Practical no. 1

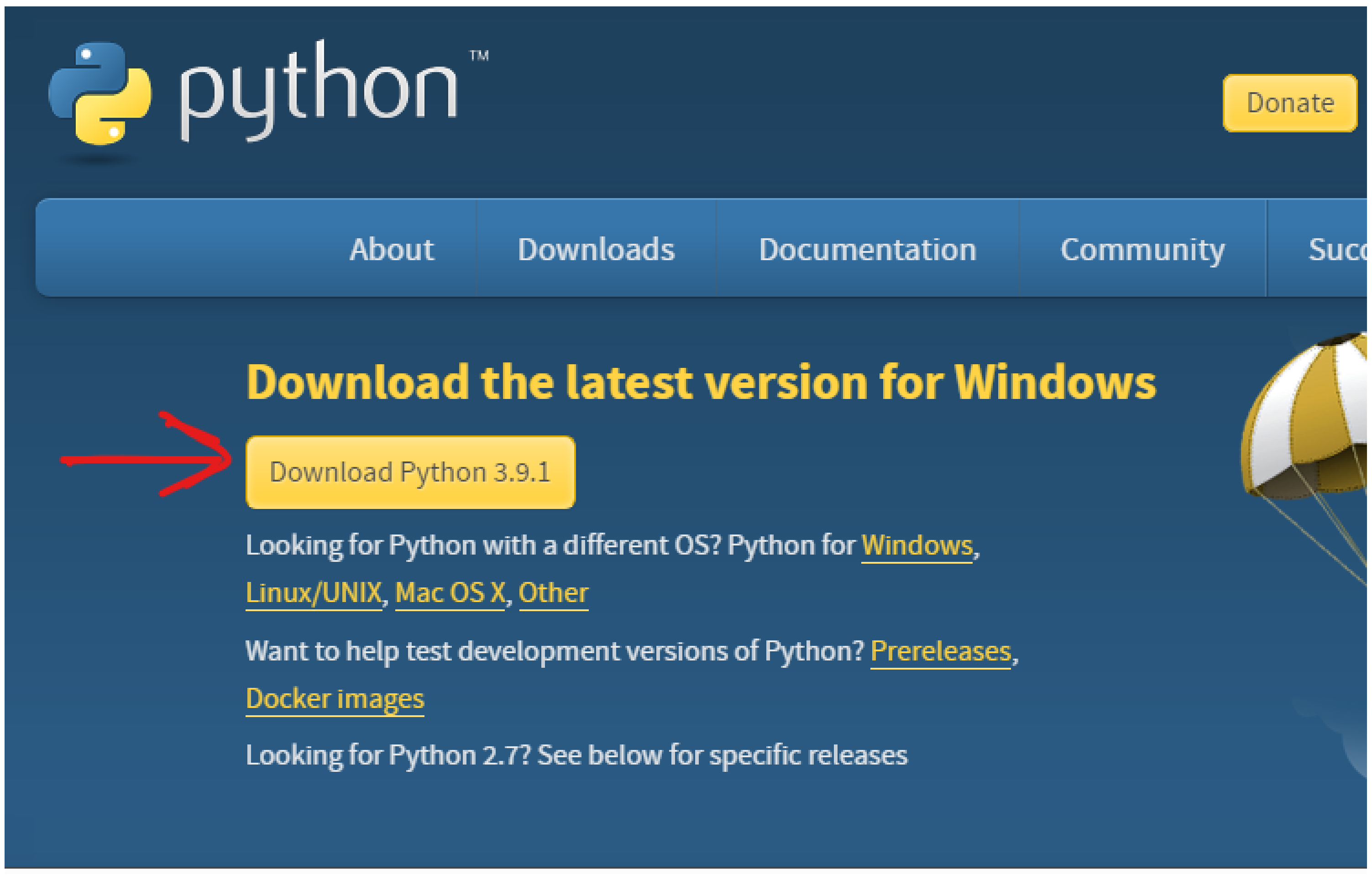
FS19CO042

Aim: Install & configure Python IDE

Theory:

Step 1: Download Python 3.9

To start, go to python.org/downloads and then click on the button to download the latest version of Python:



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

Want to help test development versions of Python? Prereleases, Docker images

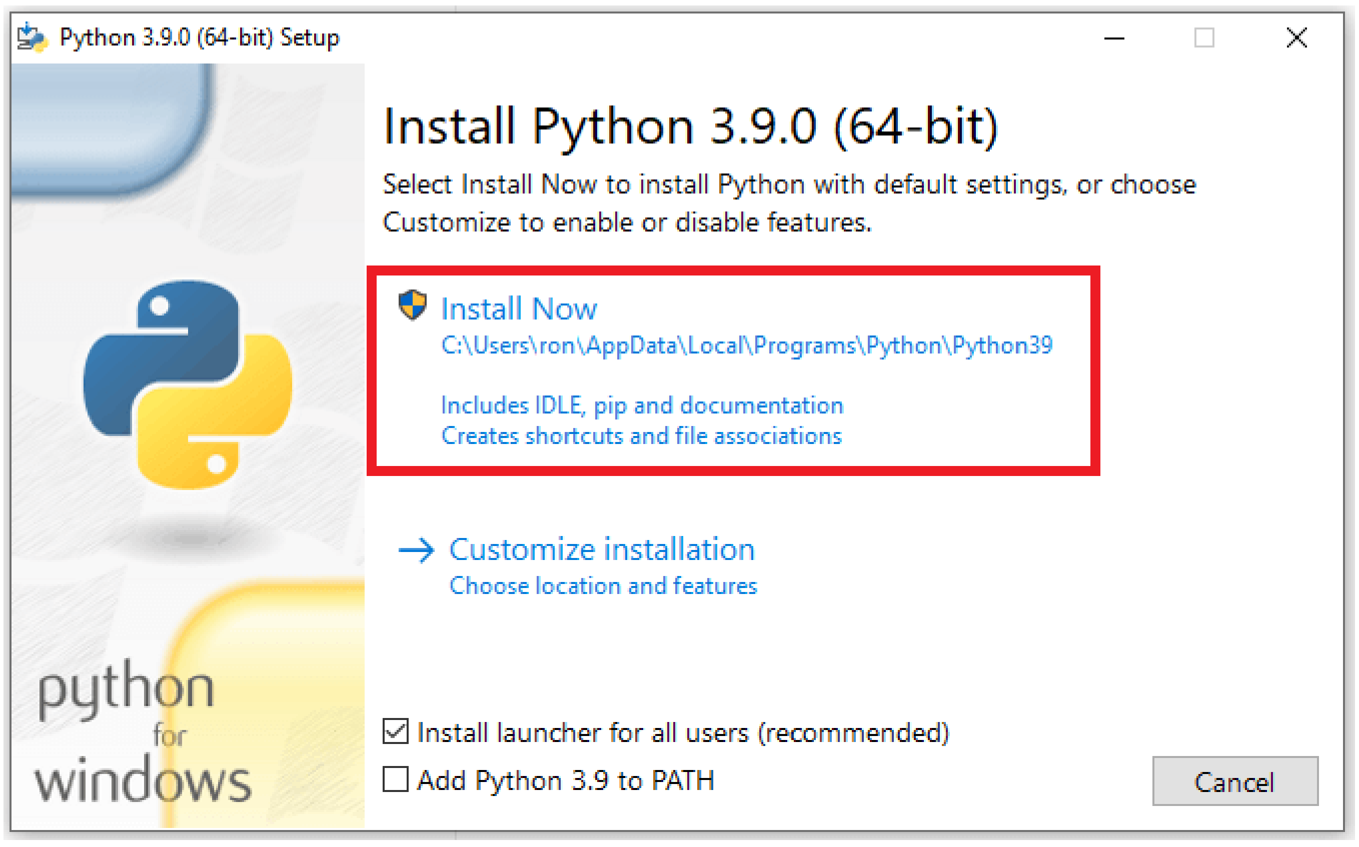
Looking for Python 2.7? See below for specific releases

Step 2: Run the .exe file

Next, run the .exe file that you just downloaded:

Step 3: Install Python 3.9

You can now start the installation of Python by clicking on Install Now:


Python 3.9.0 (64-bit) Setup

Install Python 3.9.0 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\ron\AppData\Local\Programs\Python\Python39

