Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belagavi & Approved by AICTE, New Delhi)
(ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

Department of Computer Science & Engineering

Python Programming Laboratory
21CSL46

(CBCS SCHEME)

B.E - IV Semester

Lab Manual 2022-23

CHANNABASAVESHWARA INSTITUTE OF TECHNOLOGY

(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

C.I.T
Partnering in Academic Excellence

LABORATORY MANUAL

PYTHON PROGRAMMING LABORATORY / 21CSL46

(Effective from the academic year 2022 -2023)

Prepared by:
Mrs. Shwetha S
Assistant. Professor
Dept of CSE,CIT GUBBI

Reviewed by:
Dr. Anil Kumar G
Professor
Dean Academics,CIT GUBBI

List of Experiments

PYTHON PROGRAMMING LABORATORY			
Course Code	**21CSL46**	CIE Marks	50
Teaching Hours/Week (L:T:P: S)	0:0:2:0	SEE Marks	50
Total Hours of Pedagogy	24	Total Marks	100
Credits	1	Exam Hours	03

Course Objectives:

CLO 1. Demonstrate the use of IDLE or PyCharm IDE to create Python Applications
CLO 2. Using Python programming language to develop programs for solving real-world problems
CLO 3. Implement the Object-Oriented Programming concepts in Python.
CLO 4. Appraise the need for working with various documents like Excel, PDF, Word and Others
CLO 5. Demonstrate regular expression using python programming

	Note: two hours tutorial is suggested for each laboratory sessions.
	Prerequisite
	➢ Students should be familiarized about Python installation and setting Python environment ➢ Usage of IDLE or IDE like PyCharm should be introduced
Sl. No.	***PART A – List of problems for which student should develop program and execute in the Laboratory***
1	**Aim: Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python** a) Write a python program to find the best of two test average marks out of three test's marks accepted from the user. b) Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.
2	**Aim: Demonstrating creation of functions, passing parameters and return values** a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed. b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions.
3	**Aim: Demonstration of manipulation of strings using string methods** a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters. b) Write a Python program to find the string similarity between two given strings
4	**Aim: Discuss different collections like list, tuple and dictionary** a) Write a python program to implement insertion sort and merge sort using lists b) Write a program to convert roman numbers in to integer values using dictionaries.

5	**Aim: Demonstration of pattern recognition with and without using regular expressions** a) Write a function called isphonenumber() to recognize a pattern 415-555-4242 without using regular expression and also write the code to recognize the same pattern using regular expression. b) Develop a python program that could search the text in a file for phone numbers (+919900889977) and email addresses (sample@gmail.com)
6	**Aim: Demonstration of reading, writing and organizing files.** a) Write a python program to accept a file name from the user and perform the following operations 1. Display the first N line of the file 2. Find the frequency of occurrence of the word accepted from the user in the file b) Write a python program to create a ZIP file of a particular folder which contains several files inside it.
7	**Aim: Demonstration of the concepts of classes, methods, objects and inheritance** a) By using the concept of inheritance write a python program to find the area of triangle, circle and rectangle. b) Write a python program by creating a class called Employee to store the details of Name, Employee_ID, Department and Salary, and implement a method to update salary of employees belonging to a given department.
8	**Aim: Demonstration of classes and methods with polymorphism and overriding** a) Write a python program to find the whether the given input is palindrome or not (for both string and integer) using the concept of polymorphism and inheritance.
9	**Aim: Demonstration of working with excel spreadsheets and web scraping** a) Write a python program to download the all XKCD comics b) Demonstrate python program to read the data from the spreadsheet and write the data in to the spreadsheet
10	**Aim: Demonstration of working with PDF, word and JSON files** a) Write a python program to combine select pages from many PDFs b) Write a python program to fetch current weather data from the JSON file
	PART B – Practical Based Learning
	A problem statement for each batch is to be generated in consultation with the co-examiner and student should develop an algorithm, program and execute the program for the given problem with appropriate outputs.

Course Outcome (Course Skill Set)
At the end of the course the student will be able to:

CO 1. Demonstrate proficiency in handling of loops and creation of functions.
CO 2. Identify the methods to create and manipulate lists, tuples and dictionaries.
CO 3. Discover the commonly used operations involving regular expressions and file system.
CO 4. Interpret the concepts of Object-Oriented Programming as used in Python.

CO 5. Determine the need for scraping websites and working with PDF, JSON and other file formats.
Suggested Learning Resources: 1. Al Sweigart, **"Automate the Boring Stuff with Python",** 1st Edition, No Starch Press, 2015. (Available under CC-BY-NC-SA license at https://automatetheboringstuff.com/) 2. Reema Thareja "**Python Programming Using Problem Solving Approach**" Oxford University Press. 3. Allen B. Downey, **"Think Python: How to Think Like a Computer Scientist",** 2nd Edition, Green Tea Press, 2015. (Available under CC-BY-NC license at http://greenteapress.com/thinkpython2/thinkpython2.pdf)

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each course. The student has to secure not less than 35% (18 Marks out of 50) in the semester-end examination (SEE).

