

INDEX

COURSE OUTCOME-1

SL-NO		P.NO
1.	Display future leap years from current year to a final year entered by user.	5
2.	List comprehensions: (a) Generate positive list of numbers from a given list of integers (b) Square of N numbers (c) Form a list of vowels selected from a given word (d) List ordinal value of each element of a word (Hint: use ord() to get ordinal values)	6
3.	Count the occurrences of each word in a line of text.	8
4.	Prompt the user for a list of integers. For all values greater than 100, store 'over' instead.	9
5.	Store a list of first names. Count the occurrences of 'a' within the list	11
6.	Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both	12
7.	Get a string from an input string where all occurrences of first character replaced with '\$', except first character. [eg: onion -> oni\$n]	14
8.	Create a string from given string where first and last characters exchanged. [eg: python > nythop]	15
9.	Accept the radius from user and find area of circle.	16
10.	Find biggest of 3 numbers entered.	17
11.	Accept a file name from user and print extension of that.	18
12.	Create a list of colors from comma-separated color names entered by user. Display first and last colors.	19
13.	Accept an integer n and compute n+nn+nnn.	20
14.	Print out all colors from color-list1 not contained in color-list2.	21
15.	Create a single string separated with space from two strings by swapping the character at position 1.	22

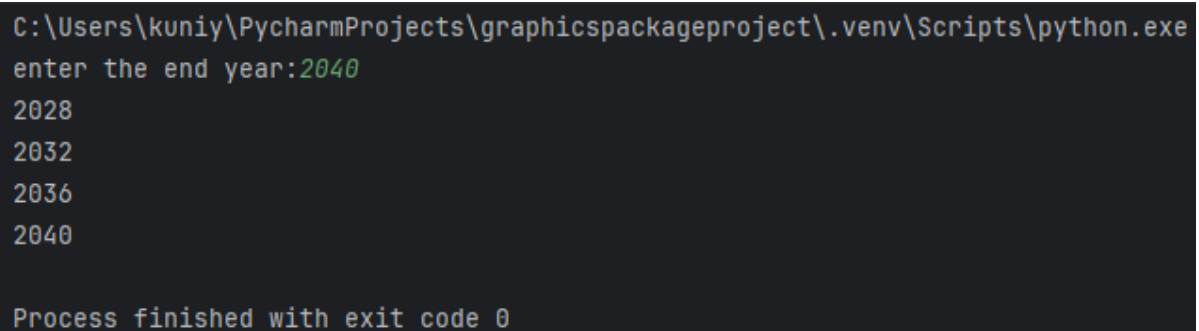
16.	Sort dictionary in ascending and descending order.	23
17.	Merge two dictionaries.	24
18.	Find gcd of 2 numbers.	25
19.	From a list of integers, create a list removing even numbers.	26
COURSE OUTCOME-2		
1.	Program to find the factorial of a number.	28
2.	Generate Fibonacci series of N terms.	29
3.	Find the sum of all items in a list.	30
4.	Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.	31
5.	Display the given pyramid with step number accepted from user. Eg: N=4 1 2 4 3 6 9 4 8 12 16	32
6.	Count the number of characters (character frequency) in a string.	34
7.	Add 'ing' at the end of a given string. If it already ends with 'ing', then add 'ly'	35
8.	Accept a list of words and return length of longest word.	36
9.	Construct following pattern using nested loop *	37
10.	Generate all factors of a number.	39
11.	Write lambda functions to find area of square, rectangle and triangle.	40
COURSE OUTCOME-3		
1.	Work with built-in packages.	43
2.	Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by	47

	different importing statements. (Include selective import of modules and import * statements)	
COURSE OUTCOME-4		
1	Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.	54
2.	Create a Bank account with members account number, name, type of account and balance. Write constructor and methods to deposit at the bank and withdraw an amount from the bank.	57
3.	Create a class Rectangle with private attributes length and width. Overload '<' operator to compare the area of 2 rectangles.	60
4.	Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.	62
5.	Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no of pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.	64
COURSE OUTCOME-5		
1.	Write a Python program to read a file line by line and store it into a list.	68
2.	Python program to copy odd lines of one file to other	70
3.	Write a Python program to read each row from a given csv file and print a list of strings.	72
4.	Python program to read specific columns of a given CSV file and print the content of the columns.	74
5.	Write a Python program to write a Python dictionary to a csv file. After writing the CSV file read the CSV file and display the content.	76

COURSE OUTCOME-1

1.Display future leap years from current year to a final year entered by user.

```
import datetime
x=datetime.datetime.now()
end=int(input("enter the end year:"))
for i in range(x.year, end+1):
    if(i%4==0 and i%100!=0) or (i%400==0):
        print(i)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\graphicspackageproject\.venv\Scripts\python.exe
enter the end year:2040
2028
2032
2036
2040

Process finished with exit code 0
```

ALGORITHM:

- Step 1: Import the datetime module using import datetime.
- Step 2: Get the current year using datetime.datetime.now().year.
- Step 3: Take user input for the end year and convert it to an integer.
- Step 4: Loop from the current year to the end year (inclusive).
- Step 5: Check if the current year in the loop is a leap year using the condition:
(year % 4 == 0 and year % 100 != 0) or (year % 400 == 0).
- Step 6: Print the year if it is a leap year.

2. List comprehensions:

(a) Generate positive list of numbers from a given list of integers

(b) Square of N numbers

(c) Form a list of vowels selected from a given word

(d) List ordinal value of each element of a word (Hint: use ord() to get ordinal values)

```
a) list=[-1,-2,-3,9, 5 ,2 ,7]
newlist=[x for x in list if x>0]
print(newlist)
```

```
b)l2=[2,3,4,5,6,7]
nl2=[x*x for x in l2]
print(nl2)
```

```
c)l3='rumaisa'
vowels=['a','e','i','o','u']
nl3=[x for x in l3 if x in vowels]
print(nl3)
```

```
d)l4='rumaisa'
nl4=[ord(x) for x in l4 ]
print(nl4)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\kuniy\PycharmProjects\pythonProject\list.py
[9, 5, 2, 7]
[4, 9, 16, 25, 36, 49]
['u', 'a', 'i', 'a']
[114, 117, 109, 97, 105, 97]

Process finished with exit code 0
```

ALGORITHM:

a)

Step 1: Create a list `list` with both positive and negative numbers.

Step 2: Use list comprehension to create a new list `newlist` that includes only the positive numbers from `list`.

Step 3: Print `newlist`.

b)

Step 1: Create a list `l2` with a series of integers.

Step 2: Use list comprehension to create a new list `nl2` where each element is the square of the corresponding element in `l2`.

Step 3: Print `nl2`.

c)

Step 1: Create a string `l3`.

Step 2: Define a list `vowels` containing the vowels 'a', 'e', 'i', 'o', 'u'.

Step 3: Use list comprehension to create a new list `nl3` containing only the characters from `l3` that are vowels.

Step 4: Print `nl3`.

d)

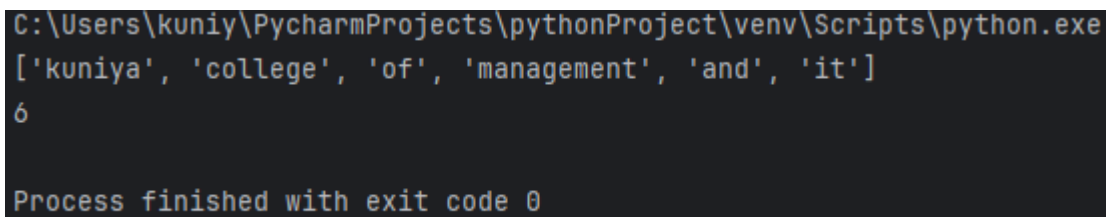
Step 1: Create a string `l4`.

Step 2: Use list comprehension to create a new list `nl4` where each element is the ASCII value of the corresponding character in `l4`.

Step 3: Print `nl4`.

3. Count the occurrences of each word in a line of text

```
text="kuniya college of management and it"  
words=text.split(" ")  
print(words)  
print(len(words))
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
['kuniya', 'college', 'of', 'management', 'and', 'it']  
6  
  
Process finished with exit code 0
```

ALGORITHM:

step1: Define a string text containing the sentence.

step2: Split the string text into individual words using split().

step3: Print the list of words.

step4: Calculate and print the number of words using len().