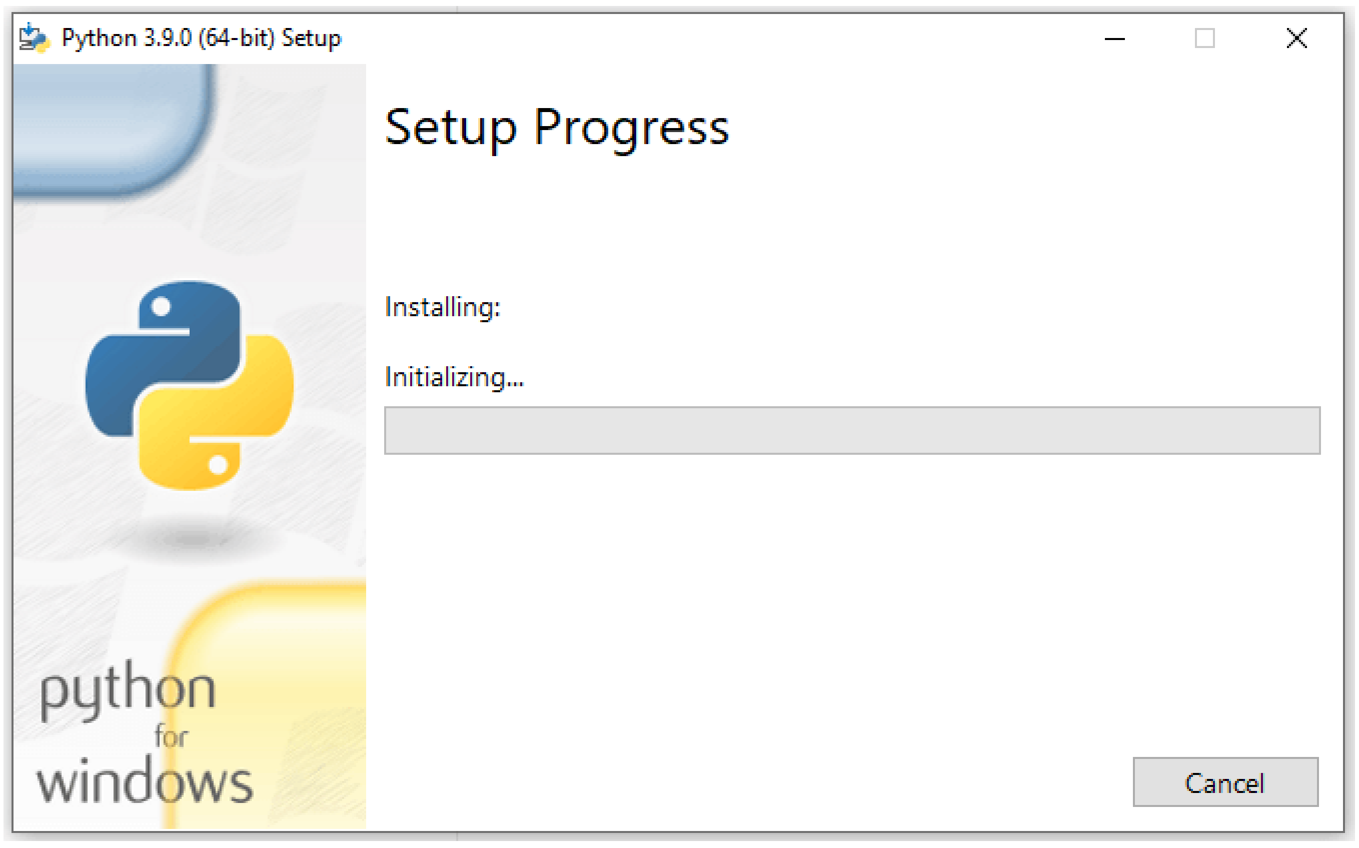
Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☐ Add Python 3.9 to PATH

Note that depending on your needs, you may also check the box to add Python to the Path.

Your installation should now begin:


Python 3.9.0 (64-bit) Setup

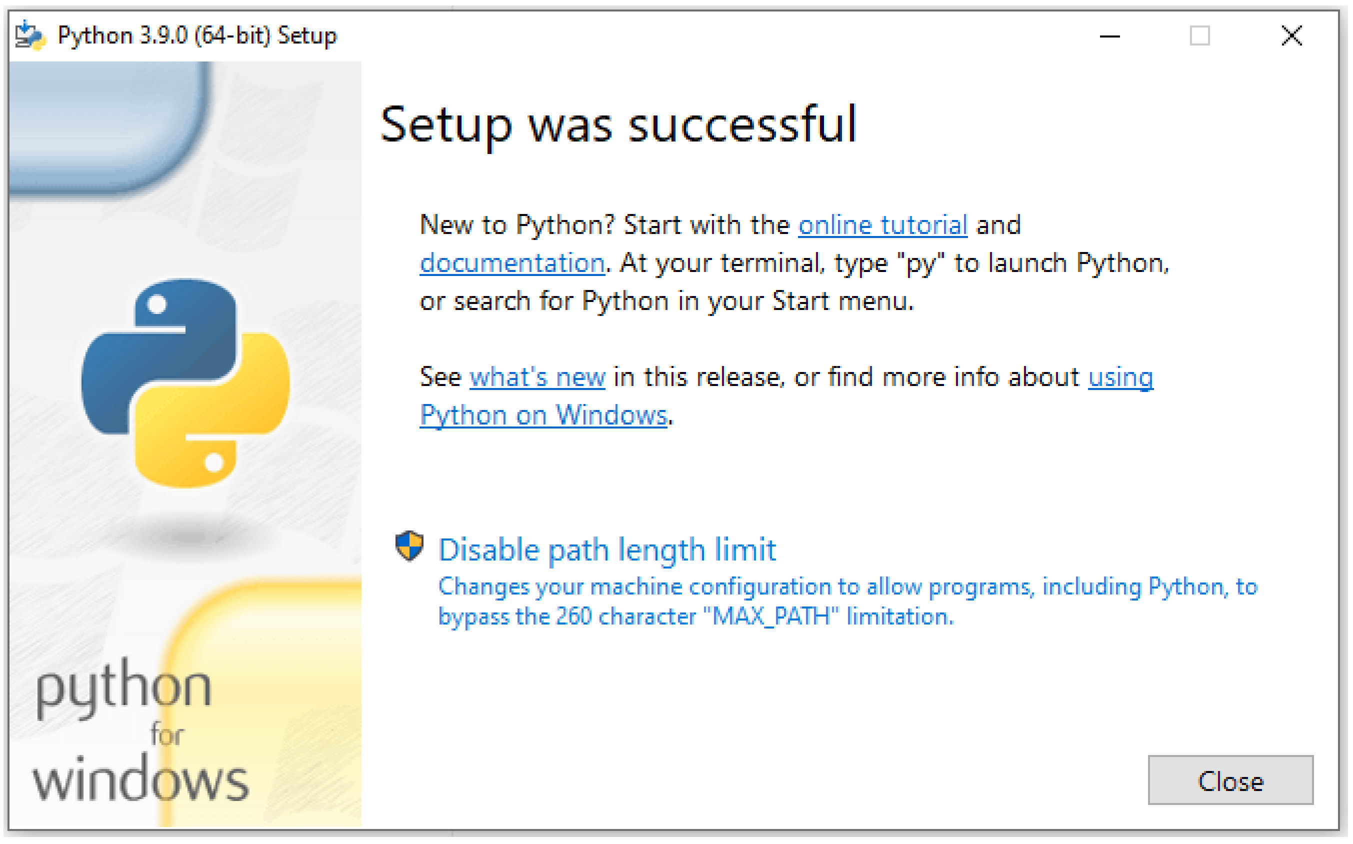
Setup Progress

Installing:

Initializing...

Setup Progress

After a short period of time, your setup would be completed:


Python 3.9.0 (64-bit) Setup

Setup was successful

New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.