Continuous Internal Evaluation (CIE):

CIE marks for the practical course is **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment to be evaluated for conduction with observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments designed by the faculty who is handling the laboratory session and is made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled downed to 30 marks (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.
- Department shall conduct 02 tests for 100 marks, the first test shall be conducted after the 8th weekof the semester and the second test shall be conducted after the 14th week of the semester.
- In each test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability. Rubrics suggested in Annexure-II of Regulation book
- The average of 02 tests is scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and average marks of two tests is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course is 50 Marks.
- SEE shall be conducted jointly by the two examiners of the same institute, examiners are appointed by the University
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer scriptto be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the internal /external examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.
- General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners)
- *Students can pick one experiment from the questions lot of PART A with equal choice to all the students in a batch. For PART B examiners should frame a question for each batch, student should develop an algorithm, program, execute and demonstrate the results with appropriate output for the given problem.*
- *Weightage of marks for PART A is 80% and for PART B is 20%. General rubrics suggested to be followed for part A and part B.*
- Change of experiment is allowed only once and Marks allotted to the procedure part to bemade zero (Not allowed for Part B).
- The duration of SEE is 03 hours

Rubrics suggested in Annexure-II of Regulation book

General Instructions to Students

1. Students should come with thorough preparation for the experiment to be conducted.
2. Students should take prior permission from the concerned faculty before availing the leave.
3. Students should come with formals and to be present on time in the laboratory.
4. Students will not be permitted to attend the laboratory unless they bring the practical record fully completed in all respects pertaining to the experiments conducted in the previous session.
5. Students will not be permitted to attend the laboratory unless they bring the observation book fully completed in all respects pertaining to the experiments conducted in the present session.
6. They should obtain the signature of the staff-in –charge in the observation book after completing each experiment.
7. Practical record should be neatly maintained.
8. Ask lab Instructor for assistance for any problem.
9. Do not download or install software without the assistance of laboratory Instructor.
10. Do not alter the configuration of system.
11. Turn off the systems after use.

SAMPLE PROGRAMS

1. Write python program to print Hello World

PROGRAM:

```
print("Hello World")
```

OUTPUT: Hello World

2. Write python program to perform String Concatenation

PROGRAM:

```
s1 = "Hello"
s2 = "Good Morning!"
s3 = s1 + " " + s2
print(s3)
```

OUTPUT:

Hello Good Morning!

3. Write python program to perform String repetition

PROGRAM:

```
numbers = [1] * 5
print(numbers)
```

OUTPUT: [1, 1, 1, 1, 1]

PROGRAM:

```
n = [0, 1, 2] * 3
print(n)
```

OUTPUT: [0, 1, 2, 0, 1, 2, 0, 1, 2]

4. Write python program to perform String operations and observe the result.

```
subj="Computer"
subj[0]
subj[0:3]
subj[:4]
subj[:]
```

5. Create a list and perform the following operations

```
list1 = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
list2= [123, 'john']

Print(list1)                    # Prints complete list
print (list1[0] )               # Prints first element of the list
print (list1[1:3])              # Prints elements starting from 2nd till 3rd
print (list1[2:] )              # Prints elements starting from 3rd element
print (list2 * 2)               # Prints list two times
print (list1 + list2)           # Prints concatenated lists
list1[0]=7634                   # Prints the new value 7634 in place of 'abcd'
```

OUTPUT:

```
['abcd', 786, 2.23, 'john', 70.2]
abcd
[786, 2.23]
[2.23, 'john', 70.2]
[123, 'john', 123, 'john']
['abcd', 786, 2.23, 'john', 70.2, 123, 'john']
[7634, 786, 2.23, 'john', 70.2]
```

6. Create a list and perform the following methods
i) append() ii) remove() iii) insert() iv) pop() v) extend()

```
fruits = ['apple', 'banana', 'orange']
fruits.append('kiwi')
```

Output: ['apple', 'banana', 'orange', 'kiwi']

```
fruits.append('apple')
```

Output: ['apple', 'banana', 'orange', 'kiwi', 'apple']

```
fruits.remove('apple')
```

Output: ['banana', 'orange', 'kiwi', 'apple']

fruits.insert(2,'strawberry')

Output: ['banana', 'orange', 'strawberry', 'kiwi', 'apple']

fruits.pop()

Output: 'apple'

fruits.pop(2)

Output: 'strawberry'

print(fruits)

Output: ['banana', 'orange', 'kiwi']

fruits = ['banana', 'orange', 'kiwi']

fruits.**extend**(['apple', 'strawberry'])

Output: ['banana', 'orange', 'kiwi', 'apple', 'strawberry']

7. Create a tuple and perform the following methods

1)Access items 2) len() 3) check for item in tuple 4)Add item

PROGRAM:

```
tup=(1,2,3,4,5,'a','bb','c','xyz')
print("Items in a tuple")
for i in tup:
   print(i,end=" ")
print("\nLength of tuple",len(tup))
i=int(input("enter a value"))
print(i in tup)
tup[1]="hello" print(tup)
```

OUTPUT:

```
Items in a tuple
1 2 3 4 5 a bb c xyz
Length of tuple 9
enter a value4
True
```

```
Traceback (most recent call last):
File "C:/Users/hi/tup1.py", line 8, in <module>
tup[1]="hello"
TypeError: 'tuple' object does not support item assignment
```

8. Write a Program to find largest of a number

PROGRAM:

```
a = 50
b = 55
c = 58
if a == b == c:
   print('They are equal' )
elif a > b and a > c:
   print('a is largest' )
elif b > a and b > c :
   print('b is largest' )
else:
   print('c is largest')
```

OUTPUT:

```
c is largest
```

9. Write a Function to add two numbers and observe the output.

```
def add(a=5, b=10):
   return a+b
add(20,30)
def add(a=5, b=10):
 return a+b
add(20)   # 20 will be going to a
def add(a=5, b=10):
   return a+b
add(b=40)  # b value will replaced
```

PROGRAM :1

1. a. Write a python program to find the best of two test average marks out of three test's marks accepted from the user.

Aim: Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python

PROGRAM : 1A

```
test1 = int(input("Enter marks of Test 1: "))
test2 = int(input("Enter marks of Test 2: "))
test3 = int(input("Enter marks of Test 3: "))
Worst_score = min(test1, test2, test3)
best_average = (test1 + test2 + test3 - Worst_score) / 2
print("Best average of two tests: ", best_average)
```

OUTPUT:

```
In [1]: runfile('E:/Python_lab_programs/program1a.py', wdir='E:/Python_lab_programs')

Enter marks of Test 1: 39

Enter marks of Test 2: 32

Enter marks of Test 3: 45
Best average of two tests:  42.0
```

1. b. Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.