4. Prompt the user for a list of integers. For all values greater than 100, store 'over' instead.

```
li=[]  
for i in range(10):  
    a=int(input("enter integer: "))  
    if a>=100:  
        li.append("over")  
    else:  
        li.append(a)  
print(li)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe C  
enter integer: 1  
enter integer: 2  
enter integer: 3  
enter integer: 105  
enter integer: 445  
enter integer: 4  
enter integer: 668  
enter integer: 14  
enter integer: 768669  
enter integer: 6543  
[1, 2, 3, 'over', 'over', 4, 'over', 14, 'over', 'over']  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1: Initialize an empty list `li = []`.

Step 2: Use a for loop to iterate 10 times.

Step 3: In each iteration, prompt the user to input an integer using `input()` and convert it to an integer with `int()`.

Step 4: Check if the input integer `a` is greater than or equal to 100:

 If true, append the string "over" to the list `li`.

 Otherwise, append the integer `a` to the list `li`.

Step 5: After the loop finishes, print the list `li`.

5.Store a list of first names. Count the occurrences of 'a' within the list

```
names=['rumaisa','abhi','hasna']
```

```
for x in names:
```

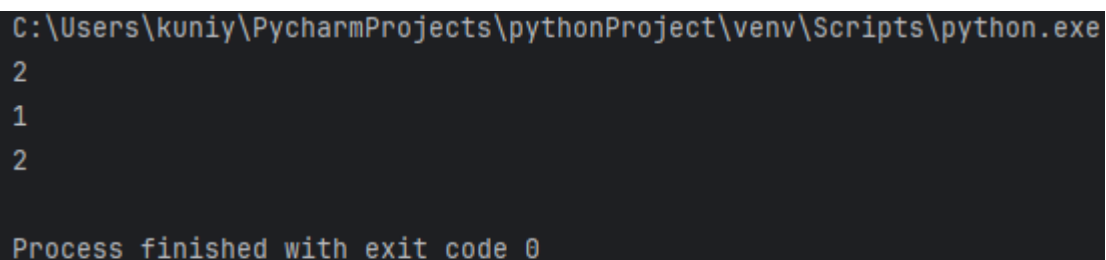
```
    s = 0
```

```
    for i in x:
```

```
        if i=='a':
```

```
            s+=1
```

```
    print(s)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
2
1
2

Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the list names.

Step 2: Start a loop over each name in names.

Step 3: Initialize a counter s = 0 to count occurrences of 'a'.

Step 4: Start a loop over each character in the current name.

Step 5: If the character is 'a', increment the counter s.

Step 6: After finishing the inner loop, print the value of s.

Step 7: Move to the next name in the outer loop (goto Step 2).

Step 8: End when all names are processed.

6. Enter 2 lists of integers. Check (a) Whether list are of same length (b) whether list sums to same value (c) whether any value occur in both.

```
l1=[1,5,9,22,66,99]
```

```
l2=[200,991,1200,661,567,3]
```

```
samelength=len(l1)==len(l2)
```

```
print("do both list have same length? :",samelength)
```

```
sumvalues=sum(l1)==sum(l2)
```

```
print("do both list have same sumvalues? :",sumvalues)
```

```
commonvalues=set(l1)&set(l2)
```

```
if commonvalues:
```

```
    print("list have common values:",commonvalues)
```

```
else:
```

```
    print("list have no common values")
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
do both list have same length? : True
do both list have same sumvalues? : False
list have no common values

Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the two lists.

Step 2: Check if both lists have the same length.

Step 3: Print whether both lists have the same length.

Step 4: Check if both lists have the same sum of values.

Step 5: Print whether both lists have the same sum of values.

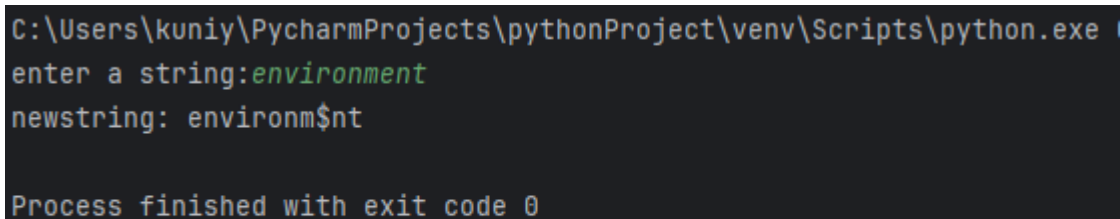
Step 6: Find the common values between the two lists.

Step 7: Print the common values or indicate if there are no common values.

7. Get a string from an input string where all occurrences of first character replaced with '\$', except first character. [eg: onion -> oni\$n]

```
string=input("enter a string:")  
newstring=string[0]+string[1:].replace(string[0],'$')  
print("newstring:",newstring)
```

OUTPUT:



```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
enter a string:environment  
newstring: environm$t  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1:Take user input for a string.

Step 2:Create a new string by concatenating the first character of the string with the rest of the string, replacing all occurrences of the first character with \$.

Step 3:Print the new string.

8. Create a string from given string where first and last characters exchanged. [eg: python > nythop]

```
s=input("enter a string:")  
if len(s)>1:  
    s=s[-1]+s[1:-1]+s[0]  
print(s)
```

OUTPUT:



```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
enter a string:python  
nythop  
  
Process finished with exit code 0
```

ALGORITHM:

- Step 1: Take user input for a string s.
- Step 2: Check if the length of the string is greater than 1.
- Step 3: If true, swap the first and last characters of the string.
- Step 4: Print the modified string s.

9. Accept the radius from user and find area of circle.

```
r=float(input("enter radius:"))  
area=3.14*r**2  
print("area of circle:",area)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
enter radius:5  
area of circle: 78.5  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1: Take user input for the radius r.

Step 2: Calculate the area of the circle using the formula $3.14 * r^2$.

Step 3: Print the calculated area of the circle.

10. Find biggest of 3 numbers entered.

```
a=int(input("enter a no:"))
b=int(input("enter a no:"))
c=int(input("enter a no:"))
if(a>b)and(a>c):
    largest=a
elif(b>a)and(b>c):
    largest=b
else:
    largest = c
print("largest no is:",largest)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter a no:77
enter a no:99
enter a no:33
largest no is: 99

Process finished with exit code 0
```

ALGORITHM:

Step 1: Take user input for three numbers a, b, and c.

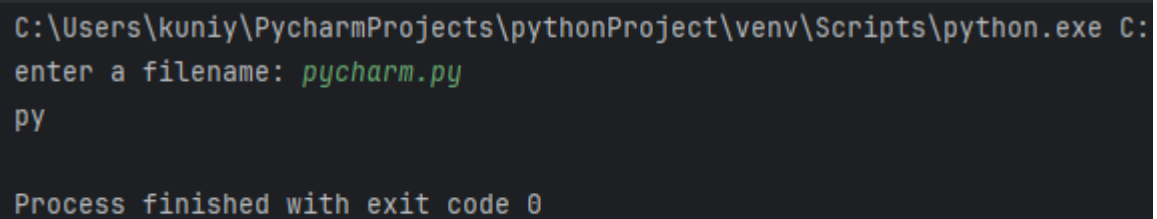
Step 2: Check if a is greater than both b and c. If true, set largest = a.

Step 3: If the previous condition is false, check if b is greater than both a and c. If true, set largest = b.

Step 4: If neither of the above conditions is true, set largest = c.

11. Accept a file name from user and print extension of that.

```
filename=input("enter a filename:")  
extension=filename.split(".")[1]  
print(extension)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe C:  
enter a filename: pycharm.py  
py  
  
Process finished with exit code 0
```

ALGORITHM:


Step 1: Take user input for the filename.

Step 2: Split the filename by the period (.) and retrieve the last part using split(".")[1].

Step 3: Print the file extension.

12. Create a list of colors from comma-separated color names entered by user.**Display first and last colors.**

```
colour=input("enter colours:")
colours=colour.split(",")
print(colours)
print("first colour:",colours[0])
print("last colour:",colours[-1])
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe C:
enter colours:yellow, green, red, blue, grey, black, white
['yellow', ' green', ' red', ' blue', ' grey', ' black', ' white']
first colour: yellow
last colour:  white

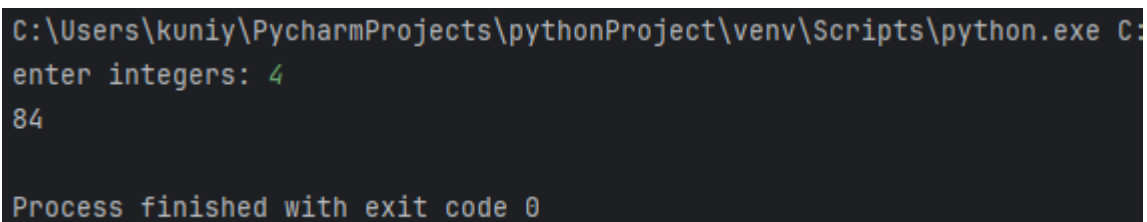
Process finished with exit code 0
```

ALGORITHM:

- Step 1: Take user input for the colours, separated by commas.
- Step 2: Split the input string into a list colours using the split(",") method.
- Step 3: Print the list of colours.
- Step 4: Print the first colour in the list.
- Step 5: Print the last colour in the list.
- Step 6: End the program after printing the first and last colours.

13. Accept an integer n and compute $n+nn+nnn$.

