PRACTICAL NO.: 1

AIM:Create a program that asks the user to enter their name and their age. Print out a message addresses to them that tells them the year that they will turn 100 years old

Variables are nothing-but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable.

The equal sign (=) is used to assign values to variables. The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the value stored in the variable.

Practical:

```
import datetime
def main():
    name = input("Enter your name: ")
    age = int(input("Enter your age: "))
    current_year = datetime.datetime.now().year
    year_turn_100 = current_year + (100 - age)
    print("Hello, {name}! You will turn 100 years old
          in the year {year_turn_100}.")
if __name__ == "__main__":
    main()
```

OUTPUT

Enter your name : John
Enter your age : 30
Hello, John! You will turn 100 years old in the year 2094.

Post Practical Questions:

1. Which one of the following is correct way of declaring and initializing a variable, x with value 5?

(a) int x:5
(b) int x=5
(c) x=5
(d) declare x=5

2. All keyword in python are in

(a) Lowercase
(b) Uppercase
(c) Both Lowercase and Uppercase
(d) None of the above

3. Which of the following cannot be a variable?

(a) __init__
(b) it
(c) in
(d) on

4. Which of the following character is used to give single-line comments in Python?

(a) //
(b) !
(c) #
(d) /*

5. What will be the output of the following code snippet?

```
print(type(5 / 2))
print(type(5 // 2))
```

(a) float and int
(b) int and float
(c) float and float
(d) int and int

Conclusion:

From this practical, we learned the basic syntax of Python and the datetime library of it.

Marks out of 10	
Signature with Date of Completion	

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL NO.: 2

AIM: Develop a python program to make a simple calculator using a conditional loop.

Decision making is required when we want to execute a code only if a certain condition is satisfied.
Syntax of if...else:
if test expression:
Body of if
else:
Body of else

While Loop:

In python, while loop is used to execute a block of statements repeatedly until a given a condition is satisfied. And when the condition becomes false, the line immediately after the loop in program is executed.

Syntax:
while expression: statement(s)

Practical

```
def calculator();
    print("Simple calculator")
    print("Operations:")
    print("1. Addition (+)")
    print("2. Subraction (-)")
    print("3. Multiplication (*)")
    print("4. Division (/)")
    print("5. Exit")
    while True:
        choice = input("Enter operation number (1-5): ")
        if choice == '5':
            print("Exiting the calculator")
            break
        if choice not in ['1', '2', '3', '4']:
            print("Invalid choice. Please enter a number from 1 to 5.")
            continue
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))
```

```
if choice == '1':
    result = num1 + num2
    print("Result: ", result)
elif choice == '2':
```

OUTPUT

```
Simple Calculator
Operations:
1. Addition(+)
2. Subtraction(-)
3. Multiplication(*)
4. Division(/)
5. Exit
Enter operation number(1-5): 1
Enter first number: 10
Enter second number: 5
Result: 15.0
```

Post Practical Questions:

(1) Which of the following is not used as loop in Python?

(a) for loop
(b) while loop
(c) do-while loop
(d) None of the above

(2) Which of the following is False regarding loops in Python?

(a) Loops are used to perform certain tasks repeatedly.

(b) While loop is used when multiple statements are to executed repeatedly until the given condition becomes False

(c) While loop is used when multiple statements are to executed repeatedly until the given condition becomes True.

(d) for loop can be used to iterate through the elements of lists.

(3) What keyword would you use to add an alternative condition to an if statement?

(a) else if
(b) elseif
(c) elif
(d) None of the above

(4) In a Python program, a control structure:

(a) Defines program-specific data structures

(b) Directs the order of execution of the statements in the program

(c) Dictates what happens before the program starts and after it terminates

(d) None of the above

Conclusion:

From this practical, we learned about the if else statement ladder in python and its implementation.