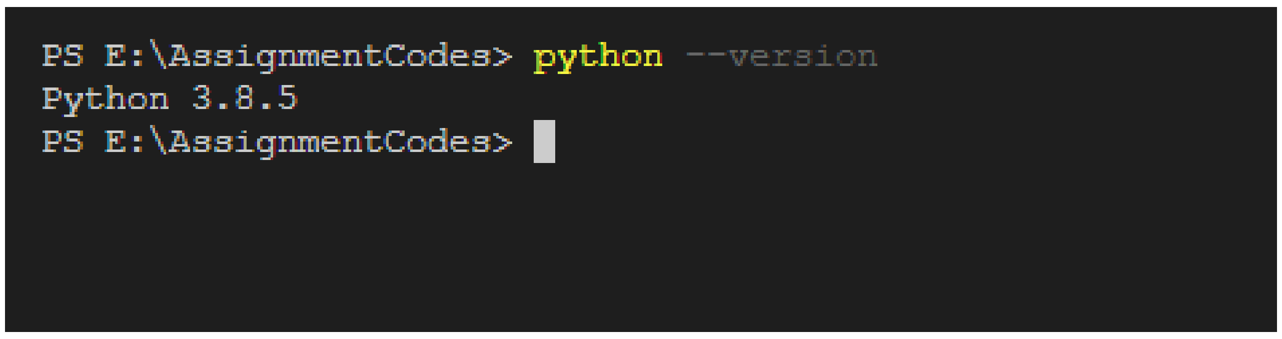
See what's new in this release, or find more info about using Python on Windows.

Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.

Successful Setup

Congrats, you just installed Python on Windows!

Now let's check that python is installed properly through the terminal.


```
PS E:\AssignmentCodes> python --version
Python 3.8.5
PS E:\AssignmentCodes>
```

<u>Conclusion:</u> Thus we installed python on our system successfully.

Practical no. 2

FS19CO042

Aim: Illustrate use of various operators.

Problem statements: Write a python program using operators.

Theory:

- **Operators**
  Operators are special symbols in Python to carry out certain computation. Value on which operator operates is called operand. Operators are used to perform operations on variables and values

Python divides the operators in the following groups:

1. **Arithmetic operators**
2. **Assignment operators**
3. **Comparison operators**
4. **Logical operators**
5. **Identity operators**
6. **Membership operators**
7. **Bitwise operators**

Arithmetic operators :

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

Operator	Meaning	Example
+	Add two operands or unary plus	x + y+ 2
-	Subtract right operand from the left or unary minus	x - y- 2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

<u>Assignment operators</u> :

Assignment operators are used in Python to assign values to variables.

a = 5 is a simple assignment operator that assigns the value 5 on the right to the variable a on the left.

There are various compound operators in Python like a += 5 that adds to the variable and later assigns the same. It is equivalent to a = a + 5.

Operator	**Example**	**Equivalent to**
=	**x = 5**	**x = 5**
+=	**x += 5**	**x = x + 5**
-=	**x -= 5**	**x = x - 5**
***=**	**x *= 5**	**x = x * 5**
/=	**x /= 5**	**x = x / 5**
%=	**x %= 5**	**x = x % 5**
//=	**x //= 5**	**x = x // 5**
****=**	**x **= 5**	**x = x ** 5**
&=	**x &= 5**	**x = x & 5**
|=	**x |= 5**	**x = x | 5**
^=	**x ^= 5**	**x = x ^ 5**
>>=	**x >>= 5**	**x = x >> 5**
<<=	**x <<= 5**	**x = x << 5**

Comparison operators :

Comparison operators are used to compare values. It returns either True or False according to the condition.

Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Logical operators :

Logical operators are the and, or, not operators.

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Identity operators :

is and is not are the identity operators in Python. They are used to check if two values (or variables) are located on the same part of the memory. Two variables that are equal does not imply that they are identical.

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

Membership operators :

in and not in are the membership operators in Python. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

In a dictionary we can only test for presence of key, not the value.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Bitwise operators :

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name.

For example, 2 is 10 in binary and 7 is 111.

In the table below: Let x = 10 (0000 1010 in binary) and y = 4 (0000 0100 in binary)

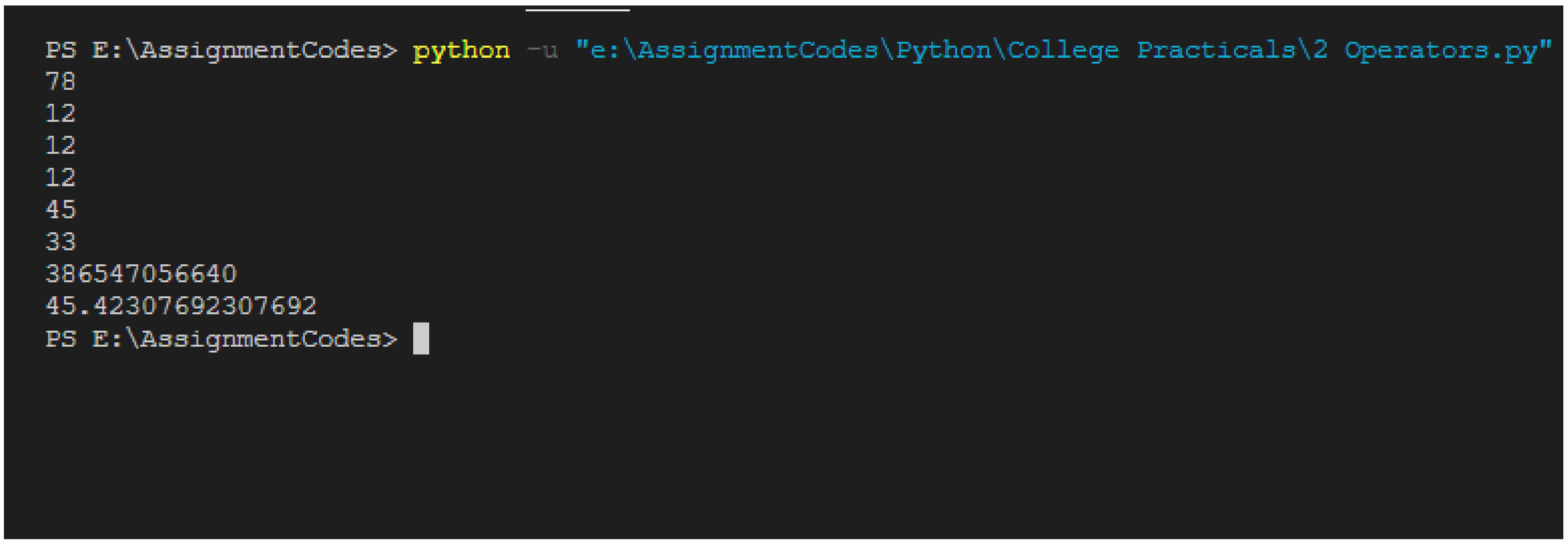
Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)

~	**Bitwise NOT**	**~x = -11 (1111 0101)**
^	**Bitwise XOR**	**x ^ y = 14 (0000 1110)**
>>	**Bitwise right shift**	**x >> 2 = 2 (0000 0010)**
<<	**Bitwise left shift**	**x << 2 = 40 (0010 1000)**

Code and outputs:

```
a = 45
b = 33
c = a+b
print(c)
print(a-b)
print(a%b)
print(a^b)
print(a|b)
print(a&b)
print(a<<b)
d = a+b/c
print(d)
```

Output:



```
PS E:\AssignmentCodes> python -u "e:\AssignmentCodes\Python\College Practicals\2 Operators.py"
78
12
12
12
45
33
386547056640
45.42307692307692
PS E:\AssignmentCodes>
```

<u>Conclusion:</u> Thus we understood and executed python program using python operators.

Practical no. 3

FS19CO042

Aim: Write python program to demonstrate math built-in functions (any two).

Problem statements:

- Use pi, calculate area of circle for a given radius.
- Use any two in built math functions
- Find roots of quadratic equations.

Theory:

Python math Module
Python has a built-in module that you can use for mathematical tasks.
The math module has a set of methods and constants.