Aim: Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python

PROGRAM : 1B

```
num = int(input("Enter number:"))
temp=num
rev=0
while(num>0):
    dig = num % 10
    rev = rev*10 + dig
    num = num//10
if(temp==rev):
    print(f"The number {temp} is a palindrome!")
else:
    print(f"The number {temp} isn't a palindrome!")
digit_counts = {}
for digit in str(temp):
    if digit in digit_counts:
        digit_counts[digit] += 1
    else:
        digit_counts[digit] = 1
print("Digit counts in", temp, ":")
for digit, count in digit_counts.items():
    print(digit, ":", count)
```

OUTPUT:

```
In [2]: runfile('E:/Python_lab_programs/program1b.py', wdir='E:/Python_lab_programs')

Enter number:12234
The number 12234 isn't a palindrome!
Digit counts in 12234 :
1 : 1
2 : 2
3 : 1
4 : 1
```

```
In [3]: runfile('E:/Python_lab_programs/program1b.py', wdir='E:/Python_lab_programs')

Enter number:345543
The number 345543 is a palindrome!
Digit counts in 345543 :
3 : 2
4 : 2
5 : 2
```

Note: Method 2

```
num = input("Enter number:")
temp=num
rev = num[::-1]
if(temp==rev):
   print(f"The number {temp} is a palindrome!")
else:
   print(f"The number {temp} isn't a palindrome!")

digit_counts = {}  # empty dictionary
for digit in temp:
   if digit in digit_counts:
      digit_counts[digit] += 1
   else:
      digit_counts[digit] = 1

print("Digit counts in", temp, ":")
for digit, count in digit_counts.items():
   print(digit, ":", count)
```

PROGRAM :2

2. a Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.

Aim: Demonstrating creation of functions, passing parameters and return values

PROGRAM: 2A

```
def Fibonacci(n):
    if n<= 0:
        print("Incorrect input, Please enter a positive integer")
    # First Fibonacci number is 0
    elif n == 1:
        return 0
    # Second Fibonacci number is 1
    elif n == 2:
        return 1
    else:
        return Fibonacci(n-1)+Fibonacci(n-2)

n = int(input("Enter a positive integer: "))
print(Fibonacci(n))
```

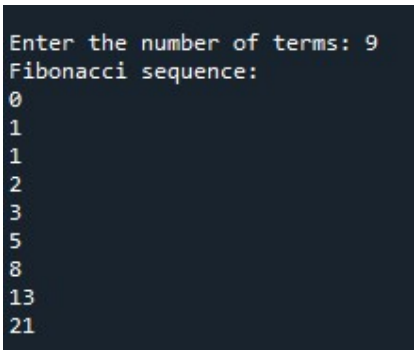
OUTPUT: 2A

```
In [4]: runfile('E:/Python_lab_programs/untitled1.py', wdir='E:/Python_lab_programs')

Enter a positive integer: 4
2

In [5]: runfile('E:/Python_lab_programs/program2a.py', wdir='E:/Python_lab_programs')

Enter a positive integer: 9
21

In [6]: runfile('E:/Python_lab_programs/program2a.py', wdir='E:/Python_lab_programs')

Enter a positive integer: -4
Incorrect input, Please enter a positive integer
None
```

Note: Method 2

```
def fibonacci(n):
    # first two terms
    a, b = 0, 1
    count = 0
    # check if the number of terms is valid
    if n <= 0:
       print("Please enter a positive integer")
    elif n == 1:
       print("Fibonacci sequence upto",n,":")
       print(a)
    else:
       print("Fibonacci sequence:")
       while count < n:
          print(a)
          c = a + b
          # update values of a and b
          a = b
          b = c
          count += 1
# take input from the user
n = int(input("Enter the number of terms: "))
fibonacci(n)
```

OUTPUT:



2. b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions.

PROGRAM: 2B (*binary to decimal)*

```
def binaryToDecimal(binary):

    decimal, i = 0, 0
    while(binary != 0):
        dec_rem = binary % 10
        decimal = decimal + dec_rem * pow(2, i)
        binary = binary//10
        i += 1
    print(f"binary number {bin} in decimal is: ", decimal)

bin = int(input("Enter the binary number for conversion: "))
binaryToDecimal(bin)
```

OUTPUT:

```
In [9]: runfile('E:/Python_lab_programs/program2b1.py', wdir='E:/Python_lab_programs')

Enter the binary number for conversion: 10011
binary number 10011 in decimal is:  19
```

PROGRAM: 2B *(octal to hexadecimal)*

```
octal_number = input("Enter octal number")

# convert octal to decimal
decimal_number = 0
power = len(str(octal_number)) - 1
for digit in str(octal_number):
    decimal_number += int(digit) * 8 ** power
    power -= 1

# convert decimal to hexadecimal
hexadecimal_number=hex(decimal_number)[2:].upper()
print(f"The hexadecimal equivalent of {octal_number} is {hexadecimal_number}"
```

OUTPUT:

```
In [11]: runfile('E:/Python_lab_programs/program2b1.py', wdir='E:/Python_lab_programs')

Enter the binary number for conversion: 1100
binary number 1100 in decimal is:  12

Enter octal number12
The hexadecimal equivalent of 12 is A
```

PROGRAM:3

3. a. Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters.