Marks out of 10	10
Signature with Date of Completion	Nidhi 23/3/24

```
    result = num1 - num2
    print("Result: ", result)
elif choice == '3':
    result = num1 * num2
    print("Result: ", result)
elif choice == '4':
    if num2 == 0:
        print("cannot divide by zero.")
    else:
        result = num1/num2
        print("Result:", result)

Calculator()
```

Practical

```
number = int(input("Enter a number:"))
if number % 2 ==0:
    print(number,"is even.")
    if number % 4 ==0:
        print(number,"is also a multiple of 4:")

else:
    print(number,"is odd!")
```

OUTPUT

```
Enter a number: 6
6 is even.
Enter a number: 9
9 is odd.
Enter a number: 16      16 is even
                        16 is also a multiple of 4.
```

Post Practical Questions:

(1) Does python have switch case statement? -

(a) True

(b) False

(c) Python has switch statement but we cannot use it.

(d) None of the given

(2) If the else statement is used with a while loop, the else statement is executed when the condition becomes _____.

(a) True
(b) False
(c) Infinite
(c) Null

(3) The _____ statement is a null operation.

(a) break
(b) exit
(c) return
(d) pass

(4) In programming, the concept of decision making or selection is implemented with the help of _____ statement

(a) while loop
(b) for loop
(c) if..else
(d).None of the above

Conclusion:

From this practical, we learned about the decision making statements in Python and blocks of if and else statements and its implementation.

Marks out of 10	10
Signature with Date of Completion	Nidhi 23/3/24

PRACTICAL NO.: 4

Aim: Take a list, say for example this one: a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89], and write a program that prints out all the elements on the list that are less than 5.

Self: Instead of printing the elements one by one, make a new list that has all the elements less than

5 from this list in it and print out this new list

List:

Lists are used to store multiple items in a single variable.

Lists are one of 4 built-in data types in Python used to store collections of data, the other 3 are Tuple, Set, and Dictionary, all with different qualities and usage.

For loop:

A for loop is used for iterating over a sequence (that is a list, a tuple, a dictionary, a set or a string).

This is less like the for keyword in other programming language, and works more like an iterator method as found in other object-oriented programming languages.

With the for loop, we can execute a set of statements, once for each item in a list, tuple, set etc.

Practical

```
a = [1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]

for element in a:
    if element < 5:
        print(element).
```

OUTPUT

1
1
2
3

Post Practical Questions:

Post Practical Questions:

(1) for loop in python are worked on

(a) Range
(b) Iteration
(c) Both A and B
(d) None

(2) Which of the following is the use of id() function in python?

(a) Every object doesn't have a unique id
(b) Id returns the identity of the objec
(c) All of the mentioned
(d) None of the mentioned

(3) What will be the output of the following Python code?

```
x = 'abcd'
for i in x:
  print(i.upper())
```

(a) a B C D
(b) a b c d
(c) error
(d) A B C D

(4) To add a new element to a list we use which Python command?

(a) list1.addEnd(5)

(b) list1.addLast(5)

(c) list1.append(5)

(d) list1.add(5)

Conclusion:

From this practical, we learned how to use the for loop in python and its implementation

Marks out of 10	10
Signature with Date of Completion	Nidhi 28/3/24

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

The aim of creating a two-player Rock-Paper-Scissors game is to provide a simple and interactive program where two players can compete against each other by choosing one of three options: Rock, Paper, or Scissors.

Steps :

1. Take the player choice using input() function.
2. Comparison of choices using conditional structure and print the winner
3. Use control structure to play again after each round

Practical

```
def play_game(player1, player2):
    if player1 == player2:
        return "It's a tie!"
    elif (player1 == 'rock' and player2 == 'scissor') or\
         (player1 == 'scissors' and player2 == 'paper') or\
         (player1 == 'paper' and player2 == 'rock'):
        return "player 1 wins!"
    else:
        return "player 2 wins!"

def main():
    while True:
        print("Welcome to the Rock-Paper-Scissors game!")
        print("Player 1, choose rock, paper, or scissors: ")
        player1_choice = input().lower()
        while player1_choice not in ['rock', 'paper', 'scissors']:
            print("Invalid choice. Please choose rock, paper,
                  or scissors: ")
        player1_choice = input().lower()
```

```
Print("Player 2, choose rock, paper, or scissors: ")
Player2_choice = input().lower()
while Player2_choice not in ['rock', 'paper', 'scissors']:
    Print("Invalid choice. Please choose rock, paper, or scissors!")
    Player2_choice = input().lower()
Print (play_game (player1_choice, player2_choice))
play_again = Input("Do you want to play again? (yes/no): ").
                                                    lower()
  if play_again != 'yes':
      print ("Thanks for playing!")
      break
if __name__ == "__main__":
    main()
```

