Display()` method on object `a`**
6. **Stop**

Flowchart Structure

Here’s how the flowchart flows from one shape to another:

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Define class Demo1 with Display()"
3. **Rectangle** (Process)
    - Label: "Define class Demo2 inheriting from Demo1 with overridden Display()"
4. **Rectangle** (Process)
    - Label: "Create object 'a' of class Demo2"
5. **Rectangle** (Process)
    - Label: "Call Display() method on object 'a'"
6. **Oval** (End)
    - Label: "Stop"

4.Write a Python Program that implement Thread.

Algorithm

1. **Start**
2. **Import `sleep` from `time` module**
3. **Import `Thread` from `threading` module**
4. **Define class `Hello` that inherits from `Thread`**
    - **Method:** Define `run()` to print "Hello" 100 times, with a 1-second delay between prints
5. **Define class `Hi` that inherits from `Thread`**
    - **Method:** Define `run()` to print "Hi" 100 times, with a 1-second delay between prints
6. **Create an instance of class `Hello`**
    - Store it in variable `t1`
7. **Create an instance of class `Hi`**
    - Store it in variable `t2`
8. **Start thread `t1`**
9. **Sleep for 0.2 seconds**
10. **Start thread `t2`**

11. **Wait for thread `t1` to finish**
12. **Wait for thread `t2` to finish**
13. **Print "Good"**
14. **Stop**

Flowchart Structure

Here’s how the flowchart flows from one shape to another:

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Import sleep from time module"
3. **Rectangle** (Process)
    - Label: "Import Thread from threading module"
4. **Rectangle** (Process)
    - Label: "Define class Hello with run() method (prints 'Hello')"
5. **Rectangle** (Process)
    - Label: "Define class Hi with run() method (prints 'Hi')"
6. **Rectangle** (Process)
    - Label: "Create object 't1' of class Hello"
7. **Rectangle** (Process)
    - Label: "Create object 't2' of class Hi"
8. **Rectangle** (Process)
    - Label: "Start thread 't1'"
9. **Rectangle** (Process)
    - Label: "Sleep for 0.2 seconds"
10. **Rectangle** (Process)
    - Label: "Start thread 't2'"
11. **Rectangle** (Process)
    - Label: "Wait for thread 't1' to finish"
12. **Rectangle** (Process)
    - Label: "Wait for thread 't2' to finish"
13. **Rectangle** (Process)
    - Label: "Print 'Good'"
14. **Oval** (End)
    - Label: "Stop"

5.Write a Python Program to draw the “filled arc” using TKinter Module

Algorithm

1. **Start**
2. **Import the `tkinter` module**
3. **Create the main window (`top`) using `tkinter.Tk()`**
4. **Create a canvas (`C`) with a green background, height 250, and width 300**
5. **Define the coordinates for the arc (`coord = 10, 50, 240, 210`)**
6. **Create an arc on the canvas with specified coordinates, start angle 0, extent 150, and fill color red**
7. **Pack the canvas into the main window**
8. **Run the main event loop with `top.mainloop()`**
9. **Stop**

Flowchart Structure

Here's how the flowchart flows from one shape to another:

1. **Oval** (Start)
   - Label: "Start"
2. **Rectangle** (Process)
   - Label: "Import tkinter module"
3. **Rectangle** (Process)
   - Label: "Create main window (top) using tkinter.Tk()"
4. **Rectangle** (Process)
   - Label: "Create canvas (C) with green background, height 250, width 300"
5. **Rectangle** (Process)
   - Label: "Define coordinates for the arc (coord = 10, 50, 240, 210)"
6. **Rectangle** (Process)
   - Label: "Create arc on canvas with specified coordinates, start=0, extent=150, fill='red'"
7. **Rectangle** (Process)
   - Label: "Pack the canvas into the main window"
8. **Rectangle** (Process)
   - Label: "Run main event loop (top.mainloop())"
9. **Oval** (End)
   - Label: "Stop"

6.Write Python program to create Menus and Submenus using Tkinter

Algorithm

1. **Start**
2. **Import all components from the `tkinter` module**
3. **Define a function `donothing()`**

- **Inside this function**:
  - Create a new window (`filewin`) using `Toplevel(root)`
  - Create a button in `filewin` with text "Do nothing button" and pack it

4. **Create the main window (`root`) using** `Tk()`
5. **Create a menubar using** `Menu(root)`
6. **Create a file menu (`filemenu`)**
   - Add commands for "New", "Open", "Save", "Save as...", "Close", and "Exit"
   - Assign `donothing` function to each command except "Exit"
   - Add a separator in the file menu
7. **Create an edit menu (`editmenu`)**
   - Add commands for "Undo", "Cut", "Copy", "Paste", "Delete", and "Select All"
   - Assign `donothing` function to each command
   - Add a separator in the edit menu
8. **Create a help menu (`helpmenu`)**
   - Add commands for "Help Index" and "About..."
   - Assign `donothing` function to each command
9. **Add the file, edit, and help menus to the menubar**
10. **Configure the main window to use the menubar**
11. **Run the main event loop with** `root.mainloop()`
12. **Stop**