```
n=int(input("enter integers:"))  
sum=0  
sum=n+(n*n)+(n*n*n)  
print(sum)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe C:  
enter integers: 4  
84  
  
Process finished with exit code 0
```

ALGORITHM:

- Step 1: Take user input for the integer n.
- Step 2: Initialize a variable sum to 0.
- Step 3: Calculate the sum as $n + (n * n) + (n * n * n)$.
- Step 4: Print the calculated sum.

14. Print out all colors from color-list1 not contained in color-list2.

```
l1=['yellow','green','white']  
l2=['blue','green','red']  
color=[color for color in l1 if color not in l2]  
print("colors in l1 not in l2:",color)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python  
colors in l1 not in l2: ['yellow', 'white']  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the list l1 with colors.

Step 2: Define the list l2 with colors.

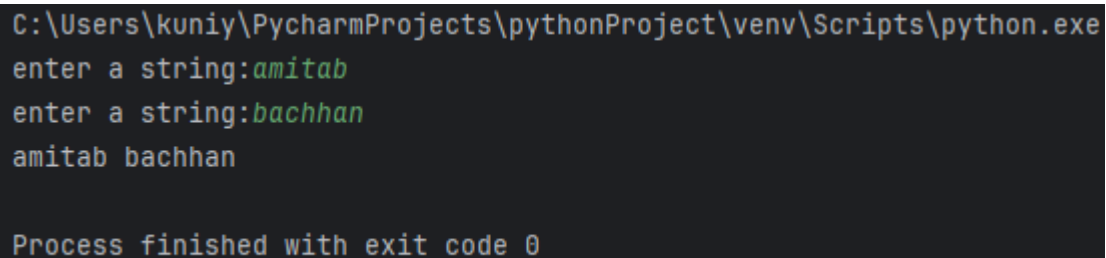
Step 3: Use list comprehension to create a new list color that contains elements from l1 that are not in l2.

Step 4: Print the resulting list color, which contains colors from l1 that are not in l2.

15. Create a single string separated with space from two strings by swapping the character at position 1.

```
s1=input("enter a string:")  
s2=input("enter a string:")  
s3=s1+" "+s2  
print(s3)
```

OUTPUT:



```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
enter a string:amitab  
enter a string:bachhan  
amitab bachhan  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1: Take user input for the first string s1.

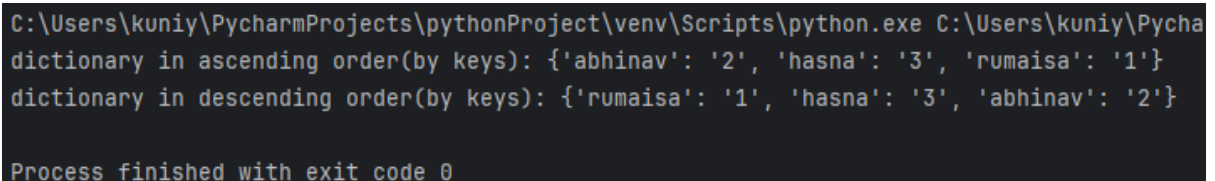
Step 2: Take user input for the second string s2.

Step 3: Concatenate s1 and s2 with a space in between and assign the result to s3.

Step 4: Print the concatenated string s3.

16. Sort dictionary in ascending and descending order.

```
a={'rumaisa':'1','abhinav':'2','hasna':'3'}  
ascending=dict(sorted(a.items()))  
print("dictionary in ascending order(by keys):",ascending)  
descending=dict(sorted(a.items(),reverse=True))  
print("dictionary in descending order(by keys):",descending)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\Users\kuniy\PycharmProjects\pythonProject\pythonProject\16.py  
dictionary in ascending order(by keys): {'abhinav': '2', 'hasna': '3', 'rumaisa': '1'}  
dictionary in descending order(by keys): {'rumaisa': '1', 'hasna': '3', 'abhinav': '2'}  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the dictionary a with key-value pairs.

Step 2: Sort the dictionary a in ascending order by keys using sorted(a.items()) and convert it back to a dictionary.

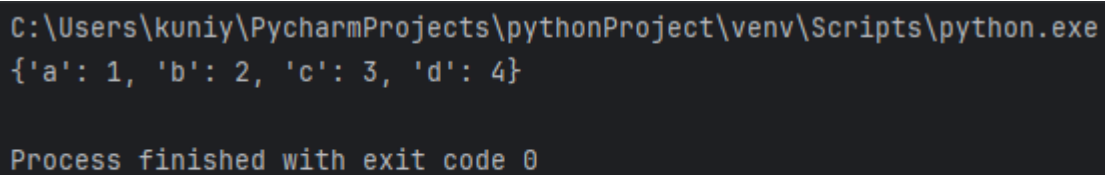
Step 3: Print the dictionary sorted in ascending order.

Step 4: Sort the dictionary a in descending order by keys using sorted(a.items(), reverse=True) and convert it back to a dictionary.

Step 5: Print the dictionary sorted in descending order.

17. Merge two dictionaries.

```
a={'a':1,'b':2,'c':3}
b={'b':2,'c':3,'d':4}
a.update(b)
print(a)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
{'a': 1, 'b': 2, 'c': 3, 'd': 4}

Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the dictionary a with key-value pairs.


Step 2: Define the dictionary b with key-value pairs.

Step 3: Update dictionary a with the items from dictionary b using a.update(b).

Step 4: Print the updated dictionary a.

18. Find gcd of 2 numbers.

```
def gcd(a,b):  
    while b != 0:  
        a,b=b,a%b  
    return a  
  
a=56  
b=98  
  
gcdvalue=gcd(a,b)  
print(gcdvalue)
```

OUTPUT:

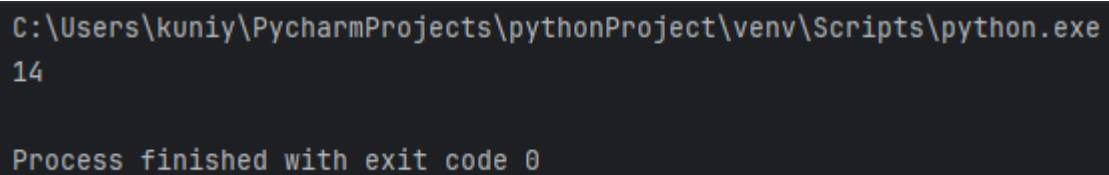
```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
14  
  
Process finished with exit code 0
```

ALGORITHM:

- Step 1: Define the function gcd(a, b) to calculate the greatest common divisor.
- Step 2: Inside the function, use a while loop to continue until b becomes zero. In each iteration, update a to the value of b and b to a % b.
- Step 3: Once the loop finishes, return the value of a, which is the greatest common divisor.
- Step 4: Assign values a = 56 and b = 98.
- Step 5: Call the gcd function with a and b as arguments and store the result in gcdvalue.
- Step 6: Print the value of gcdvalue.

19. From a list of integers, create a list removing even numbers.

```
n=[1,2,3,4,5,6,7,8,9]
odd=[x for x in n if x%2!=0]
print(odd)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
14

Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the list n with elements [1, 2, 3, 4, 5, 6, 7, 8, 9].

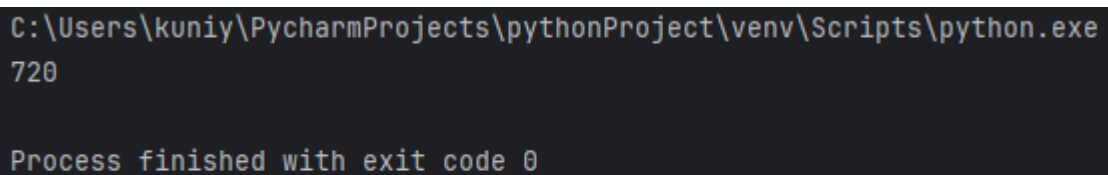
Step 2: Use list comprehension to create a new list odd that contains only the odd numbers from n.

Step 3: Print the list odd containing the odd numbers.

COURSE OUTCOME -2

1.Program to find the factorial of a number

```
f=1
for I in range(1,7):
    f*=i
print(f)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
720

Process finished with exit code 0
```

ALGORITHM:

- Step 1: Initialize $f = 1$ to store the product.
- Step 2: Use a for loop to iterate through the range from 1 to 6.
- Step 3: Multiply f by i (current loop variable) in each iteration.
- Step 4: After the loop finishes, print the value of f .

2.Generate Fibonacci series of N terms

```
n=int(input("enter a no:"))
a,b=0,1
print("fibonaaci series:")
for i in range(n):
    print(a)
    a,b=b,a+b
```

OUTPUT:



```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter a no:5
fibonaaci series:
0
1
1
2
3

Process finished with exit code 0
```

ALGORITHM:

Step 1: Take user input for the number n to specify how many Fibonacci numbers to generate.

Step 2: Initialize the first two Fibonacci numbers a = 0 and b = 1.

Step 3: Print "Fibonacci series:" to indicate the start of the series.


Step 4: Use a for loop to iterate n times.

Step 5: In each iteration, print the current value of a, then update a and b with the next Fibonacci numbers (a = b and b = a + b).