OUTPUT

```
Welcome to the Rock-Paper-Scissors game!
Player 1, enter your choice (rock, paper, or scissors):
  Scissors.
Player 1 wins!
Player 2, enter your choice (rock, paper, or scissors):
  rock.
Player 2 wins!
Do you want to play again? (yes/no): no
```

Post Practical Questions:

1. **Which function is used to get the player choices in the game?**

(a) input()
(b) takeinput()
(c) choice()
(d) None of the mentioned

2. **What rule determines the winner when one player chooses Rock and the other chooses Scissors?**

(a) Scissors beats Rock
(b) Rock beats Scissors
(c) Both
(d) None of the mentioned

3. **Which of the following control structure will be used to play the game again ?**

(a) while
(b) for
(c) nested if else
(d) None of the mentioned

Conclusion:

From, this practical we got to learn about building a simple stone, paper, scissors game in python.

Marks out of 10	10
Signature with Date of Completion	Abdl 23/3/24

PRACTICAL NO.: 6

Aim: Ask the user for a string and print out whether this string is a palindrome or not. (Using string reversal and for loops).

Strings in python are surrounded by either single quotation marks, or double quotation marks. 'hello' is the same as "hello".

A palindrome is a string which is same read forward or backwards.

For example: "dad" is the same in forward or reverse direction. Another example is "aibohphobia" which literally means, an irritable fear of palindromes.

The program takes a string and checks if a string is a palindrome or not.

Practical

```
def is_palindrome(string):
reversed_string = ""
for char in string:
    reversed_string = char + reversed_string
if string == reversed_string:
    return True.
else:
    return False
user_input = input("Enter a string:")
user_input = user_input.replace(" ", "").lower()
if is_palindrome(user_input):
    print("The string is a palindrome.")
else:
    print("The string is not a palindrome.")
```

OUTPUT

Enter a string : Radar
The string is a palindrome.

Post Practical Questions:

1. What will be the output of the following Python statement?

>>>"a"+"bc"

(a) a
(b) bc
(c) bca
(d) abc

2. What will be the output of the following Python statement?

>>>"abcd"[2:]

(a) a
(b) ab
(c) cd
(d) dc

3. What arithmetic operators cannot be used with strings?

(a) +
(b) *
(c) –
(d) All of the mentioned

4. What will be the output of the following Python code?

>>> max("what are you")

(a) error (b) u

(c) t (d) y

Conclusion:

From this practical we got to learn about reversing a string in python and checking if a string is palindrome or not.

Marks out of 10	
Signature with Date of Completion	

Practical

```
def remove_duplicate(input_list):
    unique_elements = set(input_list)
    return list(unique_elements)
input_list = [1, 2, 3, 4, 3, 2, 1]
result = remove_duplicate(input_list)
print(results)
```

OUTPUT

[1, 2, 3, 4]

Post Practical Questions:

1. What is returned by the following function?

def len_of_list():

```
alist = [3, 67, "cat", 3.14, False]
return len(alist)
```

(a) 4 (b) 5

(c) False (d) 3.14

2. What is the output of the following code

```
my_list = ["Hello", "Python"]
print("-".join(my_list))
```

(a) HelloPython- (b) Hello-Python

(c) -HelloPython (d) None of the mentioned

3. In Python, list is mutable

(a) Yes (b) No

(c) May be (d) None

4. Which one of the following is the use of function in python?

a) Functions don't provide better modularity for your application

b) you can't also create your own functions

c) Functions are reusable pieces of programs

d) All of the mentioned

Conclusion:

Through this program we get to learn about how does lists work in jetpack compose.