Math Methods:

Method	**Description**
math.acos()	Returns the arc cosine of a number
math.acosh()	Returns the inverse hyperbolic cosine of a number
math.asin()	Returns the arc sine of a number
math.asinh()	Returns the inverse hyperbolic sine of a number
math.atan()	Returns the arc tangent of a number in radians
math.atan2()	Returns the arc tangent of y/x in radians
math.atanh()	Returns the inverse hyperbolic tangent of a number
math.ceil()	Rounds a number up to the nearest integer
math.comb()	Returns the number of ways to choose k items from n items without repetition and order
math.copysign()	Returns a float consisting of the value of the first parameter and the sign of the second parameter
math.cos()	Returns the cosine of a number
math.cosh()	Returns the hyperbolic cosine of a number
math.degrees()	Converts an angle from radians to degrees
math.dist()	Returns the Euclidean distance between two points (p and q), where p and q are the coordinates of that point
math.erf()	Returns the error function of a number
math.erfc()	Returns the complementary error function of a number
math.exp()	Returns E raised to the power of x
math.expm1()	Returns E^x - 1
math.fabs()	Returns the absolute value of a number
math.factorial()	Returns the factorial of a number
math.floor()	Rounds a number down to the nearest integer
math.fmod()	Returns the remainder of x/y
math.frexp()	Returns the mantissa and the exponent, of a specified number
math.fsum()	Returns the sum of all items in any iterable (tuples, arrays, lists, etc.)
math.gamma()	Returns the gamma function at x
math.gcd()	Returns the greatest common divisor of two integers
math.hypot()	Returns the Euclidean norm
math.isclose()	Checks whether two values are close to each other, or not
math.isfinite()	Checks whether a number is finite or not
math.isinf()	Checks whether a number is infinite or not
math.isnan()	Checks whether a value is NaN (not a number) or not
math.isqrt()	Rounds a square root number downwards to the nearest integer
math.ldexp()	Returns the inverse of math.frexp() which is x * (2**i) of the given numbers x and i
math.lgamma()	Returns the log gamma value of x
math.log()	Returns the natural logarithm of a number, or the logarithm of number to base
math.log10()	Returns the base-10 logarithm of x
math.log1p()	Returns the natural logarithm of 1+x
math.log2()	Returns the base-2 logarithm of x

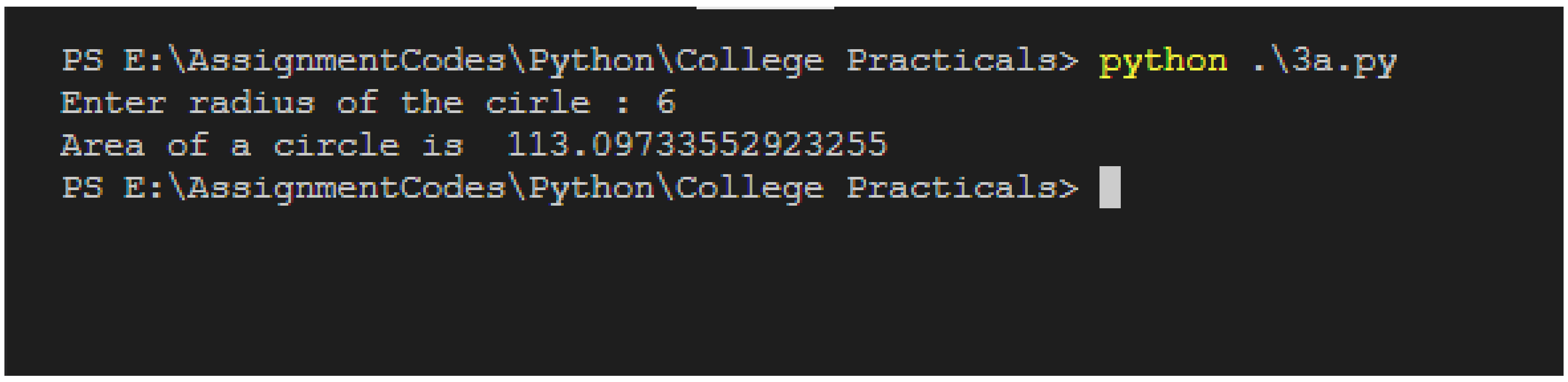
math.perm()	Returns the number of ways to choose k items from n items with order and without repetition
math.pow()	Returns the value of x to the power of y
math.prod()	Returns the product of all the elements in an iterable
math.radians()	Converts a degree value into radians
math.remainder()	Returns the closest value that can make numerator completely divisible by the denominator
math.sin()	Returns the sine of a number
math.sinh()	Returns the hyperbolic sine of a number
math.sqrt()	Returns the square root of a number
math.tan()	Returns the tangent of a number
math.tanh()	Returns the hyperbolic tangent of a number
math.trunc()	Returns the truncated integer parts of a number

Code and outputs:

1. Use pi, calculate area of circle for given radius

```
from math import pi
radius = int(input("Enter radius of the cirle : "))
area = pi * radius * radius
print("Area of a circle is ", area)
```

Output:

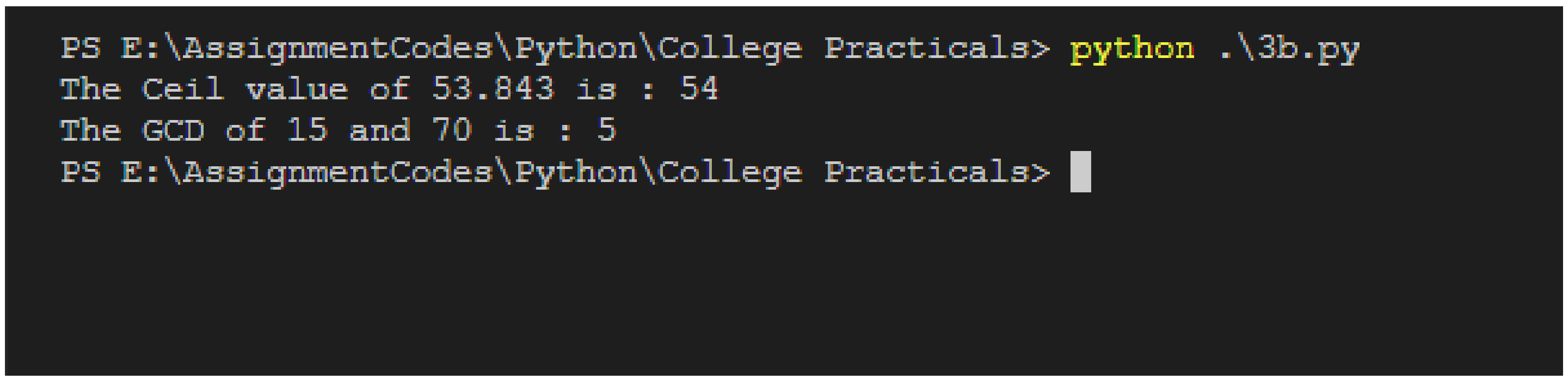


```
PS E:\AssignmentCodes\Python\College Practicals> python .\3a.py
Enter radius of the cirle : 6
Area of a circle is  113.09733552923255
PS E:\AssignmentCodes\Python\College Practicals>
```

2. Use any two built in math functions

```
import math
x = 53.843
p = 15
q = 70

# returning the ceil value i.e. rounds no. to upper value
print (f'The Ceil value of 53.843 is : {math.ceil(x)}')

# returning the GCD of two numbers
print (f'The GCD of 15 and 70 is : {math.gcd(p, q)}')
```

Output:



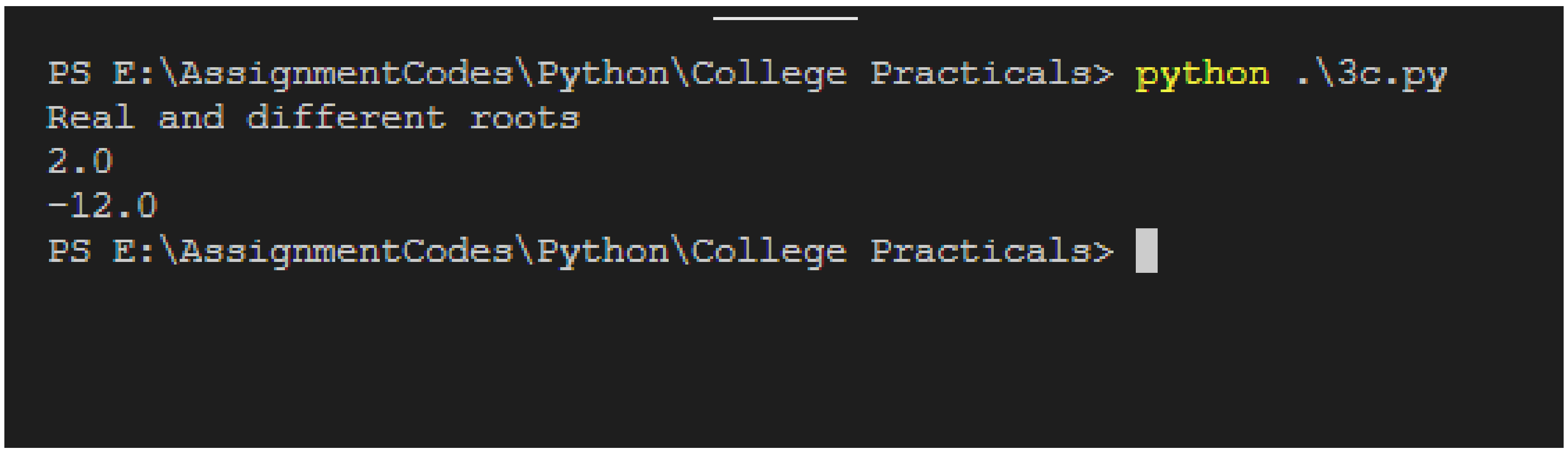
```
PS E:\AssignmentCodes\Python\College Practicals> python .\3b.py
The Ceil value of 53.843 is : 54
The GCD of 15 and 70 is : 5
PS E:\AssignmentCodes\Python\College Practicals>
```

3. Find roots of quadratic equation

```
# Python program to find roots of quadratic equation
import math
# function for finding roots
def equationroots(a, b, c):
   dis = b * b - 4 * a * c
   sqrt_val = math.sqrt(abs(dis))

   # checking condition for discriminant
   if dis > 0:
      print("Real and different roots ")
      print((-b + sqrt_val) / (2 * a))
      print((-b - sqrt_val) / (2 * a))

   elif dis == 0:
      print("Real and same roots")
      print(-b / (2 * a))

   # when discriminant is less than 0
   else:
      print("Complex Roots")
      print(- b / (2 * a), " + i", sqrt_val)
      print(- b / (2 * a), " - i", sqrt_val)

a = 1
b = 10
c = -24

# If a is 0, then incorrect equation
```

```
if a == 0:
    print("Input correct quadratic equation")
else:
    equationroots(a, b, c)
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python .\3c.py
Real and different roots
2.0
-12.0
PS E:\AssignmentCodes\Python\College Practicals>
```

<u>**Conclusion:**</u> Thus we understood and used functions from math module

Practical no. 7

FS19CO042

Aim: Write a Python program to demonstrate the use of if and if else.

Problem statements:

1. Check if given number is even or odd.
2. Display grade of student depending on percentage.
   percentage>=90: Excellent
   percentage>=80 and <90: First class
   percentage>=60 and <80: Second class
   percentage>=40 and <60: Pass class
   percentage < 40: Fail

Theory:

- **If statement**
  An if statement is used to execute a statement/set of statements only when the specified condition is satisfied i.e. expression resolves TRUE, otherwise program continues unaffected.

  Syntax:

  ```
  if expression:
          statementset
  ```

  For example,

  ```
  if n==5:
          print("No. is 5")                  # This statement gets executed only if n=5
  ```

- **If….else statement**
  An If…else statement is combination of If with else. Here, statement1 occurs only if expression resolves TRUE otherwise statement2 occurs.
  Here, either of the statement occurs.

  Syntax:

  ```
  if expression:
          statementset1
  else:
          statementset2
  ```

  For example,

  ```
  if n==5:
  ```

```
    print("No. is 5")               # This statement gets executed only if n=5
else:
    print("No. is not 5")           # If no. is not equal to 5, this statement occurs
```

- **Elif statement**

The **elif** statement allows checking multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.

Similar to the **else**, the **elif** statement is optional. One can put as many elif as possible after if.

Syntax:

```
if expression:
    statementset1
elif expression2:
    statementset2
elif expression3:
    statementset3
else:
    statementset4
```

For example,

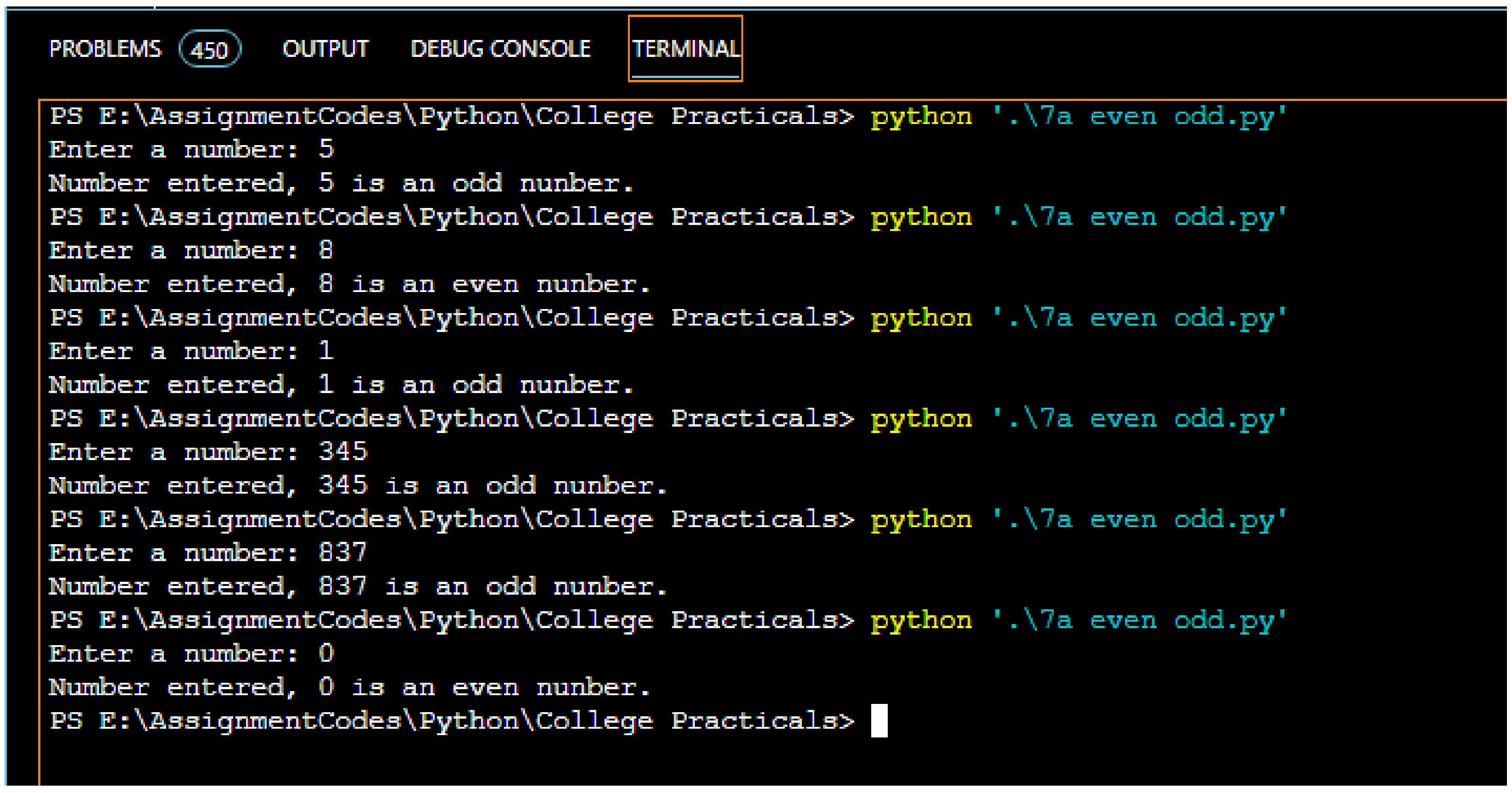
```
If n==2:
    print("n is 2!")
elif n==3:
    print("n is 3!")
elif n==4:
    print("n is 4!")
else:
    print("Don't know what it is ");
```

Code and outputs:

1. Check if given number is even or odd.

```
# Check if given number is even or odd.
number = int(input("Enter a number: "))

if number%2 == 0:
   print("Number entered,", number, "is an even nunber.")
else:
   print("Number entered,", number, "is an odd nunber.")
```