Step 6: End the loop after n Fibonacci numbers are printed.

3.Find the sum of all items in a list

```
a=[2,4,6,8,10]  
result=sum(a)  
print(result)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
30  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the list a with the elements [2, 4, 6, 8, 10].

Step 2: Use the sum() function to calculate the sum of the elements in the list a and assign it to the variable result.

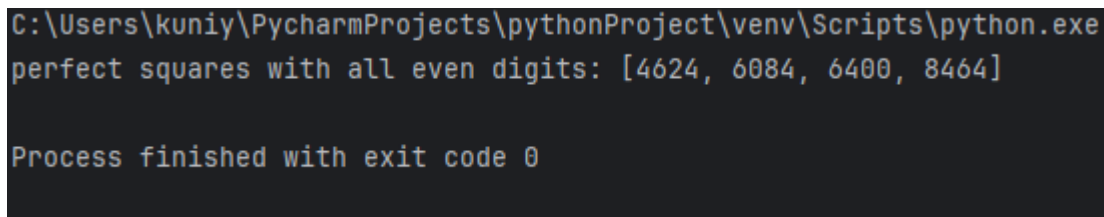
Step 3: Print the value of result, which contains the sum of the list elements.

4. Generate a list of four digit numbers in a given range with all their digits even and the number is a perfect square.

```
lower=1000
upper=9999
result=[]
for i in range(32,100):
    square=i*i
    if square >= lower and square <= upper:
        if all(int(digit)%2==0 for digit in str(square)):
            result.append(square)

print("perfect squares with all even digits:",result)
```

OUTPUT:



```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
perfect squares with all even digits: [4624, 6084, 6400, 8464]

Process finished with exit code 0
```

ALGORITHM:

Step 1: Define the range for the lower and upper bounds (lower = 1000, upper = 9999) for the 4-digit perfect squares.

Step 2: Initialize an empty list result to store the perfect squares that meet the criteria.

Step 3: Use a for loop to iterate over the numbers from 32 to 99 (since $100^2 = 10000$).

Step 4: Calculate the square of the current number i and store it in square.

Step 5: Check if the square is within the range of lower and upper (inclusive).

Step 6: For the valid squares, check if all digits of the square are even using all() and a generator expression. Convert the square to a string and check each digit.

Step 7: If all digits are even, append the square to the result list.

Step 8: After the loop finishes, print the list result containing the perfect squares with all even digits.

5. Display the given pyramid with step number accepted from user.**Eg: N=4**

1
2 4
3 6 9
4 8 12 16

```
n=int(input("enter no of steps:"))
```

```
for i in range(1,n+1):
```

```
    for j in range(1,i+1):
```

```
        print(i*j,end=" ")
```

```
    print()
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter no of steps:5
1
2 4
3 6 9
4 8 12 16
5 10 15 20 25

Process finished with exit code 0
```


ALGORITHM:

Step 1: Define the range for the lower and upper bounds (lower = 1000, upper = 9999) for the 4-digit perfect squares.

Step 2: Initialize an empty list result to store the perfect squares that meet the criteria.

Step 3: Use a for loop to iterate over the numbers from 32 to 99 (since $100^2 = 10000$).

Step 4: Calculate the square of the current number i and store it in square.

Step 5: Check if the square is within the range of lower and upper (inclusive).

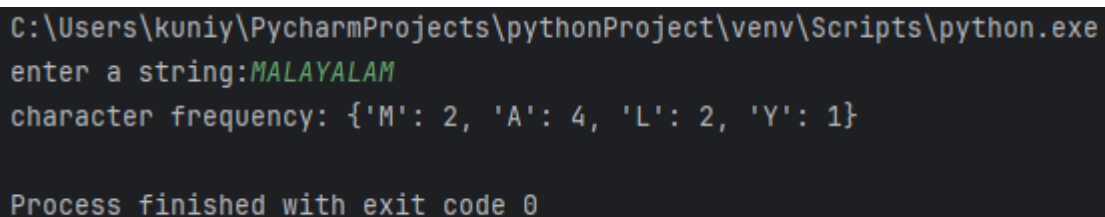
Step 6: For the valid squares, check if all digits of the square are even using all() and a generator expression. Convert the square to a string and check each digit.

Step 7: If all digits are even, append the square to the result list.

Step 8: After the loop finishes, print the list result containing the perfect squares with all even digits.

6. Count the number of characters (character frequency) in a string.

```
n=input("enter a string:")
count={ }
for char in n:
    if char in count:
        count[char]+=1
    else:
        count[char]=1
print("character frequency:",count)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter a string:MALAYALAM
character frequency: {'M': 2, 'A': 4, 'L': 2, 'Y': 1}

Process finished with exit code 0
```

ALGORITHM:

- Step 1: Take user input for the string n.
- Step 2: Initialize an empty dictionary count to store the frequency of each character.
- Step 3: Loop through each character char in the string n.
- Step 4: If the character char is already in the dictionary count, increment its value by 1.
- Step 5: If the character char is not in the dictionary, add it with a value of 1.
- Step 6: After the loop finishes, print the dictionary count, which contains the frequency of each character.

7. Add 'ing' at the end of a given string. If it already ends with 'ing', then add 'ly'

```
n=input("enter a string:")
if n.endswith('ing'):
    n+='ly'
else:
    n+='ing'
print("modified string:",n)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter a string:play
modified string: playing

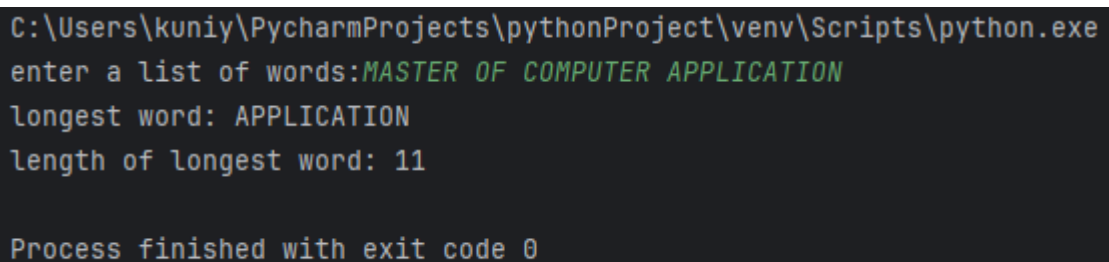
Process finished with exit code 0
```

ALGORITHM:

- Step 1: Take user input for the string n.
- Step 2: Check if the string n ends with the substring 'ing' using the endswith() method.
- Step 3: If the string ends with 'ing', append 'ly' to the string n.
- Step 4: If the string does not end with 'ing', append 'ing' to the string n.
- Step 5: Print the modified string n.

8. Accept a list of words and return length of longest word.

```
words=input("enter a list of words:").split()
longest=max(words,key=len)
print("longest word:",longest)
print("length of longest word:",len(longest))
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter a list of words:MASTER OF COMPUTER APPLICATION
longest word: APPLICATION
length of longest word: 11

Process finished with exit code 0
```

ALGORITHM:

- Step 1: Take user input for a list of words, split by spaces, and store them in the list words.
- Step 2: Use the max() function with key=len to find the longest word in the list words.
- Step 3: Print the longest word found.
- Step 4: Print the length of the longest word using the len() function.

9. Construct following pattern using nested loop

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

```
n=5
```

```
for i in range(1,n+1):
```

```
    print("*"* i)
```

```
for i in range(n-1,0,-1):
```

```
    print("*"* i)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
*  
**  
***  
****  
*****  
*****  
****  
***  
**  
*  
  
Process finished with exit code 0
```

ALGORITHM:

Step 1: Set the value of $n = 5$.

Step 2: Start a loop for i from 1 to $n + 1$ (inclusive).

Step 3: In each iteration, print i asterisks (*).

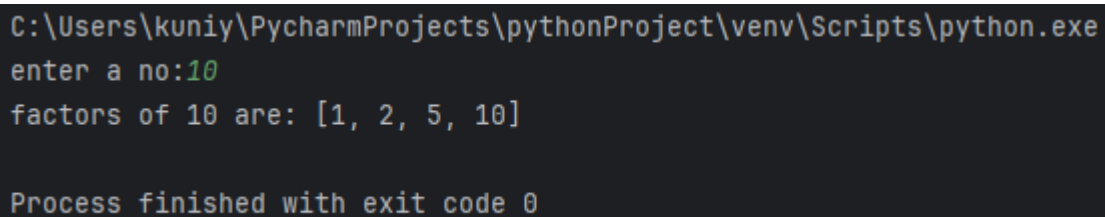
Step 4: After finishing the first loop, start a second loop for i from $n - 1$ to 1, decrementing by 1 each time ($\text{range}(n-1, 0, -1)$).

Step 5: In each iteration of the second loop, print i asterisks (*).

Step 6: End the program after completing both loops.

10. Generate all factors of a number.