Marks out of 10	9
Signature with Date of Completion	23/3/24

Practical

```
def reverse_words(sentence):
    words = sentence.split()
    reversed_sentence = " ".join(reversed(words))
    return reversed_sentence

def main():
    user_input = input("Enter a long string
                        containing multiple
                        words: ")
    reversed_string = reverse_words(user_input)
    print("Reversed string: ", reversed_string)

if __name__ == "__main__":
    main()
```

OUTPUT

Enter a long string containing multiple words : My name is Michele.

Reversed string : Michele is name My.

Post Practical Questions:

1. **Strings are immutable in Python, which means a string cannot be modified.**

a)true b)false

2. **Which of the following will give "Simon" as output?**

If str1="John,Simon,Aryan"

a) print(str1[-7:-12])
b) print(str1[-11:-7])
c) print(str1[-11:-6])
d) print(str1[-7:-11])

3. **What is the output when following code is executed ?**

print r"\nhello"

a) a new line and hello
b) \nhello
c) the letter r and then hello
d) Error

4. **What is the output of "hello +1+2+3 ?**

a) hello123
b) hello
c) Error
d) hello6

Conclusion:

From this program we got to learn about string and mututing strings in python.

Marks out of 10	9
Signature with Date of Completion	Nidhi 23/3/24

PRACTICAL NO.: 9

Aim: Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

An object is simply a collection of data (variables) and methods (functions) that act on those data. Similarly, a class is a blueprint for that object.

We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

As many houses can be made from a house's blueprint, we can create many objects from a class. An object is also called an instance of a class and the process of creating this object is called instantiation.

Practical

```
class Circle :
  def __init__(self, radius):
    self.radius = radius
  def area(self):
    return 3.14159 * self.radius ** 2
  def perimeter(self):
    return 2 * 3.14159 * self.radius
radius = 5
circle = Circle(radius).
```

```
print("Area of the circle:", circle.area())
print("Perimeters of the circle:", circle.perimeter())
```

OUTPUT

Area of the circle : 78.53975
Perimeter of the circle : 31.4159

Post Practical Questions:

1. Explain how to define class and object in python?

In python, a class is a blueprint for creating objects, defining their properties and behaviours. An object is an instance of a class, encapsulating data and functionality.

2. Explain Constructor in python.

In python, a constructor is a special method (init) within a class that initializes new objects. It is automatically called when an object is created, allowing for initialization of instance variables.

Conclusion:

From this practical we got to learn about class and objects in python.

Marks out of 10	9
Signature with Date of Completion	Nidhi 23/3/24

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL NO.: 10

Aim: Write a python program to create a multiple inheritance using two super classes as Ferrari, Benz and the subclass as Car.

Inheritance is the mechanism to achieve the re-usability of code as one class(child class) can derive the properties of another class(parent class). It also provides transitivity ie. if class C inherits from P then all the sub-classes of C would also inherit from P.

Syntax:

```
Class Base1:
    Body of the class

Class Base2:
    Body of the class

Class Derived(Base1, Base2):
    Body of the class
```

MRO is a concept used in inheritance. It is the order in which a method is searched for in a classes hierarchy and is especially useful in Python because Python supports multiple inheritance.

Practical

```
class Ferrari:
def __init__(self, model):
    self.model = model
def drive(self):
print("Ferrari (self.model) is accelerating!")
def stop(self):
printf("Ferrari {self.model) is stopping!")
```

```
class Benz :
def __init__(self, model):
    self.model =model
def drive(self):
  print f("Benz {self.model's is moving smoothly!")
def stop(self):
  printf("Benz{self.model} is coming to a halt!")
class Car (Ferrari, Benz):
    def __init__(self, model):
      super()__init__(model)
my_car = Car("GTCH Lusso")
my_car.drive()
my_car.stop()
```

OUTPUT

Ferrari GTCH Lusso is accelerating!

Ferrari GTCH Lusso is stopping!

Post Practical Questions:

1. Which of the following best describes inheritance?

a) Ability of a class to derive members of another class as a part of its own definition

b) Means of bundling instance variables and methods in order to restrict access to certain class members

c) Focuses on variables and passing of variables to functions

d) Allows for implementation of elegant software that is well designed and easily modified

2. Which of the following is not a type of inheritance?

a) Double-level

b) Multi-level

c) Single-level

d) Multiple

3.Parent class is the class being inherited from, also called?

a) derived class

b) Child class

c) Hybrid class

d) base class

4.What does built-in function type do in context of classes?

a). Determines the object name of any value

b). Determines the class name of any value

c). Determines class description of any value

d). Determines the file name of any value

Conclusion:

From this practical we got to learn about multi-level inheritance in Java.