Output:

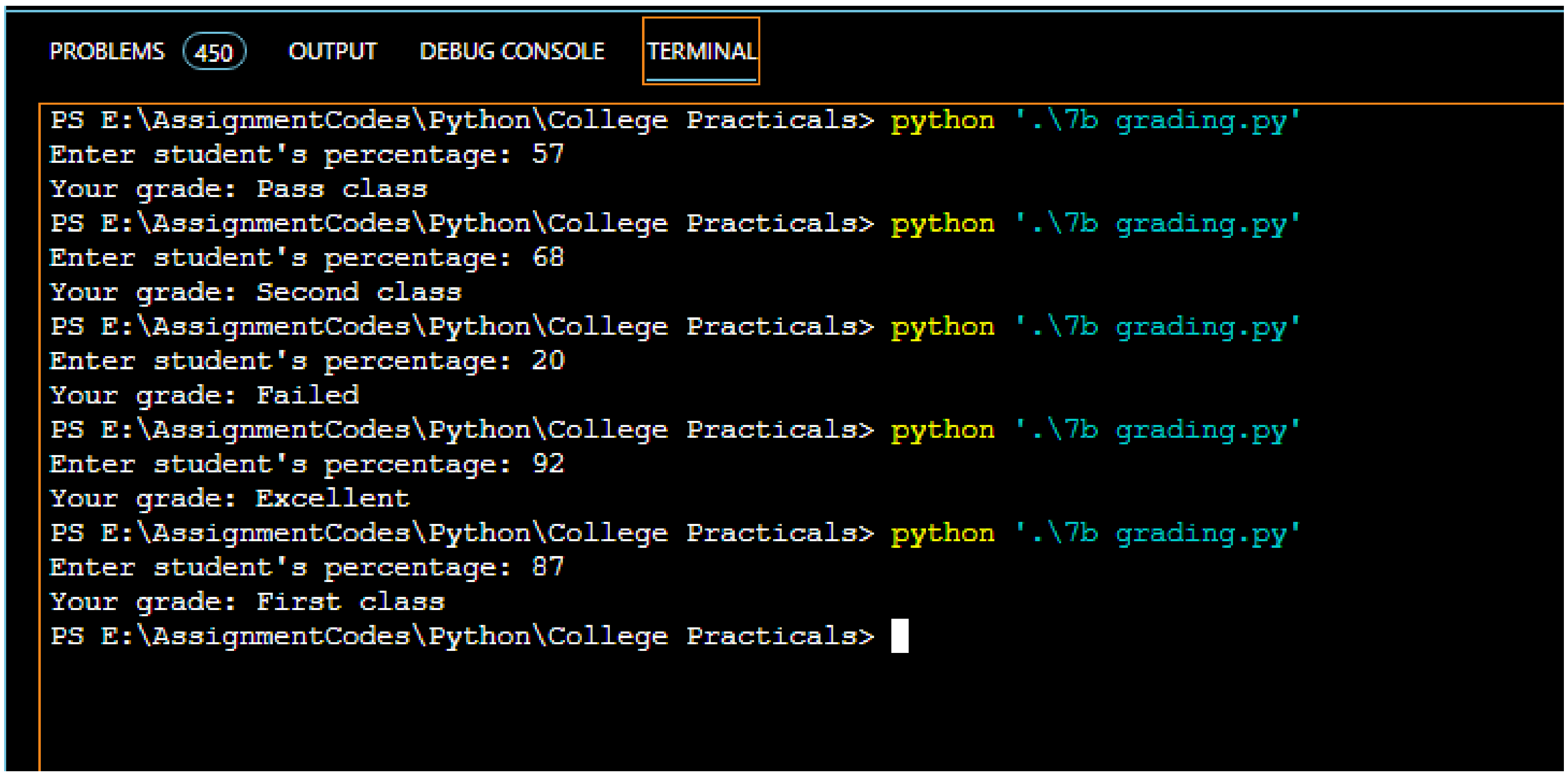


```
PS E:\AssignmentCodes\Python\College Practicals> python '.\7a even odd.py'
Enter a number: 5
Number entered, 5 is an odd nunber.
PS E:\AssignmentCodes\Python\College Practicals> python '.\7a even odd.py'
Enter a number: 8
Number entered, 8 is an even nunber.
PS E:\AssignmentCodes\Python\College Practicals> python '.\7a even odd.py'
Enter a number: 1
Number entered, 1 is an odd nunber.
PS E:\AssignmentCodes\Python\College Practicals> python '.\7a even odd.py'
Enter a number: 345
Number entered, 345 is an odd nunber.
PS E:\AssignmentCodes\Python\College Practicals> python '.\7a even odd.py'
Enter a number: 837
Number entered, 837 is an odd nunber.
PS E:\AssignmentCodes\Python\College Practicals> python '.\7a even odd.py'
Enter a number: 0
Number entered, 0 is an even nunber.
PS E:\AssignmentCodes\Python\College Practicals>
```

2. **Display grade of student depending on percentage.**
   percentage>=90: Excellent
   percentage>=80 and <90: First class
   percentage>=60 and <80: Second class
   percentage>=40 and <60: Pass class
   percentage < 40: Fail

```python
# Display grade of student depending on percentage
percentage = int(input("Enter student's percentage: "))
print("Your grade: ",end="")

if percentage >= 90:
   print("Excellent")
elif percentage >= 80:
   print("First class")
elif percentage >= 60:
   print("Second class")
elif percentage >= 40:
   print("Pass class")
else:
   print ("Failed")
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python '.\7b grading.py'
Enter student's percentage: 57
Your grade: Pass class
PS E:\AssignmentCodes\Python\College Practicals> python '.\7b grading.py'
Enter student's percentage: 68
Your grade: Second class
PS E:\AssignmentCodes\Python\College Practicals> python '.\7b grading.py'
Enter student's percentage: 20
Your grade: Failed
PS E:\AssignmentCodes\Python\College Practicals> python '.\7b grading.py'
Enter student's percentage: 92
Your grade: Excellent
PS E:\AssignmentCodes\Python\College Practicals> python '.\7b grading.py'
Enter student's percentage: 87
Your grade: First class
PS E:\AssignmentCodes\Python\College Practicals>
```

Conclusion: Thus, we understood and implemented If, elif and if-else while implementing given problem statements.

Practical no. 8

FS19CO042

Aim: Develop a user defined Python function to demonstrate the use of parameterized function & value return functions.

Problem statements:

1. Check if a number is prime or not.
2. Write a menu driven program, using user defined functions to find the area of rectangle, square, circle and triangle by accepting suitable input parameters from user.

Theory:

- **Functions in python**

  A function is a reusable block of code which only runs when it is called. It can be called as many times as possible without writing same code again and again.

  Example,

  ```
  # Defining a function
  def sayHello():
        print("Hello Pythoneeers !!")

  # Calling a function
  sayHello()
  ```

- **Parameterized functions**

  Functions can be given different arguments which take value when they're called and perform operations as per the value specified each time.

  Example

  ```
  def sayHello(name):
        Print("Hello pythoneer "+name)

  sayHello("Jonas")
  sayHello("Mark")
  ```

- **Return functions**

  A return statement ends execution of the function call and returns the result obtained by performing certain operations.