```
n=int(input("enter a no:"))
factors=[ ]
for i in range(1,n+1):
    if n%i ==0:
        factors.append(i)
print("factors of",n,"are:",factors)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter a no:10
factors of 10 are: [1, 2, 5, 10]

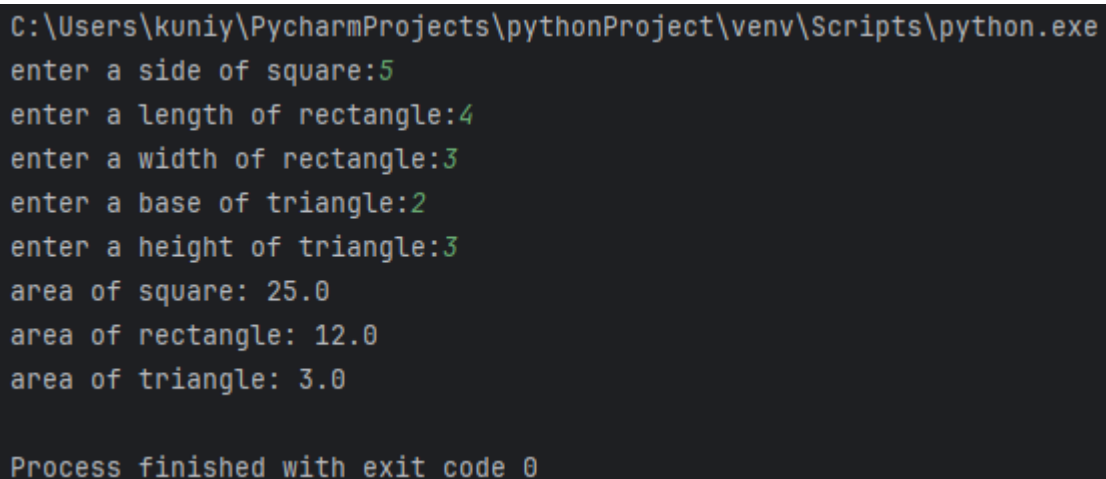
Process finished with exit code 0
```

ALGORITHM:

- Step 1: Take user input for the integer n.
- Step 2: Initialize an empty list factors to store the factors of n.
- Step 3: Use a for loop to iterate through the range from 1 to n + 1.
- Step 4: In each iteration, check if n % i == 0. If true, append i to the list factors.
- Step 5: After the loop finishes, print the list of factors for n.
- Step 6: End the program after printing the factors.

11. Write lambda functions to find area of square, rectangle and triangle

```
square=lambda side:side*side
rectangle=lambda length,width:length*width
triangle=lambda base,height:(base*height)/2
side=float(input("enter a side of square:"))
length=float(input("enter a length of rectangle:"))
width=float(input("enter a width of rectangle:"))
base=float(input("enter a base of triangle:"))
height=float(input("enter a height of triangle:"))
print("area of square:",square(side))
print("area of rectangle:",rectangle(length,width))
print("area of triangle:",triangle(base,height))
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
enter a side of square:5
enter a length of rectangle:4
enter a width of rectangle:3
enter a base of triangle:2
enter a height of triangle:3
area of square: 25.0
area of rectangle: 12.0
area of triangle: 3.0

Process finished with exit code 0
```


ALGORITHM:

Step 1: Define a lambda function square to calculate the area of a square (side * side).

Step 2: Define a lambda function rectangle to calculate the area of a rectangle (length * width).

Step 3: Define a lambda function triangle to calculate the area of a triangle ((base * height) / 2).

Step 4: Take user input for the side of the square, length and width of the rectangle, and base and height of the triangle.

Step 5: Calculate and print the area of the square using the square lambda function.

Step 6: Calculate and print the area of the rectangle using the rectangle lambda function.

Step 7: Calculate and print the area of the triangle using the triangle lambda function.

COURSE OUTCOME -3

1. Work with built-in packages

```
import math
print(math.sqrt(9))
print(math.ceil(4.2))
print(math.floor(4.8), "\n")
```

```
import random
print(random.randint(1,5))
name=['abhi','rumi','asna']
print(random.choice(name), "\n")
```

```
from datetime import datetime
now=datetime.now()
print(now)
print(now.strftime("%d/%m/%Y"), "\n")
```

```
import os
print(os.getcwd())
print(os.listdir('.'), "\n")
```

```
import json
data={"name":"alice", "age":25}
jsondata=json.dumps(data)
print(jsondata)
pydata=json.loads(jsondata)
print(pydata['name'], "\n")
```

```
import time
```

```
print("start")
time.sleep(10)
print("end","\n")

import statistics
num=[10,20,30,40]
print(statistics.mean(num),"\n")
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
3.0
5
4

3
abhi

2025-01-03 14:27:52.518875
03/01/2025

C:\Users\kuniy\PycharmProjects\pythonProject
['$.py', '.idea', '2list.py', '2lists.py', '4digits.py', 'area.py',

{"name": "alice", "age": 25}
alice

start
end

25

Process finished with exit code 0
```

ALGORITHM:**Step 1: Using the math Library**

- 1.Import the math library.
- 2.Call `math.sqrt(9)` to calculate the square root of 9.
- 3.Print the result of `math.sqrt(9)`.
- 4.Call `math.ceil(4.2)` to round 4.2 to the nearest integer up.
- 5.Print the result of `math.ceil(4.2)`.
- 6.Call `math.floor(4.8)` to round 4.8 to the nearest integer down.
- 7.Print the result of `math.floor(4.8)`.

Step 2: Using the random Library

- 1.Import the random library.
- 2.Call `random.randint(1, 5)` to generate a random integer between 1 and 5.
- 3.Print the result of `random.randint(1, 5)`.
- 4.Define a list name = ['abhi', 'rumi', 'asna'].
- 5.Call `random.choice(name)` to choose a random name from the list.
- 6.Print the randomly chosen name from the list.

Step 3: Using the datetime Library

- 1.Import datetime from the datetime module.
- 2.Call `datetime.now()` to get the current date and time.
- 3.Print the current date and time.
- 4.Call `now.strftime("%d/%m/%Y")` to format the current date in day/month/year format.
- 5.Print the formatted current date.

Step 4: Using the os Library

- 1.Import the os library.
- 2.Call `os.getcwd()` to get the current working directory.
- 3.Print the current working directory.
- 4.Call `os.listdir('.')` to list all files and directories in the current directory.
- 5.Print the list of files and directories in the current directory.

Step 5: Using the json Library

- 1.Import the json library.
- 2.Define a dictionary data = {"name": "alice", "age": 25}.
- 3.Call json.dumps(data) to convert the dictionary to a JSON string.
- 4.Print the JSON string.
- 5.Call json.loads(jsondata) to convert the JSON string back to a Python dictionary.
- 6.Print the value of the key name in the dictionary.

Step 6: Using the time Library

- 1.Import the time library.
- 2.Print "start".
- 3.Call time.sleep(10) to pause the program for 10 seconds.
- 4.Print "end" after the pause.

Step 7: Using the statistics Library

- 1.Import the statistics library.
- 2.Define a list num = [10, 20, 30, 40].
- 3.Call statistics.mean(num) to calculate the mean of the list.
- 4.Print the result of statistics.mean(num).

2. Create a package graphics with modules rectangle, circle and sub-package 3D-graphics with modules cuboid and sphere. Include methods to find area and perimeter of respective figures in each module. Write programs that finds area and perimeter of figures by different importing statements.

(Include selective import of modules and import * statements)

cuboid.py

```
def surface_area(length,width,height):  
    return 2*(length*width+width*height+height*length)  
  
def volume(length,width,height):  
    return length*width*height
```

sphere.py

```
import math  
  
def surface_area(radius):  
    return 4*math.pi*radius**2  
  
def volume(radius):  
    return(4/3)*math.pi*radius**3
```

init.py

```
from .cuboid import surface_area as cuboid_surface_area,volume as cuboid_volume  
from .sphere import surface_area as sphere_surface_area,volume as sphere_volume
```

circle.py

```
import math  
  
def area(radius):  
    return math.pi * radius*radius  
  
def perimeter(radius):  
    return 2*(math.pi * radius)
```

rectangle.py

```
def area(length,width):  
    return length*width  
def perimeter(length,width):  
    return 2*(length+width)
```

init.py

```
from .rectangle import area as rectangle_area,perimeter as rectangle_perimeter  
from .circle import area as circle_area,perimeter as circle_perimeter
```

3dgraphics.py

```
from graphics.graphics_3d.cuboid import surface_area as cuboid_area,volume as  
cuboid_volume  
  
from graphics.graphics_3d.sphere import surface_area as sphere_area,volume as  
sphere_volume  
  
print("cuboid surface area:",cuboid_area(4,5,6))  
print("cuboid volume:",cuboid_volume(4,5,6))  
  
print("sphere surface area:",sphere_area(7))  
print("sphere volume:",sphere_volume(7))
```

test entire module.py