Marks out of 10	9
Signature with Date of Completion	28/3/24

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL NO.: 11

Aim: Develop the given programs to learn regular expressions using python

1. Write a Python program to check that a string contains only a certain set of characters (in this case a-z, A-Z and 0-9)
2. Write a Python program that matches a string that has an a followed by zero or one 'b'
3. Write a Python program to find sequences of lowercase letters joined by an underscore.
4. Write a Python program that matches a word containing 'z'.

In Python, regular expressions (often abbreviated as regex or regexp) are a powerful and flexible tool for pattern matching and text manipulation. The re module in Python provides support for regular expressions, allowing you to search, match, and manipulate strings based on specified patterns.

Importing the re Module:

```
import re
```

Basic Patterns: Regular expressions consist of special characters and sequences that represent patterns. For example:

. (dot): Matches any character except a newline.

*: Matches 0 or more occurrences of the preceding character or group.

+: Matches 1 or more occurrences of the preceding character or group.

?: Matches 0 or 1 occurrence of the preceding character or group.

\d: Matches any digit (0-9).

\w: Matches any word character (alphanumeric + underscore).

The re.search() function is used to search for a pattern within a string. It returns a match object if the pattern is found, or None otherwise.

Practical

```
import re
def check_characters(string):
    pattern = re.compile(r'[a-zA-Z0-9]+$')
    if pattern.match(string):
        return True
```

```
    else:
        return False
test_string = "Hello World 123"
if check_characters(test_string):
    print("String contains only specified characters")
else:
    print("String contains characters other than
          specified")
```

OUTPUT

String contains only specified characters.

Post Practical Questions:

1. **Explain regular expression in python.**

Regular expressions in python are sequence of characters used to define search patterns. Enabling sophisticated string manipulation, matching, and extraction based on predefined rule and patterns.

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

2. **Explain use of findall and search method in python.**

In python, the 'findall' method returns all occurrences of a pattern in a string, while the 'search' method finds the first occurence of a patterns a match object.

Conclusion:

From this practical we got to learn about regular expressions in python.

Marks out of 10	
Signature with Date of Completion	

PRACTICAL NO.: 12

Aim: Design a GUI form with a vertical box layout that includes labels and entry fields for user registration information

Python has a lot of GUI frameworks, but Tkinter is the only framework that's built into the Python standard library. Tkinter has several strengths. It's cross-platform, so the same code works on Windows, macOS, and Linux. Visual elements are rendered using native operating system elements, so applications built with Tkinter look like they belong on the platform where they're run.

import the tkinter by following code :

```
import tkinter as tk
```

Tk() creates the main window.

Label, Entry, and Button are different types of widgets.

Label widgets are used to create labels for user instructions.

Entry widget is used for user input.

Button widget is used to trigger an action.

grid method is used for arranging widgets in a grid layout. row and column parameters determine the position in the grid.

pack() is used to organize the widgets in the window. The pady parameter adds vertical padding between each widget, providing spacing where padx parameter adds horizontal padding between each widget, providing spacing.

columnspan is used to make a widget span multiple columns in grid layout.

mainloop() starts the GUI event loop.

Practical

```
import tkinter as tk
def submit_form():
    username = username_entry.get()
    email = email_entry.get()
    password = password_entry.get()
    print("Username:", username)
    print("Email:", email)
    print("Password:", password)
```

```
root = tk.Tk()
root.title("User Registration")
min_frame = tk.Frame(root)
main_frame.pack(padx=20, pady=20)
username_label = tk.label(main_frame, text="Username:")
username_label.grid(row=0, column=0, sticky="w")
username_entry = tk.Entry(main_frame)
username_entry.grid(row=0, column=1)
email_label = tk.label(main_frame, text="Email:")
email_label.grid(row=1, column=0, sticky="w")
password.label = tk.Label(main_frame, text="Password:")
password_entry.grid(row=2, column=0, sticky="w")
password_entry.grid(row=2, column=1)
submit_button = tk.Button(main_frame, text="Submit",
                          command=submit_form)
submit_button.grid(row=3, column span=2, pady=10)
root.mainloop()
```

