

A
Lab Record of
Python Programming with Problem
Solving

Master of Computer Application - I Sem



RUNGTA INTERNATIONAL SKILLS UNIVERSITY

SESSION: 2025-26

Guided By:-
Mrs. Kavita Kanwar
(Assistant Professor)

Submitted By:-
Sumit Kumar
Ref No. 11485

Submitted To:

RUNGTA INTERNATIONAL SKILLS
UNIVERSITY, CG

SCHOOL OF INFORMATION TECHNOLOGY

Index

S. No	Name Of Practical	Submission Date	Remarks
1.	Write a Python program to Print the Patterns.		
2.	Write a Python program to the probability of rolling a dice / flipping a coin.		
3.	Write a Python program to find factorial of a number using recursion.		
4.	Write a program to search for an item in a user-provided list and display the position if found, otherwise print "Item not found."		
5.	Given a list of employee records as dictionaries, sort them by salary and display the sorted list.		
6.	Write a program that reads a text file and counts the number of lines, words, and characters.		
7.	Read a sentence and display how many times each word appears, ignoring case and punctuation.		
8.	Write a Python program that lists all files in a directory and categorizes them by file extension.		

PRACTICAL ASSIGNMENT

AIM 1 - Write a Python program to Print the Patterns

python_practical_1.ipynb X

C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > python > python_practical_1.ipynb > print("Pyramid Pattern")

Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...

Get

```
print("Number Pattern")
for i in range(1,6):#Row
    for j in range(1,i+1):#Colom
        print(j,end=' ')
    print() #For New Line
```

[1]

... Number Pattern

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

▷

```
print("Pyramid Pattern")
for i in range(1,6):#For Row
    for s in range(5-i):#For Space
        print(" ",end='')#Space print
    for j in range(1,i+1):#For Column
        print("*",end=' ')
    print()
```

[7] ✓ 0.0s

... Pyramid Pattern

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

python_practical_1.ipynb X

C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > python > python_practical_1.ipynb > print("Pyramid Pattern")

Generate + Code + Markdown | Run All Restart Clear All Outputs Jupyter Variables Outline ...

```
print("Diamond Payyern")

#upper diamond
space1=4
for i in range(1,6):
    for s in range(space1):
        print(" ",end='')
    for j in range(1,i+1):
        print("*",end=' ')
    print()
    space1-=1
#lower diamond
space2=0
for i in range(5,0,-1):
    for s in range(space2):
        print(" ",end='')
    for j in range(1,i+1):
        print("*",end=' ')
    print()
    space2+=1
```

[1]

... Diamond Payyern

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

python_practical_1.ipynb X

C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > python > python_practical_1.ipynb > print("X Shaped Pattern")

Generate + Code + Markdown | Run All Restart Clear All Outputs Jupyter Variables Outline ...

```
print("Hallow Square Pattern")

for row in range(1,6):
    for col in range(1,6):
        if col==1 or col==5 or row==1 or row==5:
            print("*",end=' ')
        else:
            print(" ",end=' ')
    print()
```

[2]

... Hallow Square Pattern

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

```
print("X Shaped Pattern")
```

```
n = 5 # must be odd
for i in range(n):
    for j in range(n):
        if i == j or i + j == n - 1:
            print("*", end="")
        else:
            print(" ", end="")
    print()
```

[8]

... X Shaped Pattern

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

AIM 2 - Write a Python program to the probability of rolling a dice / flipping a coin.

python_practical_2.ipynb X

C:\Users> sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > python > python_practical_2.ipynb > #Function to find probability of rollong a specific number on a dic

Generate + Code + Markdown Run All Clear All Outputs Outline ...

```
#Function to find probability of rollong a specific number on a dice
def dice_probability(number):
    if 1 <= number <= 6:
        probability = 1 / 6 #total sides in a dice 6
        return probability
    else:
        return "Invalid number! Dice has numbers from 1 to 6 only."

#Funtion to find probability of flipping a coin (head or tail)
def coin_probability(outcome):
    outcomes=['head','tail']
    if outcome.lower() in outcomes:
        probability = 1 / 2
        return probability
    else:
        return "Invalid outcome! Choose 'head' or 'tail'."
```