Aim: Demonstration of manipulation of strings using string methods

PROGRAM: 3A

```
sentence = input("Enter a sentence: ")

num_words = len(sentence.split())
num_digits = 0
num_uppercase = 0
num_lowercase = 0

for char in sentence:
    if char.isdigit():
        num_digits += 1
    elif char.isupper():
        num_uppercase += 1
    elif char.islower():
        num_lowercase += 1

print("Number of words:", num_words)
print("Number of digits:", num_digits)
print("Number of uppercase letters:", num_uppercase)
print("Number of lowercase letters:", num_lowercase)
```

OUTPUT:3A

```
In [14]: runfile('E:/Python_lab_programs/program3a.py', wdir='E:/Python_lab_programs')

Enter a sentence: Working on python version 3
Number of words: 5
Number of digits: 1
Number of uppercase letters: 1
Number of lowercase letters: 21
```

3b. Write a Python program to find the string similarity between two given strings. Sample Output: Original string: Python Exercises Python Exercises Similarity between two said strings: 1.0 Sample Output: Original string: Python Exercises Python Exercise Similarity between two said strings: 0.967741935483871

PROGRAM: 3B

```
from difflib import SequenceMatcher
string1 = "Python Exercises"
string2 = "Python Exercise"
similarity = SequenceMatcher(None, string1, string2).ratio()
print("Original string:")
print(string1)
print(string2)
print("Similarity between two said strings:")
print(similarity)
```

OUTPUT:

```
In [15]: runfile('E:/Python_lab_programs/program3b.py', wdir='E:/Python_lab_programs')
Original string:
Python Exercises
Python Exercise
Similarity between two said strings:
0.967741935483871

In [16]: runfile('E:/Python_lab_programs/program3b.py', wdir='E:/Python_lab_programs')
Original string:
Python Exercises
Python Exercises
Similarity between two said strings:
1.0
```

PROGRAM:4

4. a. Write a python program to implement insertion sort and merge sort using lists

Aim: Discuss different collections like list, tuple and dictionary

PROGRAM: 4A *(insertion sort)*

```
def insertion_sort(alist):
    for i in range(1, len(alist)):
        temp = alist[i]
        j = i - 1
        while (j >= 0 and temp < alist[j]):
            alist[j + 1] = alist[j]
            j = j - 1
        alist[j + 1] = temp


alist = input('Enter the list of numbers: ').split()
alist = [int(x) for x in alist]
insertion_sort(alist)
print('Sorted list: ', end='')
print(alist)
```

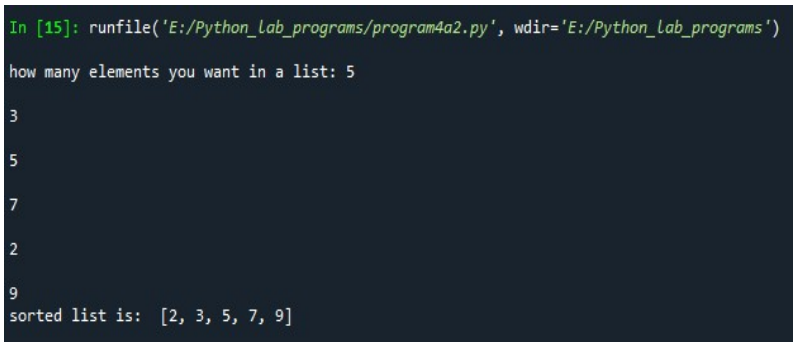
OUTPUT: 4A (insertion sort)

```
In [17]: runfile('E:/Python_lab_programs/program4a.py', wdir='E:/Python_lab_programs')

Enter the list of numbers: 1 0 -5 3 8 9
Sorted list: [-5, 0, 1, 3, 8, 9]
```

PROGRAM: 4A *(merge sort)*

```
def merge_sort(arr):
    if len(arr) > 1:
        mid = len(arr) // 2
        left_half = arr[:mid]
        right_half = arr[mid:]
        merge_sort(left_half)
        merge_sort(right_half)
        i = j = k = 0

        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                arr[k] = left_half[i]
                i += 1
            else:
                arr[k] = right_half[j]
                j += 1
            k += 1

        while i < len(left_half):
            arr[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            arr[k] = right_half[j]
            j += 1
            k += 1
    return arr
num = int(input("how many elements you want in a list: "))
arr = [int(input()) for x in range(num)]
merge_sort(arr)
print("sorted list is: ", arr)
```

OUTPUT:



4. b. Write a program to convert roman numbers in to integer values using dictionaries.

PROGRAM: 4B

```
def roman_to_int(roman_numeral):
    roman_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}
    result = 0
    prev_value = 0
    for i in range(len(roman_numeral)-1, -1, -1):
        current_value = roman_dict[roman_numeral[i]]
        if current_value < prev_value:
            result -= current_value
        else:
            result += current_value
        prev_value = current_value
    return result


print(roman_to_int('XXIV'))
print(roman_to_int('MMMCMLXXXVI'))
```

OUTPUT:

```
In [20]: runfile('E:/Python_lab_programs/program4b.py', wdir='E:/Python_lab_programs')
24
3986
```

PROGRAM: 5

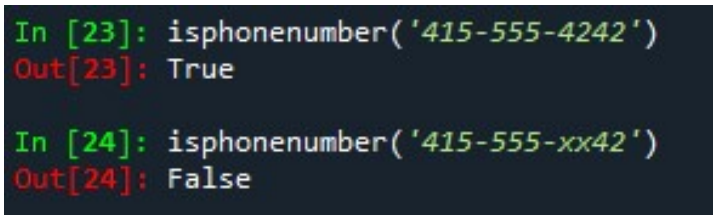
5. a. Write a function called isphonenumber() to recognize a pattern 415-555-4242 without using regular expression and also write the code to recognize the same pattern using regular expression.