  Example,

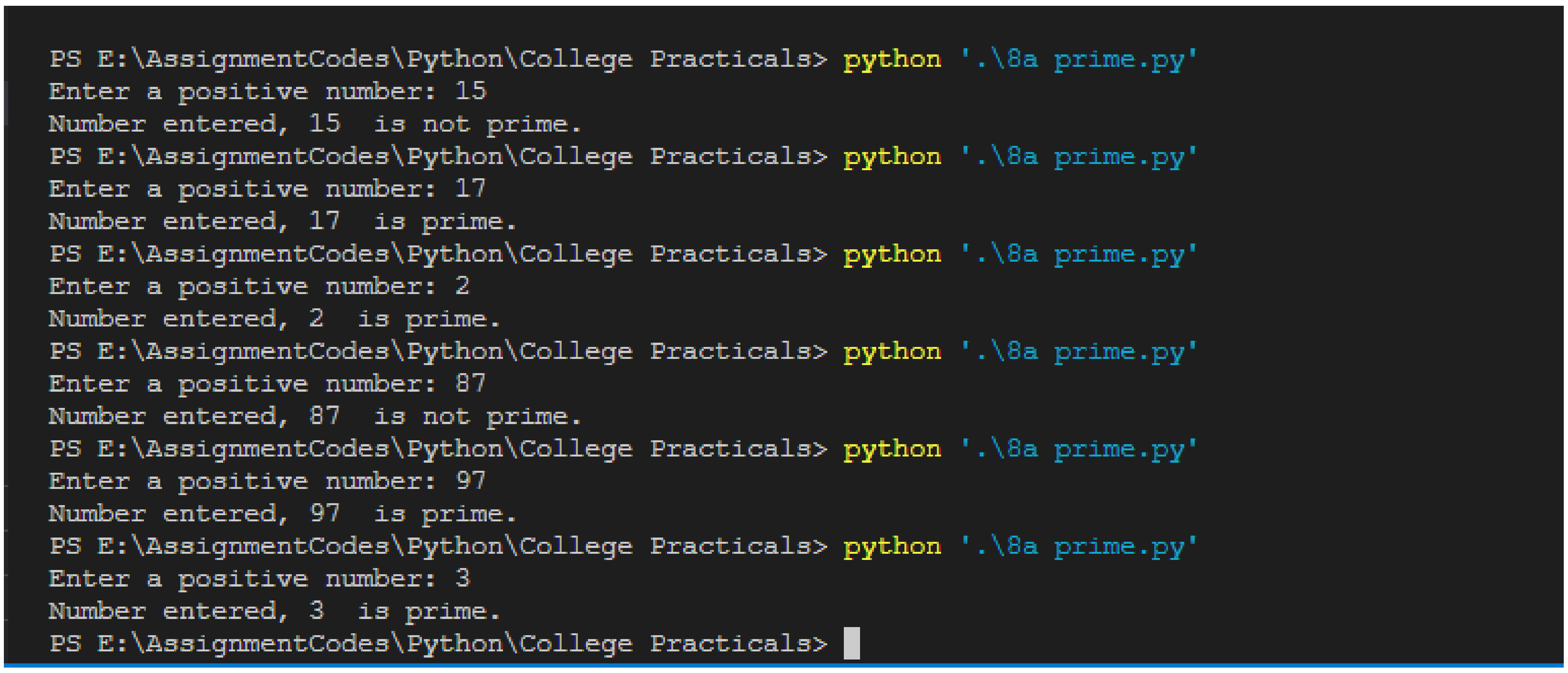
  ```
  def add(a,b):
        return a+b

  print(add(5,7))          # 12
  print(add(32,3))         # 35
  ```

Code and outputs:

1. Check if given number is even or odd.

```
def isPrime(number):
    if number <= 2:
        return True
    # If number is even it is never prime except for 2
    if number%2 == 0:
        return False

    i = 3
    # Iterate over all numbers until square root of number.
    while i*i<=number:
        if number%i == 0:
            return False
        i += 2
    return True

number = int(input("Enter a positive number: "));
if number == 1:
    print("Number is neither prime nor composite !")
else:
    if isPrime(number):
        print("Number entered,", number, " is prime.")
    else:
        print("Number entered,", number, " is not prime.")
```

Output:

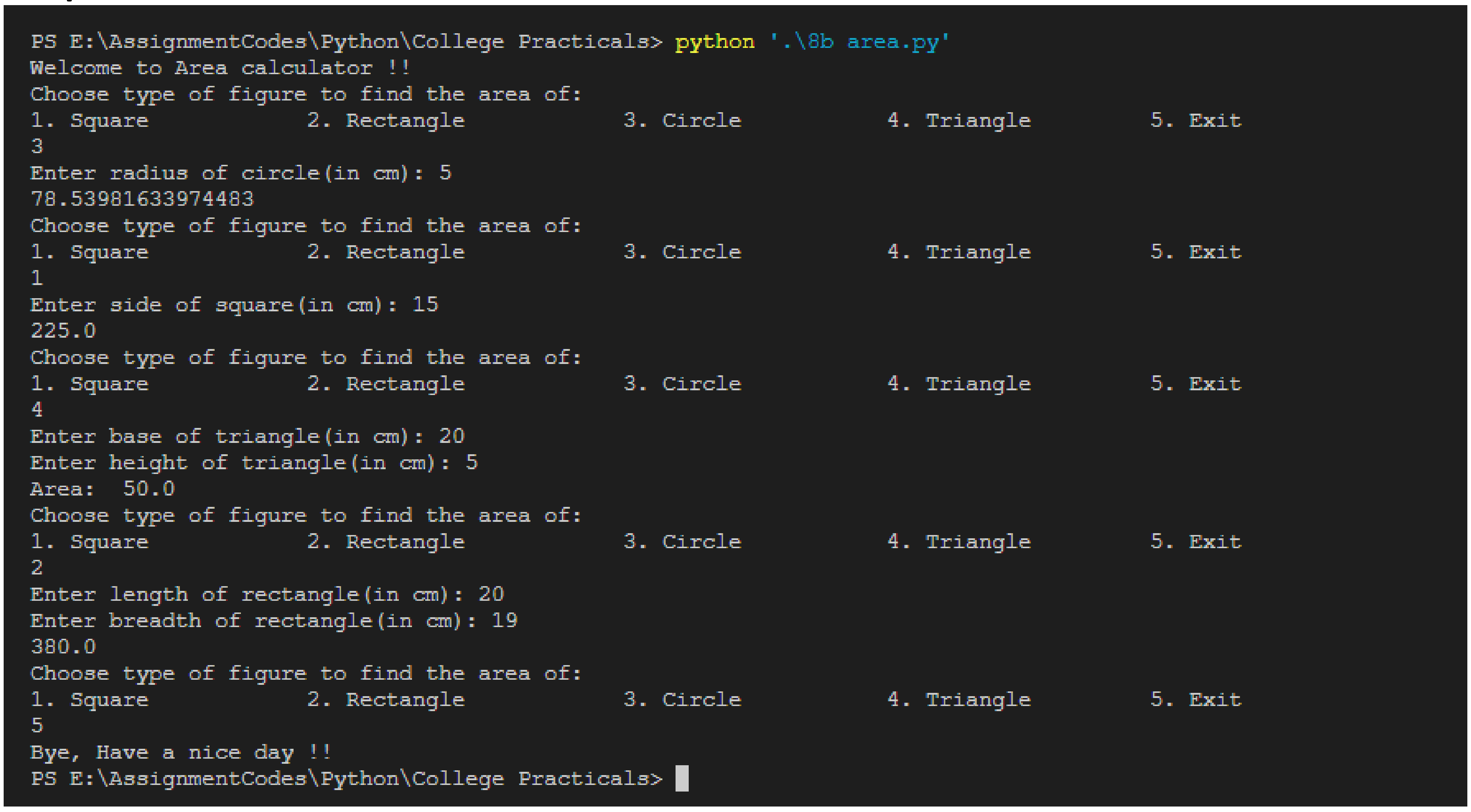


```
PS E:\AssignmentCodes\Python\College Practicals> python '.\8a prime.py'
Enter a positive number: 15
Number entered, 15  is not prime.
PS E:\AssignmentCodes\Python\College Practicals> python '.\8a prime.py'
Enter a positive number: 17
Number entered, 17  is prime.
PS E:\AssignmentCodes\Python\College Practicals> python '.\8a prime.py'
Enter a positive number: 2
Number entered, 2  is prime.
PS E:\AssignmentCodes\Python\College Practicals> python '.\8a prime.py'
Enter a positive number: 87
Number entered, 87  is not prime.
PS E:\AssignmentCodes\Python\College Practicals> python '.\8a prime.py'
Enter a positive number: 97
Number entered, 97  is prime.
PS E:\AssignmentCodes\Python\College Practicals> python '.\8a prime.py'
Enter a positive number: 3
Number entered, 3  is prime.
PS E:\AssignmentCodes\Python\College Practicals>
```