OUTPUT

Username : johndoe

Email : johndoe@example.com

Password : samplepassword123

Post Practical Questions:

1. **What does the Label widget in Tkinter do?**

   a) Display images

   b) Create text entry fields

   c) Show static text

   d) Handle button clicks

2. **In Tkinter, how can you make a widget respond to a button click?**

   a) Using the bind() method

   b) Setting the command attribute

   c) Connecting a callback function

   d) All of the above

3. **Which widget is used for taking user input in Tkinter?**

   a) Label

   b) Text

   c) Entry

   d) Button

Conclusion: From this practical, we got to learn about widgets in python and tkinter library in python.

Marks out of 10	
Signature with Date of Completion	

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL NO.: 13

Aim: Create a GUI window with a grid layout that contains buttons representing a 3x3 game board

grid method is used for arranging widgets in a grid layout. row and column parameters determine the position in the grid.

pack() is used to organize the widgets in the window. The pady parameter adds vertical padding between each widget, providing spacing where padx parameter adds horizontal padding between each widget, providing spacing.

columnspan is used to make a widget span multiple columns in grid layout.

The sticky parameter is used to specify the placement of the widget within its grid cell.
mainloop() starts the GUI event loop.

```
window = tk.Tk()
window.title("Grid Layout Example")

# Create and add widgets to the window with a grid layout
label_name = tk.Label(window, text="Enter your name:")
label_name.grid(row=0, column=0, padx=10, pady=10, sticky=tk.E)

entry_name = tk.Entry(window)
entry_name.grid(row=0, column=1, padx=10, pady=10)

button_submit = tk.Button(window, text="Submit", command=on_button_click)
button_submit.grid(row=1, column=0, columnspan=2, pady=10)

label_result = tk.Label(window, text="")
label_result.grid(row=2, column=0, columnspan=2, pady=10)

# Start the GUI event loop
window.mainloop()
```

Practical

```
import tkinter as tk
class GameBoard:
    def __init__(self, master):
        self.master = master
        self.master.title("Tic Tac Toe")
```

```
        self.buttons = []
    for i in range(3):
        row=[]
        for j in range(3):
            button = tk.Button(master, width=10, height=5,
                               command=lambda, row=i, col=j:
                               self.button_click(row,col))
            button.grid(row=i, column=j, padx=5, pady=5)
        row.append(button)
    def button_click(self, row, col):
        print("Button clicked at row:", row, "column:", col)
    def main():
        root = tk.TK()
        game = GameBoard(root)
        root.mainloop()
if __name__ == "__main__":
    main().
```

OUTPUT

Button clicked at row: 1 column: 2

Post Practical Questions:

1. **What Tkinter method is commonly used to organize widgets in a grid layout in Python?**

   a) grid()
   b) pack()
   c) place()
   d) layout()

2. **Briefly explain the purpose of the pack() method in Tkinter.**

The pack() method in Tkinter is used to organize and display widgets within a container by automatically arranging them in a specified order and filling available space efficiently.

3. **Which library is commonly used for creating GUI applications in Python?**

   a) NumPy
   b) Matplotlib
   c) Tkinter
   d) Requests

4. **Which Tkinter geometry manager is used for organizing widgets in a grid layout?**

   a) place()
   b) pack()
   c) grid()

d) gridview()

5. **Describe two common layout managers used in Tkinter for organizing widgets in a GUI.**

Two common layout managers used in Tkinter for organising widgets in a GUI are:
1) Pack Manager 2) Grid Manager.

Conclusion:

From this practical we got to learn about how to use Grid layout in GUI in python.