Flowchart Structure

Here’s how the flowchart flows from one shape to another:

1. **Oval** (Start)
   - Label: "Start"
2. **Rectangle** (Process)
   - Label: "Import all components from tkinter module"
3. **Rectangle** (Process)
   - Label: "Define function donothing()"
   - **Sub-Process:** "Create Toplevel window (filewin) and button"
4. **Rectangle** (Process)
   - Label: "Create main window (root) using Tk()"
5. **Rectangle** (Process)
   - Label: "Create menubar using Menu(root)"
6. **Rectangle** (Process)
   - Label: "Create file menu (filemenu) and add commands"
7. **Rectangle** (Process)
   - Label: "Create edit menu (editmenu) and add commands"
8. **Rectangle** (Process)

- Label: "Create help menu (helpmenu) and add commands"

9. **Rectangle** (Process)
    - Label: "Add menus to menubar"
10. **Rectangle** (Process)
    - Label: "Configure main window to use menubar"
11. **Rectangle** (Process)
    - Label: "Run main event loop (root.mainloop())"
12. **Oval** (End)
    - Label: "Stop"

7.Write a Python Program to Show the concept of Exception handling.

Algorithm

1. **Start**
2. **Initialize variables:**
    - Set `a = 10`
    - Set `b = 5`
3. **Try to execute the following block:**
    - Calculate `d = a / b`
    - Print the value of `d`
4. **Catch the `ZeroDivisionError` exception:**
    - If an exception occurs, print "Division by zero not allowed"
5. **Print "Rest of the code"**
6. **Stop**

Flowchart Structure

Here's how the flowchart flows from one shape to another:

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Initialize a = 10 and b = 5"
3. **Diamond** (Decision)
    - Label: "Try to calculate d = a / b"
4. **Rectangle** (Process)
    - Label: "Print value of d"
    - **From Diamond if successful**
5. **Diamond** (Decision)
    - Label: "Is there a ZeroDivisionError?"
    - **From Diamond if an exception occurs**

6. **Rectangle** (Process)
    - Label: "Print 'Division by zero not allowed'"
    - **From Diamond if exception occurs**
7. **Rectangle** (Process)
    - Label: "Print 'Rest of the code'"
8. **Oval** (End)
    - Label: "Stop"

8.Write a Program in Python that show the use of following Built-In Functions.
i)append() ii) reverse() iii) index iv) getattr() v) setattr()

Algorithm for All Code Snippets

i) `append()`

1. **Start**
2. **Create a list of fruits:**
    - `fruits = ['apple', 'banana', 'cherry']`
3. **Append "orange" to the list:**
    - `fruits.append("orange")`
4. **Print the updated list of fruits**
5. **Stop**

ii) `reverse()`

1. **Start**
2. **Create a list of fruits:**
    - `fruits = ['apple', 'banana', 'cherry']`
3. **Reverse the list of fruits:**
    - `fruits.reverse()`
4. **Stop**

iii) `index`

1. **Start**
2. **Create a list of fruits:**
    - `fruits = ['apple', 'banana', 'cherry']`
3. **Find the index of "cherry":**
    - `x = fruits.index("cherry")`
4. **Print the index of "cherry"**
5. **Stop**

iv) `getattr()`

1. **Start**
2. **Define class `Person` with attributes:**
   - `name = "John"`
   - `age = 36`
   - `country = "Norway"`
3. **Get the value of the attribute `age`:**
   - `x = getattr(Person, 'age')`
4. **Print the value of `x`**
5. **Stop**

v) `setattr()`

1. **Start**
2. **Define class `Person` with attributes:**
   - `name = "John"`
   - `age = 36`
   - `country = "Norway"`
3. **Set the attribute `age` to 40:**
   - `setattr(Person, 'age', 40)`
4. **Stop**

Flowchart Structure for Each Operation

i) `append()`

1. **Oval** (Start)
   - Label: "Start"
2. **Rectangle** (Process)
   - Label: "Create fruits = ['apple', 'banana', 'cherry']"
3. **Rectangle** (Process)
   - Label: "Append 'orange' to fruits"
4. **Rectangle** (Process)
   - Label: "Print fruits"
5. **Oval** (End)
   - Label: "Stop"

---

ii) `reverse()`

1. **Oval** (Start)
   - Label: "Start"

2. **Rectangle** (Process)
    - Label: "Create fruits = ['apple', 'banana', 'cherry']"
3. **Rectangle** (Process)
    - Label: "Reverse the list"
4. **Rectangle** (Process)
    - Label: "Print reversed fruits"
5. **Oval** (End)
    - Label: "Stop"

---

iii) `index`

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Create fruits = ['apple', 'banana', 'cherry']"
3. **Rectangle** (Process)
    - Label: "Find index of 'cherry'"
4. **Rectangle** (Process)
    - Label: "Print index of 'cherry'"
5. **Oval** (End)
    - Label: "Stop"

---

iv) `getattr()`

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Define class Person with attributes"
3. **Rectangle** (Process)
    - Label: "Get age using getattr"
4. **Rectangle** (Process)
    - Label: "Print age"
5. **Oval** (End)
    - Label: "Stop"

---

v) `setattr()`

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Define class Person with attributes"
3. **Rectangle** (Process)
    - Label: "Set age to 40 using setattr"
4. **Rectangle** (Process)
    - Label: "Print updated age"
5. **Oval** (End)
    - Label: "Stop"

9.Write a Python program that Show the OS name,Version of System,path and Current working directory.