```
import graphics.rectangle  
import graphics.circle  
  
print("rectangle area:",graphics.rectangle.area(5,3))  
print("rectangle perimeter:",graphics.rectangle.perimeter(5,3))  
  
print("circle area:",graphics.circle.area(7))  
print("circle perimeter:",graphics.circle.perimeter(7))
```


test_import_all.py

```
from graphics.rectangle import *
```

```
print("rectangle area:",area(5,3))
```

```
print("rectangle perimeter:",perimeter(5,3))
```

test_selective_import.py

```
from graphics.rectangle import area as rect_area,perimeter as rect_perimeter
```

```
from graphics.circle import area as circle_area,perimeter as circle_perimeter
```

```
print("rectangle area:",rect_area(5,3))
```

```
print("rectangle perimeter:",rect_perimeter(5,3))
```

```
print("circle area:",circle_area(7))
```

```
print("circle perimeter:",circle_perimeter(7))
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\graphicspackageproject\.venv\Scripts\python.exe
cuboid surface area: 148
cuboid volume: 120
sphere surface area: 615.7521601035994
sphere volume: 1436.7550402417319

Process finished with exit code 0
```

```
C:\Users\kuniy\PycharmProjects\graphicspackageproject\.venv\Scripts\python.exe
rectangle area: 15
rectangle perimeter: 16
circle area: 153.93804002589985
circle perimeter: 43.982297150257104

Process finished with exit code 0
```

```
C:\Users\kuniy\PycharmProjects\graphicspackageproject\.venv\Scripts\python.exe
rectangle area: 15
rectangle perimeter: 16

Process finished with exit code 0
```

```
C:\Users\kuniy\PycharmProjects\graphicspackageproject\.venv\Scripts\python.exe
rectangle area: 15
rectangle perimeter: 16
circle area: 153.93804002589985
circle perimeter: 43.982297150257104

Process finished with exit code 0
```

ALGORITHM:

Cuboid (cuboid.py):

Step 1: Define surface_area(length, width, height).

Step 2: Define volume(length, width, height).

Sphere (sphere.py):

Step 1: Import math.

Step 2: Define surface_area(radius).

Step 3: Define volume(radius).

3D Graphics Module (__init__.py):

Step 1: Import functions from cuboid.py and sphere.py with aliasing.

Circle (circle.py):

Step 1: Import math.

Step 2: Define area(radius).

Step 3: Define perimeter(radius).

Rectangle (rectangle.py):

Step 1: Define area(length, width).

Step 2: Define perimeter(length, width).

2D Graphics Module (__init__.py):

Step 1: Import functions from rectangle.py and circle.py with aliasing.

Test 3D Graphics (3dgraphics.py):

Step 1: Import 3D functions.

Step 2: Print cuboid surface area and volume.

Step 3: Print sphere surface area and volume.

Test Entire Module (test_entire_module.py):

Step 1: Import 2D functions.

Step 2: Print rectangle area and perimeter.

Step 3: Print circle area and perimeter.

Test All Functions Import (test_import_all.py):

Step 1: Import all functions from graphics.rectangle.

Step 2: Print rectangle area and perimeter.

Test Selective Import (test_selective_import.py):

Step 1: Import selected functions from `graphics.rectangle` and `graphics.circle`.

Step 2: Print rectangle area and perimeter.

Step 3: Print circle area and perimeter.

COURSE OUTCOME -4

1. Create Rectangle class with attributes length and breadth and methods to find area and perimeter. Compare two Rectangle objects by their area.

Rectangle.py

```
def area(length,width):  
    return length*width  
def perimeter(length,width):  
    return 2*(length+width)
```

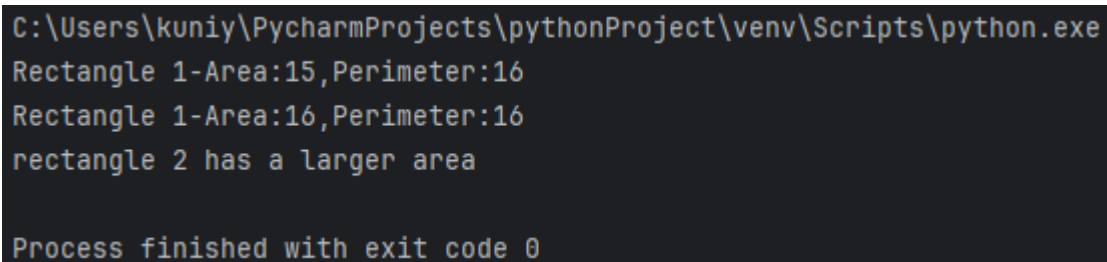
rect.py

```
class Rectangle:  
    def __init__(self,length,breadth):  
        self.length=length  
        self.breadth=breadth  
    def area(self):  
        return self.length*self.breadth  
    def perimeter(self):  
        return 2*(self.length+self.breadth)  
    def compare_area(self,other):  
        if self.area()>other.area():  
            return "rectangle 1 has a larger area"  
        elif self.area()<other.area():  
            return "rectangle 2 has a larger area"  
        else:  
            return "both rectangles has the same area"
```

Rectcomparison.py

```
from rect import Rectangle  
rect1=Rectangle(5,3)  
rect2=Rectangle(4,4)
```

```
print(f"Rectangle 1-Area:{rect1.area()},Perimeter:{rect1.perimeter()}")
print(f"Rectangle 1-Area:{rect2.area()},Perimeter:{rect2.perimeter()}")
result=rect1.compare_area(rect2)
print(result)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
Rectangle 1-Area:15,Perimeter:16
Rectangle 1-Area:16,Perimeter:16
rectangle 2 has a larger area

Process finished with exit code 0
```

ALGORITHM:

Rectangle.py:

Step 1: Define function area(length, width) to calculate area.

Step 2: Define function perimeter(length, width) to calculate perimeter.

Rect.py:

Step 1: Define class Rectangle.

Step 2: Define __init__(self, length, breadth) to initialize attributes length and breadth.

Step 3: Define method area(self) to calculate area of the rectangle.

Step 4: Define method perimeter(self) to calculate perimeter of the rectangle.

Step 5: Define method compare_area(self, other) to compare the areas of two rectangles:

If self.area() > other.area(), return "rectangle 1 has a larger area".

If self.area() < other.area(), return "rectangle 2 has a larger area".

Otherwise, return "both rectangles have the same area".

Rectcomparison.py:

Step 1: Import Rectangle class from rect.py.

Step 2: Create two instances of the Rectangle class, rect1 and rect2 with specified dimensions.

Step 3: Print the area and perimeter of rect1.

Step 4: Print the area and perimeter of rect2.

Step 5: Call rect1.compare_area(rect2) to compare areas of rect1 and rect2.

Step 6: Print the result of the comparison.

2. Create a Bank account with members account number, name, type of account and balance.

Write constructor and methods to deposit at the bank and withdraw an amount from the bank.

```
class BankAccount:
```

```
    def __init__(self, account_number, name, account_type, balance=0.0):
```

```
        self.account_number = account_number
```

```
        self.name = name
```

```
        self.account_type = account_type
```

```
        self.balance = balance
```

```
    def deposit(self, amount):
```

```
        if amount > 0:
```

```
            self.balance += amount
```

```
            print(f"₹{amount} deposited successfully. New balance: ₹{self.balance}")
```

```
        else:
```

```
            print("Deposit amount must be greater than 0.")
```

```
    def withdraw(self, amount):
```

```
        if amount > self.balance:
```

```
            print(f"Insufficient balance. Available balance: ₹{self.balance}")
```

```
        elif amount <= 0:
```

```
            print("Withdrawal amount must be greater than 0.")
```

```
        else:
```

```
            self.balance -= amount
```

```
            print(f"₹{amount} withdrawn successfully. Remaining balance: ₹{self.balance}")
```

```
    def display_account_details(self):
```

```
        print("\nAccount Details:")
```

```
        print(f"Account Number: {self.account_number}")
```

```
print(f"Account Holder: {self.name}")  
print(f"Account Type: {self.account_type}")  
print(f"Balance: ₹{self.balance}\n")
```

```
account = BankAccount(123456, "John Doe", "Savings", 5000)  
account.display_account_details()  
account.deposit(2000)  
account.withdraw(3000)  
account.withdraw(5000)  
account.display_account_details()
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
  
Account Details:  
Account Number: 123456  
Account Holder: John Doe  
Account Type: Savings  
Balance: ₹5000  
  
₹2000 deposited successfully. New balance: ₹7000  
₹3000 withdrawn successfully. Remaining balance: ₹4000  
Insufficient balance. Available balance: ₹4000  
  
Account Details:  
Account Number: 123456  
Account Holder: John Doe  
Account Type: Savings  
Balance: ₹4000  
  
Process finished with exit code 0
```

ALGORITHM:

Define BankAccount Class:

Step 1: Define BankAccount class.

Step 2: Define `__init__(self, account_number, name, account_type, balance=0.0)` constructor:

Initialize `account_number`, `name`, `account_type`, and `balance`.

Step 3: Define `deposit(self, amount)` method:

If `amount > 0`, add amount to balance and print success message.

Otherwise, print "Deposit amount must be greater than 0."

Step 4: Define `withdraw(self, amount)` method:

If `amount > balance`, print "Insufficient balance."

If `amount <= 0`, print "Withdrawal amount must be greater than 0."

Otherwise, deduct amount from balance and print success message.

Step 5: Define `display_account_details(self)` method:

Print account number, account holder name, account type, and current balance.

Main Program:

Step 1: Create an instance of BankAccount with account number 123456, account holder name John Doe, account type Savings, and initial balance 5000.

Step 2: Call `account.display_account_details()` to display initial account details.

Step 3: Call `account.deposit(2000)` to deposit ₹2000.

Step 4: Call `account.withdraw(3000)` to withdraw ₹3000.

Step 5: Call `account.withdraw(5000)` to withdraw ₹5000 (greater than current balance).

Step 6: Call `account.display_account_details()` again to display updated account details.

3. Create a class Rectangle with private attributes length and width. Overload '<' operator to compare the area of 2 rectangles.