Marks out of 10	
Signature with Date of Completion	

SILVER OAK UNIVERSITY
EDUCATION TO INNOVATION

PRACTICAL NO.: 14

Aim: Create a canvas in your GUI program and draw simple shapes such as rectangles, circles, and lines

The Canvas widget lets us display various graphics on the application. It can be used to draw simple shapes to complicated graphs. We can also display various kinds of custom widgets according to our needs.

Syntax:

C = Canvas(root, height, width, bd, bg, ..)

root = root window.
height = height of the canvas widget.
width = width of the canvas widget.
bg = background colour for canvas.
bd = border of the canvas window.
scrollregion (w, n, e, s)tuple defined as a region for scrolling left, top, bottom and right.
highlightcolor colour shown in the focus highlight.
cursor It can be defined as a cursor for the canvas which can be a circle, a do, an arrow etc.
confine decides if canvas can be accessed outside the scroll region.
relief type of the border which can be SUNKEN, RAISED, GROOVE and RIDGE.

Some common drawing methods:

Creating an Oval

```
oval = C.create_oval(x0, y0, x1, y1, options)
```

Creating an arc

```
arc = C.create_arc(20, 50, 190, 240, start=0, extent=110, fill="red")
```

Creating a Line

```
line = C.create_line(x0, y0, x1, y1, ..., xn, yn, options)
```

Creating a polygon

```
oval = C.create_polygon(x0, y0, x1, y1, ...xn, yn, options)
```

Practical

```
import tkinter as tk
class DrawingApp:
def __init__(self, root):
self.root = root
self.root.title("Drawing App")
self.canvas = tk.Canvas(root, width=400, height=400,
self.canvas.pack()          bg="white")
self.start_x = 50 self.start_y = none
self.selected_shape = none
```

```
        self.canvas.bind("<Button-1>", self.on_click)
        self.canvas.bind("<B1-Motion>", self.on_drag)
        self.canvas.bind("<Button Release-1>", self.on_replace)
        self.shapes = []
        self.shape_type = tk.StringVar(value="Rectangle")
        self.shape_menu = tk.OptionMenu(root, self.shape_type,
                                        "Rectangle", "Circle", "Line")
        self.shape_menu.pack()
def main():
    root = tk.Tk()
    app = DrawingApp(root)
    root.mainloop()
if __name__ == "__main__":
    main()
```

OUTPUT

code executed.

Post Practical Questions:

1. **Describe the role of the Canvas widget in Tkinter.**

The canvas widget in tkinter serves as a drawing surface, allowing users to create and manipulate graphical objects dynamically.

2. **What is the primary purpose of the Canvas widget in Tkinter?**

The primary purpose of the canvas widget in Tkinter is to provide a drawing area for creating and manipulating graphical elements.

3. **Which Tkinter method is commonly used to draw a rectangle on a Canvas?**

The 'create_rectangle()' method in Tkinter is commonly used to draw rectangles on a canvas widget.

4. **Explain the role of the create_line method in the Tkinter Canvas widget.**

The 'create_line' method in Tkinter's canvas widget draws straight lines between specified points, allowing for the creation of line-based graphics.

5. **In Tkinter, what is the purpose of the tag parameter when creating items on a Canvas?**

The tag parameter in Tkinter's canvas widget is used to assign a unique identifier to items, facilitating group operations and manipulation.

Conclusion:

From this practical, we got to learn how to draw shapes on canvas in Python using Tkinter.

Marks out of 10	
Signature with Date of Completion	

arrays, you can use multiple indices:.
example :

```
print(arr[0])
print(arr[1,3])
```

where arr is your single and multidimensional array created using numpy.

Slicing: Slicing allows you to extract a portion (subarray) from an array. The general syntax is start:stop:step. Use negative indices for backward indexing:

example:

```
print(arr[1:4])
print(matrix[:2, 1:]) #rows 0 to 1, columns 1 to end
```