Aim: Demonstration of pattern recognition with and without using regular expressions

PROGRAM: 5A

```
# How to recognize the pattern without using regular expressions:

    def isphonenumber(text):
        if len(text) != 12:
            return False
        for i in range(0, 3):
            if not text[i].isdecimal():
                return False
        if text[3] != '-':
            return False
        for i in range(4, 7):
            if not text[i].isdecimal():
                return False
        if text[7] != '-':
            return False
        for i in range(8, 12):
            if not text[i].isdecimal():
                return False
        return True
# call function with a string argument to check if it matches the pattern of a phone number
isphonenumber('415-555-4242')
```

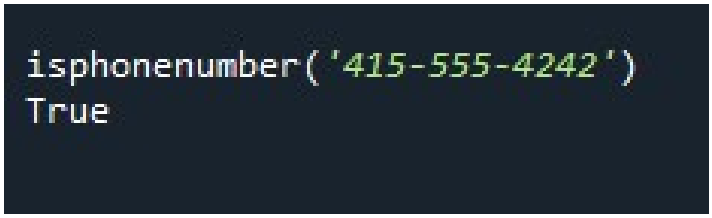
OUTPUT:



 # How to recognize the pattern using regular expressions:

```
import re

number_pattern = re.compile(r'\d{3}-\d{3}-\d{4}')

def isphonenumber(number):
    if number_pattern.search(number):
        return True
    else:
        return False

isphonenumber('415-555-4242')
```

OUTPUT:



5. b. Write Develop a python program that could search the text in a file for phone numbers (+919900889977) and email addresses (sample@gmail.com)

PROGRAM:5B

```
with open('filename.txt', 'r') as file:
text = file.read()
import re
# Search for phone numbers using regular expressions
phone_numbers = re.findall(r'\+91\d{10}', text)

# Search for email addresses using regular expressions
email_addresses = re.findall(r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b', text)

# Print the results
print("Phone numbers found:", phone_numbers)
print("Email addresses found:", email_addresses)
```

OUTPUT:

```
In [1]: runfile('E:/Python_lab_programs/program5b.py', wdir='E:/Python_lab_programs')
Phone numbers found: ['+919900889977']
Email addresses found: ['sample@gmail.com', 'tejaswini@gmail.com']
```

PROGRAM: 6

6 a. Write a python program to accept a file name from the user and perform the following operations

1. Display the first N line of the file

2. Find the frequency of occurrence of the word accepted from the user in the file

PROGRAM:6A.1

```
inputFile = "exampletextfile.txt"          # input text file
N = int(input("Enter N value: "))          # Enter N value
with open(inputFile, 'r') as filedata:     # Opening the given file in read-only mode
   linesList= filedata.readlines()         # Read the file lines using readlines()
print("The following are the first",N,"lines of a text file:")
for textline in (linesList[:N]):           # Traverse in the list of lines to retrieve the first N lines of a file
   print(textline, end ='')                # Printing the first N lines of the file line by line.
filedata.close()                           # Closing the input file
```

OUTPUT:

```
In [1]: runfile('E:/Python_lab_programs/program6a.py', wdir='E:/Python_lab_programs')

Enter N value: 3
The following are the first 3 lines of a text file:
Python is a high-level, general-purpose programming language.
Its design philosophy emphasizes code readability with the use of significant indentation
via the off-side rule.[33]
Python is dynamically typed and garbage-collected.
```

PROGRAM:6A.2

```
fname = input("Enter file name: ")
word=input("Enter word to be searched:")
k = 0
with open(fname, 'r') as f:
    for line in f:
        words = line.split()
        for i in words:
            if(i==word):
                k=k+1
print(f"Occurrences of the word {word} is:" , k )
```

OUTPUT:

```
In [4]: runfile('E:/Python_lab_programs/program6a2.py', wdir='E:/Python_lab_programs')

Enter file name: exampletextfile.txt

Enter word to be searched:Python
Occurrences of the word Python is: 9
```

6 b. Write a python program to create a ZIP file of a particular folder which contains several files inside it.

PROGRAM:6B

```
import os
from zipfile import ZipFile

# Create object of ZipFile
with ZipFile('E:/Zipped file.zip', 'w') as zip_object:
  # Traverse all files in directory
  for folder_name, sub_folders, file_names in os.walk('E:\CIT\DSD lab programs'):
    for filename in file_names:
      # Create filepath of files in directory
      file_path = os.path.join(folder_name, filename)
      # Add files to zip file
      zip_object.write(file_path, os.path.basename(file_path))

if os.path.exists('E:/Zipped file.zip'):
  print("ZIP file created")
else:
  print("ZIP file not created")
```

OUTPUT:

```
In [5]: runfile('E:/Python_lab_programs/program6b.py', wdir='E:/Python_lab_programs')
ZIP file created
```

PROGRAM: 7

7 a. By using the concept of inheritance write a python program to find the area of triangle, circle and rectangle.

PROGRAM: 7A

```
class Shape:
    def area(self):
        pass
class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height
    def area(self):
        return 0.5 * self.base * self.height
class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius
    def area(self):
        return 3.14 * self.radius * self.radius
class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width
    def area(self):
        return self.length * self.width

# Example usage
t = Triangle(10, 5)
print("Area of triangle:", t.area())

c = Circle(7)
print("Area of circle:", c.area())

r = Rectangle(8, 6)
print("Area of rectangle:", r.area())
```

OUTPUT:

```
In [1]: runfile('E:/Python_lab_programs/untitled0.py', wdir='E:/Python_lab_programs')
Area of triangle: 25.0
Area of circle: 153.86
Area of rectangle: 48
```

7b. Write a python program by creating a class called Employee to store the details of Name, Employee_ID, Department and Salary, and implement a method to update salary of employees belonging to a given department.