```
class Rectangle:
    def __init__(self, length, width):
        self.__length = length
        self.__width = width

    def area(self):
        return self.__length * self.__width

    def __lt__(self, other):
        return self.area() < other.area()

rect1 = Rectangle(5, 3)
rect2 = Rectangle(4, 4)

if rect1 < rect2:
    print("Rectangle 1 has a smaller area than Rectangle 2")
else:
    print("Rectangle 1 has a larger or equal area compared to Rectangle 2")
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
Rectangle 1 has a smaller area than Rectangle 2

Process finished with exit code 0
```

ALGORITHM:

Define the Rectangle Class:

Step 1: Define class Rectangle.

Step 2: In the constructor (`__init__`), initialize the private attributes `__length` and `__width`.

Step 3: Define `area(self)` method to calculate and return the area of the rectangle:
`self.__length * self.__width`.

Step 4: Define `__lt__(self, other)` method for comparison:

If the area of self is smaller than the area of other, return True.

Otherwise, return False.

Main Program:

Step 1: Create an instance `rect1` of Rectangle with length 5 and width 3.

Step 2: Create an instance `rect2` of Rectangle with length 4 and width 4.

Step 3: Compare the areas of `rect1` and `rect2` using the `<` operator, which internally calls the `__lt__` method:

If `rect1.area()` is smaller than `rect2.area()`, print "Rectangle 1 has a smaller area than Rectangle 2".

Otherwise, print "Rectangle 1 has a larger or equal area compared to Rectangle 2".

4. Create a class Time with private attributes hour, minute and second. Overload '+' operator to find sum of 2 time.

```
class Time:
    def __init__(self, hour, minute, second):
        self.__hour = hour
        self.__minute = minute
        self.__second = second

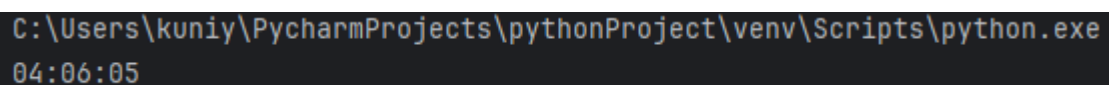
    def __add__(self, other):
        second = self.__second + other.__second
        minute = self.__minute + other.__minute + (second // 60)
        hour = self.__hour + other.__hour + (minute // 60)
        return Time(hour % 24, minute % 60, second % 60)

    def display(self):
        print(f"{self.__hour:02}:{self.__minute:02}:{self.__second:02}")

time1 = Time(2, 45, 50)
time2 = Time(1, 20, 15)

time3 = time1 + time2
time3.display()
```

OUTPUT:



```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
04:06:05
```

ALGORITHM:

Define the Time Class:

Step 1: Define the Time class with the `__init__(self, hour, minute, second)` constructor to initialize private attributes `__hour`, `__minute`, and `__second`.

Step 2: Define `__add__(self, other)` method to add two Time objects:

Add the seconds of both times and calculate the minutes and hours that may result from overflow.

Normalize seconds and minutes using modulo operation: `second % 60`, `minute % 60`, and `hour % 24`.

Return a new Time object with the resulting hours, minutes, and seconds.

Step 3: Define `display(self)` method to display the time in hh:mm:ss format with leading zeros if needed.

Main Program:

Step 1: Create an instance `time1` of Time with hour 2, minute 45, and second 50.

Step 2: Create an instance `time2` of Time with hour 1, minute 20, and second 15.

Step 3: Add the two Time objects (`time1 + time2`) which calls the `__add__()` method.

Step 4: Call `display()` method on the resulting Time object (`time3`) to show the final time in hh:mm:ss format.

5. Create a class Publisher (name). Derive class Book from Publisher with attributes title and author. Derive class Python from Book with attributes price and no_of_pages. Write a program that displays information about a Python book. Use base class constructor invocation and method overriding.

```
class Publisher:
```

```
    def __init__(self, name):
```

```
        self.name = name
```

```
    def display(self):
```

```
        print(f"Publisher: {self.name}")
```

```
class Book(Publisher):
```

```
    def __init__(self, name, title, author):
```

```
        super().__init__(name)
```

```
        self.title = title
```

```
        self.author = author
```

```
    def display(self):
```

```
        print(f"Publisher: {self.name}")
```

```
        print(f"Title: {self.title}")
```

```
        print(f"Author: {self.author}")
```

```
class Python(Book):
```

```
    def __init__(self, name, title, author, price, no_of_pages):
```

```
        super().__init__(name, title, author)
```

```
        self.price = price
```

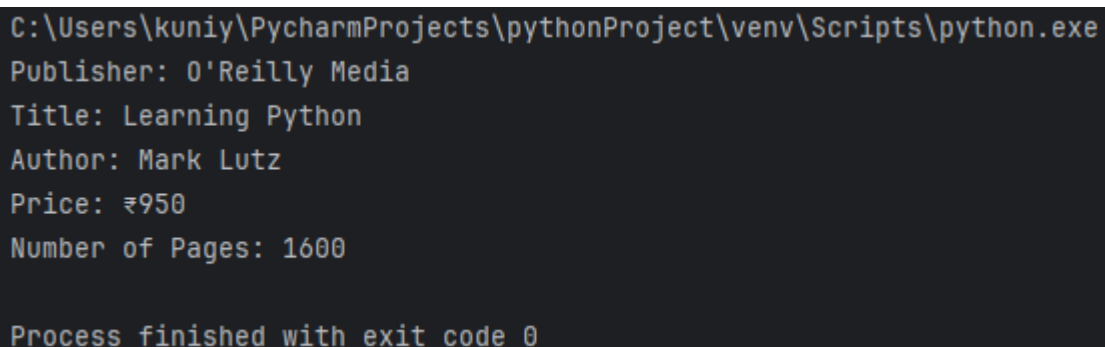
```
        self.no_of_pages = no_of_pages
```



```
def display(self):  
    print(f"Publisher: {self.name}")  
    print(f"Title: {self.title}")  
    print(f"Author: {self.author}")  
    print(f"Price: ₹{self.price}")  
    print(f"Number of Pages: {self.no_of_pages}")
```

```
python_book = Python("O'Reilly Media", "Learning Python", "Mark Lutz", 950, 1600)  
python_book.display()
```

OUTPUT:



```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe  
Publisher: O'Reilly Media  
Title: Learning Python  
Author: Mark Lutz  
Price: ₹950  
Number of Pages: 1600  
  
Process finished with exit code 0
```

ALGORITHM:

Define the Publisher Class:

Step 1: Define the Publisher class with the `__init__(self, name)` constructor to initialize the name attribute.

Step 2: Define the `display(self)` method to print the publisher's name.

Define the Book Class (Inherits Publisher):

Step 1: Define the Book class, inheriting from the Publisher class.

Step 2: Define the `__init__(self, name, title, author)` constructor to initialize the name, title, and author attributes. Call `super().__init__(name)` to initialize the name from the Publisher class.

Step 3: Override the `display(self)` method to print the publisher's name, title, and author.

Define the Python Class (Inherits Book):

Step 1: Define the Python class, inheriting from the Book class.

Step 2: Define the `__init__(self, name, title, author, price, no_of_pages)` constructor to initialize the price and no_of_pages attributes in addition to the inherited attributes.

Step 3: Override the `display(self)` method to print the publisher's name, title, author, price, and number of pages.

Main Program:

Step 1: Create an instance `python_book` of the Python class with the specified details.

Step 2: Call the `display()` method on the `python_book` object to print all details about the Python book.

COURSE OUTCOME -5

1. Write a Python program to read a file line by line and store it into a list

```
filename = "example.txt"
```

```
with open(filename, 'r') as file:
```

```
    lines = file.readlines()
```

```
for line in lines:
```

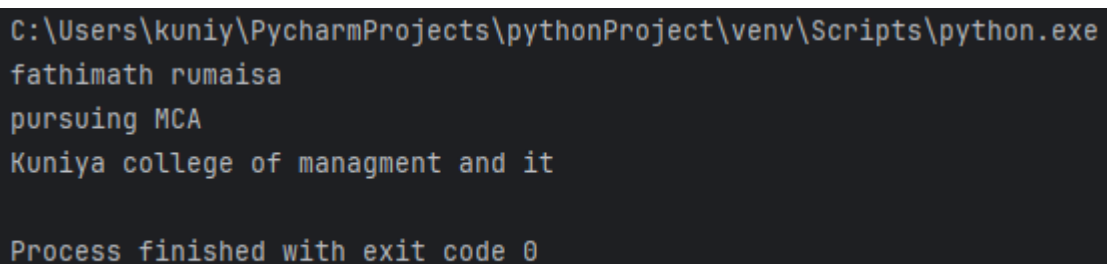
```
    print(line.strip())
```

example.txt

fathimath rumaisa

pursuing MCA

Kuniya college of managment and it

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
fathimath rumaisa
pursuing MCA
Kuniya college of managment and it

Process finished with exit code 0
```

ALGORITHM:

Define the Filename:

Step 1: Define the filename variable with the value "example.txt".