Algorithm for the Code Snippets

Step 1: Import `sys` and Display `sys.path`

1. **Start**
2. **Import the `sys` module**
3. **Retrieve and print `sys.path`**
4. **Stop**

Step 2: Display Python Version

1. **Start**
2. **Import the `sys` module** (already imported)
3. **Retrieve and print `sys.version`**
4. **Stop**

Step 3: Get Current Working Directory

1. **Start**
2. **Import the `os` module**
3. **Retrieve and print the current working directory using `os.getcwd()`**
4. **Stop**

Step 4: Get OS Name

1. **Start**
2. **Import the `os` module** (already imported)
3. **Print the name of the operating system using `os.name`**
4. **Stop**

Flowchart Structure with Shapes

Here's the flowchart with specific shapes mentioned for each step:

1. **Oval** (Start)
    - Label: "Start"
2. **Rectangle** (Process)
    - Label: "Import sys module"
3. **Rectangle** (Process)
    - Label: "Retrieve sys.path"
4. **Rectangle** (Process)
    - Label: "Print sys.path"
5. **Rectangle** (Process)
    - Label: "Retrieve sys.version"
6. **Rectangle** (Process)
    - Label: "Print sys.version"
7. **Rectangle** (Process)
    - Label: "Import os module"
8. **Rectangle** (Process)
    - Label: "Retrieve current working directory using os.getcwd()"
9. **Rectangle** (Process)
    - Label: "Print current working directory"
10. **Rectangle** (Process)
    - Label: "Import os module (again)"
11. **Rectangle** (Process)
    - Label: "Print OS name using os.name"
12. **Oval** (End)
    - Label: "Stop"

10.Write a Python program to draw Colorful Star using Turtle module.

Algorithm for the Turtle Graphics Code

1. **Start**
2. **Import the `turtle` module**
3. **Create a turtle object named `star`**
4. **For `i` in range from 0 to 99:**
    - a. Move the turtle forward by 100 units: `star.forward(100)`
    - b. Turn the turtle right by 144 degrees: `star.right(144)`
5. **Finish drawing and display the window with `turtle.done()`**
6. **Stop**

Flowchart Structure with Shapes

Here's the flowchart with specific shapes mentioned for each step:

1. **Oval** (Start)
   - Label: "Start"
2. **Rectangle** (Process)
   - Label: "Import turtle module"
3. **Rectangle** (Process)
   - Label: "Create turtle object named star"
4. **Diamond** (Decision)
   - Label: "For i in range(100)"
5. **Rectangle** (Process)
   - Label: "Move star forward by 100 units"
6. **Rectangle** (Process)
   - Label: "Turn star right by 144 degrees"
7. **Arrow** (Loop back to the diamond)
8. **Rectangle** (Process)
   - Label: "Finish drawing with turtle.done()"
9. **Oval** (End)

```
- Label: "Stop"
```

11.Write a Python Program that Show HostName and IP Address using Socket module.

Algorithm for the Turtle Graphics Code

1. **Start**
2. **Import the `turtle` module**
3. **Create a turtle object named `star`**
4. **For `i` in range from 0 to 99:**
   - a. Move the turtle forward by 100 units: `star.forward(100)`
   - b. Turn the turtle right by 144 degrees: `star.right(144)`
5. **Finish drawing and display the window with `turtle.done()`**
6. **Stop**

Flowchart Structure with Shapes

Here's the flowchart with specific shapes mentioned for each step:

1. **Oval** (Start)
   - Label: "Start"

2. **Rectangle** (Process)
    - Label: "Import socket module"
3. **Rectangle** (Process)
    - Label: "Define function print_machine_info()"
4. **Rectangle** (Process)
    - Label: "Retrieve host name using socket.gethostname()"
5. **Rectangle** (Process)
    - Label: "Retrieve IP address using socket.gethostbyname(host_name)"
6. **Rectangle** (Process)
    - Label: "Print host name"
7. **Rectangle** (Process)
    - Label: "Print IP address"
8. **Diamond** (Decision)
    - Label: "Is **name** == '**main**'?"
9. **Arrow** (If True) → Rectangle
    - Label: "Call print_machine_info()"
10. **Oval** (End)
    - Label: "Stop"