PROGRAM: 7B

```
class Employee:
    def __init__(self):
        self.name = ""
        self.employee_Id = ""
        self.department = ""
        self.salary = 0

    def getEmpDetails(self):
        self.name = input("Enter Employee name : ")
        self.employee_Id = input("Enter Employee ID : ")
        self.department = input("Enter Employee Dept : ")
        self.salary = int(input("Enter Employee Salary : "))

    def showEmpDetails(self):
        print("Employee Details")
        print("Name : ", self.name)
        print("ID : ", self.employee_Id)
        print("Dept : ", self.department)
        print("Salary : ", self.salary)

    def updtSalary(self):
        self.salary = int(input("Enter new Salary : "))
        print("Updated Salary", self.salary)


e1 = Employee()
e1.getEmpDetails()
e1.showEmpDetails()
e1.updtSalary()
```

OUTPUT:

```
In [2]: runfile('E:/Python_lab_programs/untitled1.py', wdir='E:/Python_lab_programs')

Enter Employee name : Usha

Enter Employee ID : 897

Enter Employee Dept : MBA

Enter Employee Salary : 45000
Employee Details
Name :  Usha
ID :  897
Dept :  MBA
Salary :  45000

Enter new Salary : 55000
Updated Salary 55000
```

PROGRAM: 8

8. Aim: Demonstration of classes and methods with polymorphism and overriding

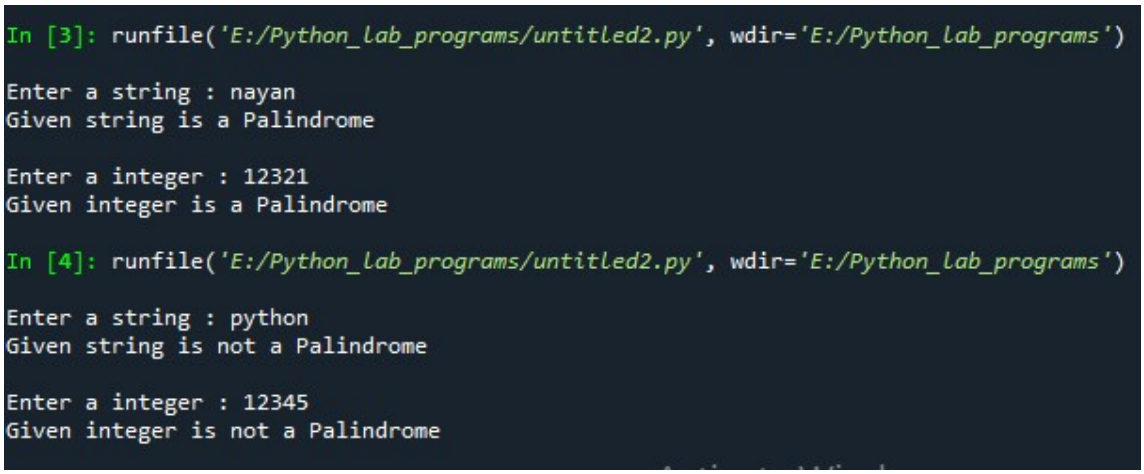
a) Write a python program to find the whether the given input is palindrome or not (for both string and integer) using the concept of polymorphism and inheritance.

PROGRAM: 8

```
class PaliStr:
    def __init__(self):
        self.isPali = False

    def chkPalindrome(self, myStr):
        if myStr == myStr[::-1]:
            self.isPali = True
        else:
            self.isPali = False

        return self.isPali

class PaliInt(PaliStr):
    def __init__(self):
        self.isPali = False

    def chkPalindrome(self, val):
        temp = val
        rev = 0
        while temp != 0:
            dig = temp % 10
            rev = (rev*10) + dig
            temp = temp //10

        if val == rev:
            self.isPali = True
        else:
            self.isPali = False

        return self.isPali

st = input("Enter a string : ")

stObj = PaliStr()
```

```
if stObj.chkPalindrome(st):
    print("Given string is a Palindrome")
else:
    print("Given string is not a Palindrome")

val = int(input("Enter a integer : "))

intObj = PaliInt()
if intObj.chkPalindrome(val):
    print("Given integer is a Palindrome")
else:
    print("Given integer is not a Palindrome")
```

OUTPUT:



PROGRAM: 9

9. Aim: Demonstration of working with excel spreadsheets and web scraping
a) Write a python program to download the all XKCD comics
b) Demonstrate python program to read the data from the spreadsheet and write the data in to the spreadsheet

PROGRAM: 9.A

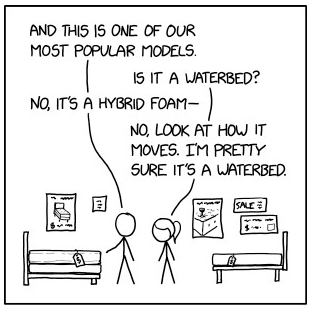
```
import requests
import os
from bs4 import BeautifulSoup

# Set the URL of the first XKCD comic
url = 'https://xkcd.com/1/'

# Create a folder to store the comics
if not os.path.exists('xkcd_comics'):
    os.makedirs('xkcd_comics')

# Loop through all the comics
while True:
    # Download the page content
    res = requests.get(url)
    res.raise_for_status()

    # Parse the page content using BeautifulSoup
    soup = BeautifulSoup(res.text, 'html.parser')

    # Find the URL of the comic image
    comic_elem = soup.select('#comic img')
    if comic_elem == []:
```

```
            print('Could not find comic image.')
        else:
            comic_url = 'https:' + comic_elem[0].get('src')

            # Download the comic image
            print(f'Downloading {comic_url}...')
            res = requests.get(comic_url)
            res.raise_for_status()

            # Save the comic image to the xkcd_comics folder
              image_file = open(os.path.join('xkcd_comics',
os.path.basename(comic_url)), 'wb')
            for chunk in res.iter_content(100000):
               image_file.write(chunk)
            image_file.close()

        # Get the URL of the previous comic
        prev_link = soup.select('a[rel="prev"]')[0]
        if not prev_link:
            break
        url = 'https://xkcd.com' + prev_link.get('href')

    print('All comics downloaded.')
```