Open the File:

Step 1: Use the open() function with the 'r' mode to open the file in read-only mode.

Step 2: Use the with statement to ensure that the file is properly closed after reading.

Step 3: Use file.readlines() to read all lines from the file and store them in the lines list.

Process Each Line:

Step 1: Loop through each line in the lines list.

Step 2: Use strip() to remove any leading or trailing whitespace characters from the line.

Step 3: Print the stripped line.

File Content Example:

fathimath rumaisa

pursuing MCA

Kuniya college of managment and it

2. Python program to copy odd lines of one file to other

```
input_filename = "input.txt"
output_filename = "output.txt"

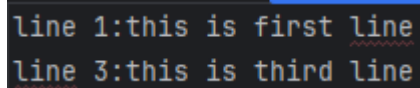
with open(input_filename, 'r') as input_file:
    lines = input_file.readlines()

with open(output_filename, 'w') as output_file:
    for index, line in enumerate(lines, start=1):
        if index % 2 != 0:
            output_file.write(line)
```

input.txt

line 1:this is first line
line 2:this is second line
line 3:this is third line
line 4:this is fourth line

OUTPUT:



```
line 1:this is first line
line 3:this is third line
```

ALGORITHM:

Define Input and Output Filenames:

Step 1: Define input_filename as "input.txt".

Step 2: Define output_filename as "output.txt".

Read the Input File:

Step 1: Use the open() function with 'r' mode to open the input file for reading.

Step 2: Use the with statement to ensure the file is closed automatically after reading.

Step 3: Use input_file.readlines() to read all the lines from the input file and store them in the lines list.

Write to the Output File:

Step 1: Use the open() function with 'w' mode to open the output file for writing.

Step 2: Use the with statement to ensure the file is closed automatically after writing.

Step 3: Loop through the lines in the lines list using enumerate() to track the line index (starting from 1).

Step 4: Check if the line index is odd (i.e., index % 2 != 0).

Step 5: If the index is odd, write the line to the output file using output_file.write(line).

File Content Example:

Input File (input.txt):

line 1:this is first line

line 2:this is second line

line 3:this is third line

line 4:this is fourth line

Output File (output.txt) (lines with odd indexes):

line 1:this is first line

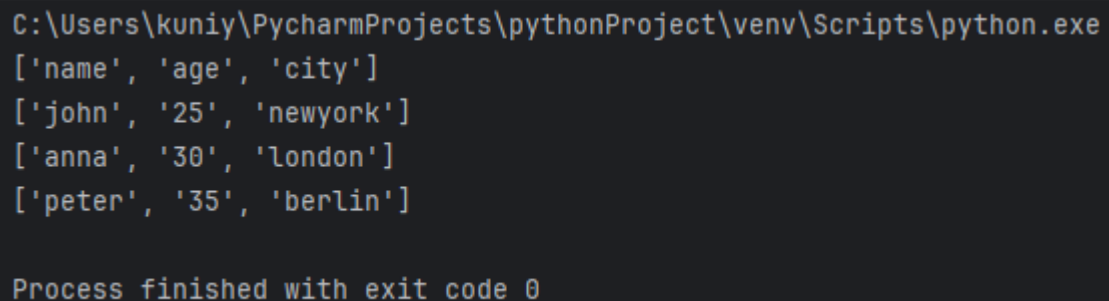
line 3:this is third line

3. Write a Python program to read each row from a given csv file and print a list of strings.

```
import csv

filename = "example.csv" # Replace with the path to your CSV file

with open(filename, 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
['name', 'age', 'city']
['john', '25', 'newyork']
['anna', '30', 'london']
['peter', '35', 'berlin']

Process finished with exit code 0
```

ALGORITHM:

Import the CSV module:

Step 1: Import the csv module using import csv.

Define the Filename:

Step 1: Set the variable filename to the path of the CSV file ("example.csv").

Open the CSV File:

Step 1: Use the `open()` function with 'r' mode to open the CSV file for reading.

Step 2: Use the `with` statement to ensure the file is automatically closed when the block is finished.

Read the CSV File:

Step 1: Use `csv.reader(file)` to create a reader object that will iterate over lines in the given CSV file.

Step 2: Loop through each row in the CSV file using a `for` loop.

Step 3: Print each row using `print(row)`

Example CSV File (`example.csv`)

4. Write a Python program to read specific columns of a given CSV file and print the content of the columns.

```
import csv

filename = "example.csv" # Replace with the path to your CSV file

columns_to_read = [0, 2] # Specify the column indices you want to read (e.g., column 0 and
column 2)

with open(filename, 'r') as file:

    reader = csv.reader(file)

    for row in reader:

        selected_columns = [row[i] for i in columns_to_read] # Extract specific columns

        print(selected_columns)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
['name', 'city']
['john', 'newyork']
['anna', 'london']
['peter', 'berlin']

Process finished with exit code 0
```

ALGORITHM:

Import the CSV module:

Step 1: Import the csv module using `import csv`.

Define the Filename and Columns to Read:

Step 1: Set the variable `filename` to the path of the CSV file ("`example.csv`").

Step 2: Define `columns_to_read` as a list of column indices you want to extract from each row (e.g., `[0, 2]` for the first and third columns).

Open the CSV File:

Step 1: Use `open()` with 'r' mode to open the CSV file for reading.

Step 2: Use `with` to ensure the file is closed automatically after processing.

Read the CSV File:

Step 1: Use `csv.reader(file)` to create a reader object.

Step 2: Loop through each row in the CSV file using a for loop.

Step 3: Use a list comprehension `[row[i] for i in columns_to_read]` to extract the values from the specified columns (based on the indices in `columns_to_read`).

Print the Selected Columns:

Step 1: Print the extracted columns from each row using `print(selected_columns)`.

Example CSV File (`example.csv`)

5. Write a Python program to write a Python dictionary to a csv file. After writing the CSV file read the CSV file and display the content.

```
import csv

data_dict = [
    {'Name': 'John', 'Age': 25, 'City': 'New York'},
    {'Name': 'Anna', 'Age': 28, 'City': 'London'},
    {'Name': 'Peter', 'Age': 34, 'City': 'Berlin'}
]

filename = 'output.csv'

headers = data_dict[0].keys()

with open(filename, 'w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=headers)
    writer.writeheader()
    writer.writerows(data_dict)

with open(filename, 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)
```

OUTPUT:

```
C:\Users\kuniy\PycharmProjects\pythonProject\venv\Scripts\python.exe
['name', 'city']
['john', 'newyork']
['anna', 'london']
['peter', 'berlin']

Process finished with exit code 0
```

ALGORITHM:

Import the CSV module:

Step 1: Import the csv module using import csv.

Prepare Data to Write:

Step 1: Define data_dict as a list of dictionaries, each containing Name, Age, and City as keys.

Example data_dict:

python

Copy code

```
data_dict = [
    {'Name': 'John', 'Age': 25, 'City': 'New York'},
    {'Name': 'Anna', 'Age': 28, 'City': 'London'},
    {'Name': 'Peter', 'Age': 34, 'City': 'Berlin'}
]
```

Set the Filename:

Step 1: Define the filename variable as 'output.csv'.

Write Data to CSV:

Step 1: Extract the headers (column names) from the first dictionary of data_dict using data_dict[0].keys().

Step 2: Open the CSV file in write mode using open(filename, 'w', newline="").

Step 3: Create a csv.DictWriter object, passing file and fieldnames=headers to it.

Step 4: Write the headers using writer.writeheader().

Step 5: Write the rows of data from data_dict using writer.writerows(data_dict).

Read Data from CSV:

Step 1: Open the CSV file in read mode using open(filename, 'r').

Step 2: Create a csv.reader object and loop through each row in the CSV file using a for loop.

Step 3: Print each row using print(row).