[]

```
#Main prigram
print("----probability calculator ----")
choice = input("Enter 'dice' to roll a dice or 'coin' to flip a coin: ").lower()
print(f'choice:{choice}')

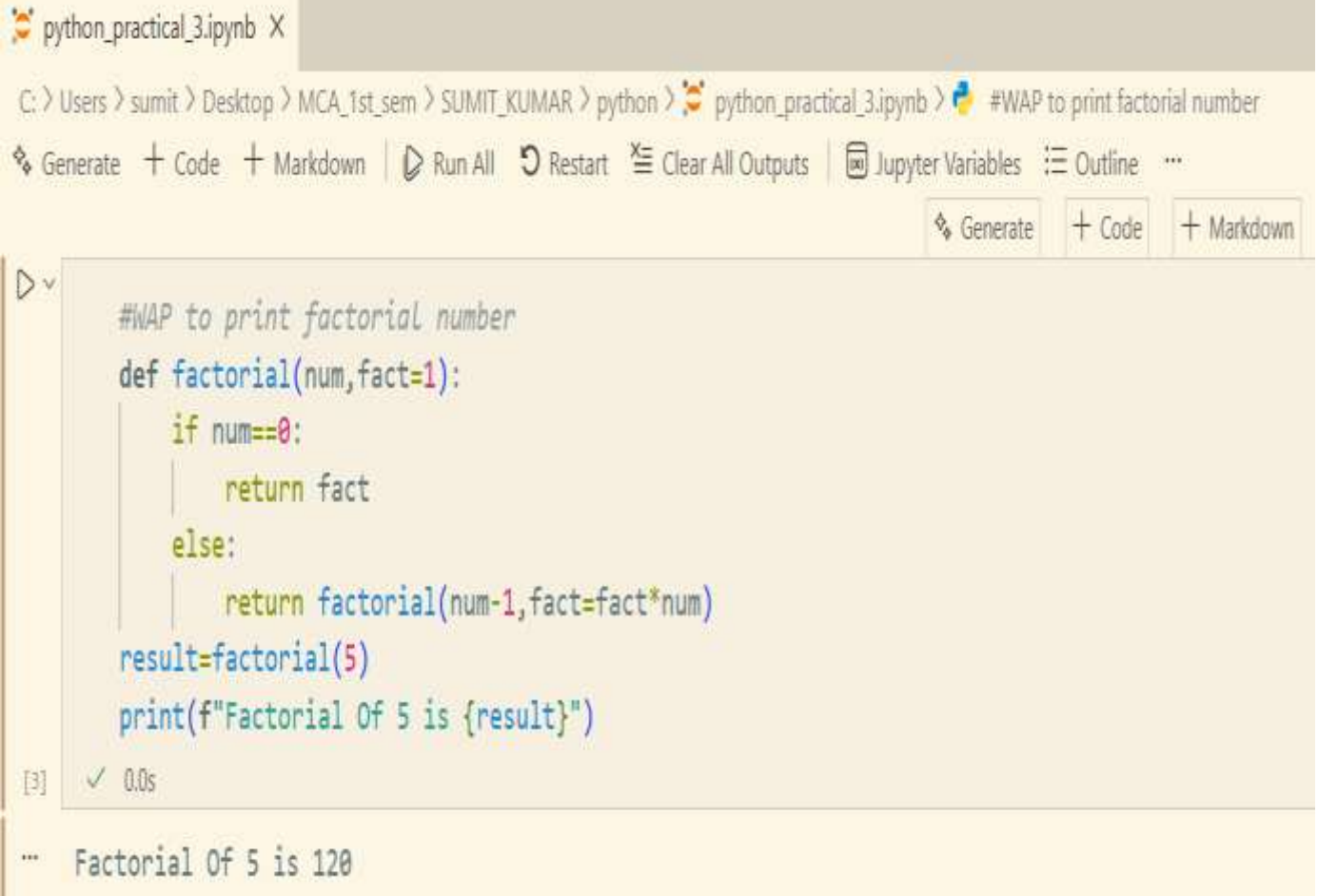
if choice == 'dice':
    number = int(input("Enter the number you want(1-6):"))
    result = dice_probability(number)
    print(f"The probability of getting {number} is: {result}")

elif choice == 'coin':
    outcome = input("Enter your choice(Head/Tail):")
    result = coin_probability(outcome)
    print(f"The probability of getting '{outcome}' is: {result}")

else:
    print("Invalid choice! please enter 'dice' or 'coin'.")
```

''' ----probability calculator ----

AIM 3 - Write a Python program to find factorial of a number using recursion.



The screenshot shows a Jupyter Notebook interface with a single code cell. The code defines a recursive function `factorial` that calculates the factorial of a number. The function has two parameters: `num` and `fact` (with a default value of 1). The base case is `if num==0: return fact`. The recursive case is `else: return factorial(num-1, fact=fact*num)`. The code then calls `result=factorial(5)` and prints the result using `print(f"Factorial Of 5 is {result}")`. The output of the cell is "Factorial Of 5 is 120".

```
python_practical_3.ipynb X  
C:\Users\sumit\Desktop\MCA_1st_sem\SUMIT_KUMAR>python>python_practical_3.ipynb> #WAP to print factorial number  
Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline ...  
Generate + Code + Markdown  
#WAP to print factorial number  
def factorial(num,fact=1):  
    if num==0:  
        return fact  
    else:  
        return factorial(num-1,fact=fact*num)  
result=factorial(5)  
print(f"Factorial Of 5 is {result}")  
[3] ✓ 0.0s  
... Factorial Of 5 is 120
```