Practical

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
print(arr[0], arr[2])
print(arr[1:4], arr[2:], arr[:4])
print(arr[-1], arr[-2])
arr_2nd = np.array([[1, 2, 3],[4, 5, 6], [7, 8, 9]])
print(arr_2nd[1,1], arr_2nd[0], arr_2d[:, -1])
```

Print (arr-2d[:2, :])

OUTPUT

[2 3 4] [3 4 5] [1 2 3 4]

5 4

5 [1 2 3] [3 6 9]

[[1 2 3] [4 5 6]]

Post Practical Questions:

1: **What is the syntax for slicing in NumPy?**

a) start:stop:step
b) index(start, stop, step)
c) slice(start, stop, step)
d) begin:end:increment

2. **How do you access elements in a 2D NumPy array using indexing?**

a) array(row, column)
b) array[row][column]
c) array[row, column]
d) array.column(row)

3. **What does negative indexing in NumPy allow you to do?**

a) Access elements in reverse order.
b) Perform subtraction operations on elements.
c) Access elements using negative numbers.
d) Create negative numbers in the array.

4. **Which of the following statements about slicing is true in NumPy?**

a) Slicing always includes the stop index.
b) Slicing is not applicable for multi-dimensional arrays.
c) Slicing allows you to skip every second element.
d) Slicing is performed using only positive indices.

Conclusion:

From this practical we got to learn about Indexing and slicing operations on numpy in python.

Marks out of 10	
Signature with Date of Completion	

Practical

```
import pandas as pd
url = "remote-url-here
column_names = ["sepal_length", "sepal_width",
                "petal_length", "petal_width", "class"]
iris_df = pd.read_csv(url, names = column_names)
print("First few rows of the dataset :")
print(iris_df.head())
print("\n checking for missing values :")
print(iris_df.isnull().sum())
print("\n checking for duplicate rows :")
print(iris_df.duplicated().sum())
print("\n Summary statistics :")
print(iris_df.describe())
iris_df = iris_df.drop_duplicate()
iris_df = iris_df.reset_index(drop = True)
print("\n First few rows after data cleaning")
print(iris_df.head())
```

OUTPUT

code executed...

Post Practical Questions:

1. **What is data cleaning?**

Data cleaning in python involves identifying and rectifying errors or inconsistencies in datasets to ensure data quality and accuracy.

2. **How do you identify and remove duplicate entries?**

To identify and remove duplicate entries in python, you can use the 'duplicated()' method to find duplicates and 'drop_duplicates()' to remove them.

3. **What are common types of data errors?**

Common data errors include missing values, duplicates, outliers, inconsistend formats, incorrect data types and inaccurate values.

4. **What is the difference between data cleaning and data preprocessing?**

Data cleaning involves identifying and correcting errors in data. Data preprocessing involves transforming and preparing data for analysis or modeling.

Conclusion:

From this practical, we got to learn about how can we use pandas library in python.

Marks out of 10	
Signature with Date of Completion	

PRACTICAL NO.: 17

Aim: Create a simple line chart using Matplotlib to visualize a dataset.

Matplotlib is a data visualization library in Python. The pyplot, a sublibrary of Matplotlib, is a collection of functions that helps in creating a variety of charts. Line charts are used to represent the relation between two data X and Y on a different axis.

syntax to import the pyplot is :

```
matplotlib.pyplot
```

A simple line chart is generated using NumPy to define data values. The x-values are evenly spaced points, and the y-values are calculated as twice the corresponding x-values.

Practical

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [2, 3, 5, 7, 11]
plt.plot(x, y)
plt.xlabel('X axis')
plt.ylabel('Y axis')
plt.title('Simple Line chart')
# Display chart
plt.show().
```

OUTPUT

Post Practical Questions:

1. **How do you install Matplotlib?**

You can install Matplotlib using pip : 'pip install matplotlib'. Makesure you have python and pip installed before hand.

2. **How do you import Matplotlib in a Python script?**

Your import Matplotlib in a python script using : 'import matplotlib.pyplot as plt' for typical plotting functions.

3. **How do you add labels to the x-axis and y-axis?**

To add labels to the x axis and y axis in Matplot-lib, use 'plt.xlabel("X Label")' and 'plt.ylabel("Y Label")' respectively.

4. **How to give a title to the line chart?**

To give a title to the line chart in python using Matplotlib, use 'plt.title("Chart Title")' before displaying the plot.

Conclusion:

From this practical, we got to learn about Matplotlib in python.

Marks out of 10	
Signature with Date of Completion	