OUTPUT:

```
In [1]: runfile('E:/Python_lab_programs/untitled0.py', wdir='E:/Python_lab_programs')
Downloading https://imgs.xkcd.com/comics/barrel_cropped_(1).jpg...
Downloading https://imgs.xkcd.com/comics/physical_quantities.png...
Downloading https://imgs.xkcd.com/comics/exoplanet_high_5.png...
Downloading https://imgs.xkcd.com/comics/cuisine.png...
Downloading https://imgs.xkcd.com/comics/noise_filter.png...
Downloading https://imgs.xkcd.com/comics/crystal_ball.png...
Downloading https://imgs.xkcd.com/comics/siphon.png...
Downloading https://imgs.xkcd.com/comics/taxiing.png...
Downloading https://imgs.xkcd.com/comics/planetary_scientist.png...
Downloading https://imgs.xkcd.com/comics/commemorative_plaque.png...
Downloading https://imgs.xkcd.com/comics/college_knowledge.png...
Downloading https://imgs.xkcd.com/comics/tapetum_lucidum.png...
Downloading https://imgs.xkcd.com/comics/overlapping_circles.png...
Downloading https://imgs.xkcd.com/comics/definition_of_e.png...
Downloading https://imgs.xkcd.com/comics/recipe_relativity.png...
```

Few comics downloaded from above output



planetary_scientist.png

PROGRAM: 9.B

```
from openpyxl import Workbook
from openpyxl.styles import Font

wb = Workbook()
sheet = wb.active
sheet.title = "Language"
wb.create_sheet(title = "Capital")


lang = ["Kannada", "Telugu", "Tamil"]
state = ["Karnataka", "Telangana", "Tamil Nadu"]
capital = ["Bengaluru", "Hyderabad", "Chennai"]
code =['KA', 'TS', 'TN']


sheet.cell(row = 1, column = 1).value = "State"
sheet.cell(row = 1, column = 2).value = "Language"
```

```
sheet.cell(row = 1, column = 3).value = "Code"


ft = Font(bold=True)
for row in sheet["A1:C1"]:
    for cell in row:
        cell.font = ft

for i in range(2,5):
    sheet.cell(row = i, column = 1).value = state[i-2]
    sheet.cell(row = i, column = 2).value = lang[i-2]
    sheet.cell(row = i, column = 3).value = code[i-2]


wb.save("demo.xlsx")
sheet = wb["Capital"]
sheet.cell(row = 1, column = 1).value = "State"
sheet.cell(row = 1, column = 2).value = "Capital"
sheet.cell(row = 1, column = 3).value = "Code"


ft = Font(bold=True)
for row in sheet["A1:C1"]:
    for cell in row:
        cell.font = ft
for i in range(2,5):
    sheet.cell(row = i, column = 1).value = state[i-2]
    sheet.cell(row = i, column = 2).value = capital[i-2]
    sheet.cell(row = i, column = 3).value = code[i-2]

wb.save("demo.xlsx")
srchCode = input("Enter state code for finding capital ")
for i in range(2,5):
    data = sheet.cell(row = i, column = 3).value
    if data == srchCode:
        print("Corresponding capital for code", srchCode, "is", sheet.cell(row = i, column =
```

```
    2).value)

sheet = wb["Language"]
srchCode = input("Enter state code for finding language ")
for i in range(2,5):
    data = sheet.cell(row = i, column = 3).value
    if data == srchCode:
        print("Corresponding language for code", srchCode, "is", sheet.cell(row = i, column =
    2).value)
    wb.close()
```

OUTPUT:

```
In [6]: runfile('E:/Python_lab_programs/program9b.py', wdir='E:/Python_lab_programs')

Enter state code for finding capital KA
Corresponding capital for code KA is Bengaluru

Enter state code for finding language TN
Corresponding language for code TN is Tamil
```

PROGRAM: 10

10. Aim: Demonstration of working with PDF, word and JSON files

a) Write a python program to combine select pages from many PDFs

b) Write a python program to fetch current weather data from the JSON file

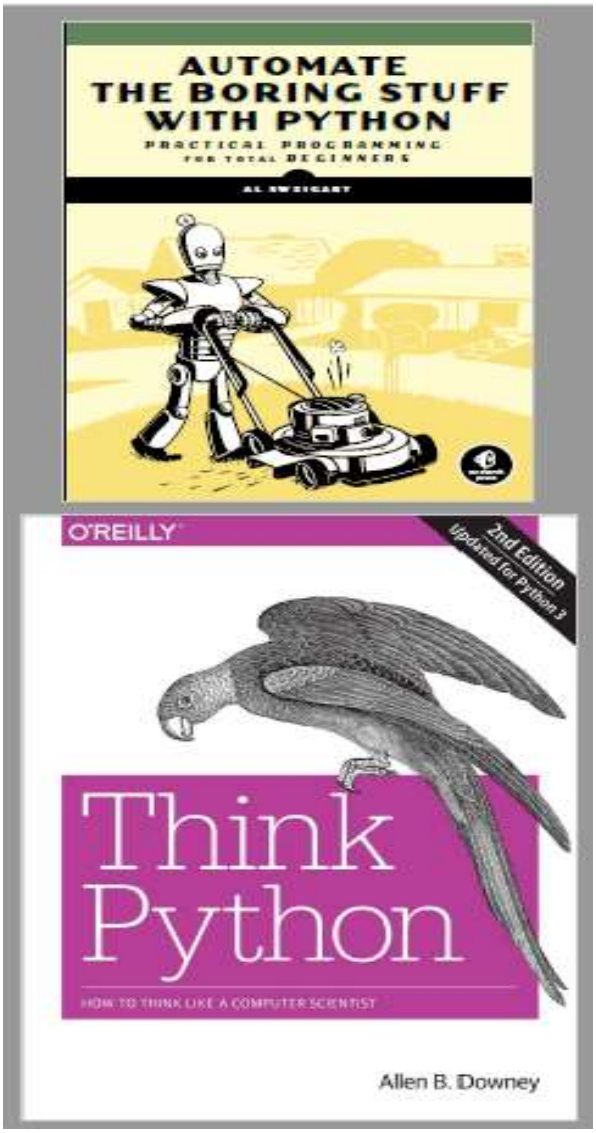
PROGRAM: 10.A

```
from PyPDF2 import PdfWriter, PdfReader

num = int(input("Enter page number you want combine from multiple documents "))

pdf1 = open('10a1.pdf', 'rb')
pdf2 = open('10a2.pdf', 'rb')

pdf_writer = PdfWriter()
```

```
pdf1_reader = PdfReader(pdf1)
page = pdf1_reader.pages[num - 1]
pdf_writer.add_page(page)

pdf2_reader = PdfReader(pdf2)
page = pdf2_reader.pages[num - 1]
pdf_writer.add_page(page)

with open('output.pdf', 'wb') as output:
   pdf_writer.write(output)
```