AIM 4 - Write a program to search for an item in a user-provided list and display the position if found, otherwise print “Item not found.”

```
python_practical_4.ipynb X Enter the list element or type 'exit' to stop: (Press Enter to confirm or escape to stop)

C:\Users\sumit\Desktop> MCA_1st_sem > SUMIT_KUMAR > python > python_practical_4.ipynb > def item_search(list_elements):

Generate + Code + Markdown | Interrupt Restart Clear All Outputs Go To Jupyter Variables Outline ...

def item_search(list_elements):
    item = input("Enter the item to be searched: ")
    position = 1

    print(f"Searched item: {item}")

    if item not in list_elements:
        print("Item not found")
    else:
        for element in list_elements:
            if element == item:
                print(f"{item} : Item found at position {position}")
                position += 1

def take_input():
    list_elements = []

    while True:
        element = input("Enter the list element or type 'exit' to stop: ")

        if element.lower() == "exit":
            break
        else:
            list_elements.append(element)

    return list_elements

# Run the program
items = take_input()
item_search(items)
```


AIM 5 - Given a list of employee records as dictionaries, sort them by salary and display the sorted list.

```
python_practical_5.ipynb X

C:\Users> sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > python > python_practical_5.ipynb > employee_list=[

Generate + Code + Markdown Run All Restart Clear All Outputs Jupyter Variables Outline ...

Generate + Code + Markdown

> v
employee_list=[
    {'Name':'John','Salary':45000},
    {'Name':'Mary','Salary':52000},
    {'Name':'Alex','Salary':48000}
]
print(f'Unsorted list:{employee_list}')

#sort the original list in place
employee_list.sort(key=lambda x:x["Salary"])
print(f'Sorted list by Salary:{employee_list}')

[4] ✓ 0.0s

... Unsorted list:[{'Name': 'John', 'Salary': 45000}, {'Name': 'Mary', 'Salary': 52000}, {'Name': 'Alex', 'Salary': 48000}]
Sorted list by Salary:[{'Name': 'John', 'Salary': 45000}, {'Name': 'Alex', 'Salary': 48000}, {'Name': 'Mary', 'Salary': 52000}]
```

AIM 6 - Write a program that reads a text file and counts the number of lines, words, and characters.

python_practical_6.ipynb X

C:\> Users\sumit\Desktop> MCA_1st_sem> SUMIT_KUMAR> python> python_practical_6.ipynb> def count_line():

Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...

▶

```
def count_line():
    line_count=0
    words_count=0
    characters_count=0

    while True:
        with open("sample.txt",'r') as fobj:
            if line_count == 0:
                print("Sample text file(sample.txt)")
                for iterate in fobj.readlines():
                    print(iterate,end='')
                    line_count+=1
                print(f"\nLines:{line_count}")
            elif words_count == 0:
                for iterate in fobj.readlines():
                    words_count+=len(iterate.split())
                print(f'Words:{words_count}')
            elif characters_count == 0:
                for iterate in list(fobj.read()):
                    if iterate != ' ' and iterate != '\n':
                        characters_count+=1
                print(f'characters:{characters_count}')
            else:
                break

count_line()
```

[5] ✓ 0.0s

```
''' Sample text file(sample.txt)
Sample text file(sample.txt)
Lines:1
Words:3
characters:26
```

AIM 7 - Read a sentence and display how many times each word appears, ignoring case and punctuation.

python_practical_7.ipynb X

C:\Users> sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > python > python_practical_7.ipynb > import string

Generate + Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ...



```
import string

def word_frequency(sentence):
    sentence = sentence.lower()

    for ch in string.punctuation:
        sentence = sentence.replace(ch, "")

    words = sentence.split()

    word_count = {}
    for word in words:
        word_count[word] = word_count.get(word, 0) + 1

    print("\nWord Frequency:")
    for word, count in word_count.items():
        print(f"{word} : {count}")

# Direct value (Jupyter-friendly)
sentence = "Python is easy, and Python is powerful!"

word_frequency(sentence)
```

[5] ✓ 0.0s

...

Word Frequency:

python : 2

is : 2

easy : 1

and : 1

powerful : 1

AIM 8 - Write a Python program that lists all files in a directory and categorizes them by file extension.

python_practical_8.ipynb X

C: > Users > sumit > Desktop > MCA_1st_sem > SUMIT_KUMAR > python > python_practical_8.ipynb > import os

Generate + Code + Markdown | Run All Restart Clear All Outputs Jupyter Variables Outline ...

```
import os
from collections import defaultdict

def categorize_files(directory):
    file_types = defaultdict(list) # Dictionary to hold extensions and file names

    # List all files in the directory
    for filename in os.listdir(directory):
        if os.path.isfile(os.path.join(directory, filename)):
            # Split filename and extension
            name, ext = os.path.splitext(filename)
            ext = ext.lower() if ext else "No Extension"
            file_types[ext].append(filename)

    # Display categorized files
    for ext, files in file_types.items():
        print(f"\n{ext}:")
        for f in files:
            print(f"    {f}")

# Set directory path (current directory in this case)
directory_path = "."
categorize_files(directory_path)
```

[1] ✓ 0.0s

...

.ipynb:

- python_practical_1.ipynb
- python_practical_2.ipynb
- python_practical_3.ipynb
- python_practical_4.ipynb
- python_practical_5.ipynb
- python_practical_6.ipynb
- python_practical_7.ipynb
- python_practical_8.ipynb

.txt:

- sample.txt