OUTPUT:

This program allows you to extract specific pages from two PDF files, “10a1.pdf” and “10a2.pdf,” by entering the page numbers as user input. Once you input the desired page numbers, the program fetches those pages from both PDF files and combines them into a new file called “output.pdf.” This way, you can easily compile the desired pages from multiple PDF files into one document for your convenience.


AUTOMATE
THE BORING STUFF
WITH PYTHON
PRACTICAL PROGRAMMING
FOR TOTAL BEGINNERS
O'REILLY
2nd Edition
Updated for Python 3
Think
Python
HOW TO THINK LIKE A COMPUTER SCIENTIST
Allen B. Downey

PROGRAM: 10.B

Steps:

1. Import the `json` module in your Python script.
2. Open the JSON file using the `open()` function and read its contents using the `read()` method.
3. Use the `json.loads()` method to parse the JSON data into a Python dictionary.
4. Access the weather data from the dictionary using the appropriate keys.

```
import json

# Open the JSON file and read its contents
with open('weather.json', 'r') as f:
    data = f.read()

# Parse the JSON data into a Python dictionary
weather_dict = json.loads(data)

# Access the weather data from the dictionary
temperature = weather_dict['main']['temp']
humidity = weather_dict['main']['humidity']
description = weather_dict['weather'][0]['description']

# Print the weather data
print(f"Temperature: {temperature}°C")
print(f"Humidity: {humidity}%")
print(f"Description: {description}")
```

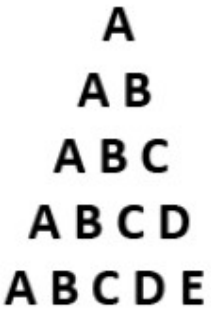
OUTPUT:

```
In [6]: runfile('E:/Python_lab_programs/program10bN.py', wdir='E:/Python_lab_programs')
Temperature: 15.45°C
Humidity: 64%
Description: clear sky
```

PART B

SAMPLE QUESTIONS ONLY

1. Write Python programs to print the following Patterns:



2. Create a simple Calculator which performs basic operations (+,-,/,*) .
3. Implement the following Searching and Sorting techniques:
   i) Bubble Sort ii) Linear
4. Write a program to ask the user to enter a number and determine if the number is 1 digit, 2 digits or more than 2 digits long.
5. Write a function that computes and displays all multiplication table combinations from 2 to 12.
6. Write a python program that displays all of the prime numbers from 2 to 100.
7. You are designing a decision structure to convert a student's numeric grade to a letter grade.
   The program must assign a letter grade as specified in the following table:
   Percentage range Letter grade 90 through 100 A
   80 through 89 B
   70 through 79 C
   65 through 69 D
   O through 64 F
   For example, if the user enters a 90, the output should be," Your letter grade is A."
   Likewise, if a user enters an 89, the Output should be "Your letter grade is B."
8. You are writing a function that assigns a rating based on a user's age.
   The function must meet the following requirements:
   Anyone 18 years old or older receives a rating of “A”
   Anyone 13 or older, but younger than 18, receives a rating of "T”.
   Anyone 12 years old or younger receives a rating of “c”
   If the age is unknown, the rating is set to "C".
9. Create a list that contains words in uppercase and lowercase and write a program to display only uppercase words.
10. Write a program to reverse a given number.

VIVA QUESTIONS

1. What is Python? What are the applications of python?
2. Define variables in python. List Rules for Python variables.
3. What is the use of type() function?
4. What is type conversion in python?
5. Explain different datatypes in python.
6. Define list?
7. List out the methods of list?
8. What is list indexing and slicing with an example?
9. What is the difference between lists and tuples?
10. What is dictionary in python?
11. What are python modules?
12. List Python Operators.
13. What are the various Python Conditions and If statements?
14. What is for loop?
15. What are called as flow control statements in python?
16. Define range() function.
17. Define function.
18. How to create a Function in python?
19. Define exception?
20. List out different types of errors and exception?
21. Explain in brief how to handle exceptions?
22. Define file and what are the different modes of files?
23. Differentiate between built – in and user-define functions?
24. What are the common built-in data types in python?
25. Define Exception Handling.
26. What are the File opening modes in python.
27. Explain all file processing modes supported in python?
28. Define Object Methods.
29. Define class and object creation with syntax?
30. What are the different types of constructors define them?
31. Define Python Inheritance
32. What is RegEx ?
33. What is a Package?
34. What is the use of JSON.