2. Write a menu driven program, using user defined functions to find the area of rectangle, square, circle and triangle by accepting suitable input parameters from user

```
from math import pi

def areaOfCircle(radius):
    return pi*radius*radius

def areaOfSquare(side):
    return side*side

def areaOfRectangle(length, breadth):
    return length*breadth

def areaOfTriangle(base, height):
    return 0.5 * base * height

print("Welcome to Area calculator !!")

while(True):
    print("Choose type of figure to find the area of: ")
    print("1. Square        2. Rectangle        3. Circle        4. Triangle        5. Exit")

    choice =  float(input())

    if choice == 1:
        side =  float(input("Enter side of square(in cm): "))
        print(areaOfSquare(side))
    elif choice == 2:
        len =  float(input("Enter length of rectangle(in cm): "))
        breadth =  float(input("Enter breadth of rectangle(in cm): "))
        print(areaOfRectangle(len, breadth))
    elif choice == 3:
        radius =  float(input("Enter radius of circle(in cm): "))
        print(areaOfCircle(radius))
    elif choice == 4:
        base =  float(input("Enter base of triangle(in cm): "))
        height =  float(input("Enter height of triangle(in cm): "))
        print("Area: ", areaOfTriangle(base, height))
    elif choice == 5:
        print("Bye, Have a nice day !!")
        break
    else:
        print("Invalid choice, try again")
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python '.\8b area.py'
Welcome to Area calculator !!
Choose type of figure to find the area of:
1. Square           2. Rectangle          3. Circle           4. Triangle         5. Exit
3
Enter radius of circle(in cm): 5
78.53981633974483
Choose type of figure to find the area of:
1. Square           2. Rectangle          3. Circle           4. Triangle         5. Exit
1
Enter side of square(in cm): 15
225.0
Choose type of figure to find the area of:
1. Square           2. Rectangle          3. Circle           4. Triangle         5. Exit
4
Enter base of triangle(in cm): 20
Enter height of triangle(in cm): 5
Area:  50.0
Choose type of figure to find the area of:
1. Square           2. Rectangle          3. Circle           4. Triangle         5. Exit
2
Enter length of rectangle(in cm): 20
Enter breadth of rectangle(in cm): 19
380.0
Choose type of figure to find the area of:
1. Square           2. Rectangle          3. Circle           4. Triangle         5. Exit
5
Bye, Have a nice day !!
PS E:\AssignmentCodes\Python\College Practicals>
```

Conclusion:

Thus we understood and implemented parameterized and return functions along with menu driven program.

Practical no. 9

FS19CO042

Aim: Write a Python program to perform following operations on tuples:

a. Create b. Access c. Update d. Delete

Problem statements:

Create a tuple of 5 strings and perform various operations

Theory:

- **Tuples**

  Tuples is a immutable collection of elements into single variable. It is a built in data type. Tuples are unchangeable (immutable) and ordered (remain in same order as created)

  Example:
  ```
  tupleName = (“abc”, “cde”, “xyz”)
  print(tupleName)
  ```

- **Creating tuples from tuple() constructor**

  Tuple constructor is used to create tuple from already defined variables like lists, sets, etc.

  Example:
  ```
  Ls = [“abc”, “cde”, “efg”]
  Tup = tuple(Ls)
  print(Tup)
  ```

Tuples can never be changes, to update tuple, we may create list from tuple, update that list and then convert that list to tuple again.

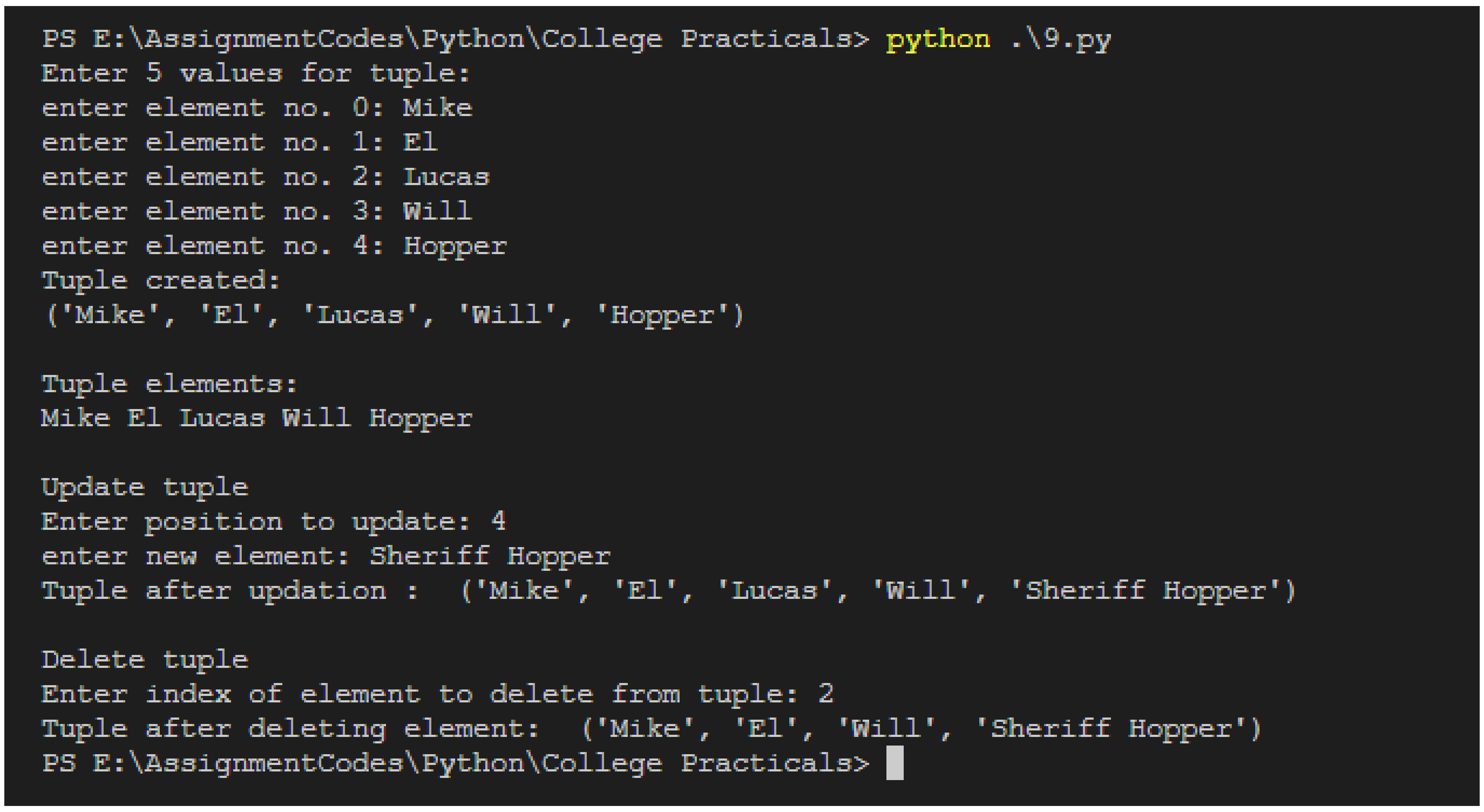
Code and outputs:

Code:

```
# Create tuple
print("Enter 5 values for tuple: ")
ls = []
for i in range(5):
   ls.append(input(f'enter element no. {i}: '))
tup1 = tuple(ls)
print("Tuple created:")
print(tup1)
print()
```

```
# Accessing element by element
print("Tuple elements: ")
for x in tup1:
    print(x, end=" ")
print()
print()

# Update tuple
print('Update tuple ')
idx = int(input('Enter position to update: '))
ele = input('enter new element: ')
tempList = list(tup1)
tempList[idx] = ele
tup1 = tuple(tempList)
print('Tuple after updation : ', tup1)
print()

# Delete element from tuple
print('Delete tuple')
idx = int(input('Enter index of element to delete from tuple: '))
tempList = list(tup1)
tempList.pop(idx)
tup1=tuple(tempList)
print('Tuple after deleting element: ', tup1)
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python .\9.py
Enter 5 values for tuple:
enter element no. 0: Mike
enter element no. 1: El
enter element no. 2: Lucas
enter element no. 3: Will
enter element no. 4: Hopper
Tuple created:
('Mike', 'El', 'Lucas', 'Will', 'Hopper')

Tuple elements:
Mike El Lucas Will Hopper

Update tuple
Enter position to update: 4
enter new element: Sheriff Hopper
Tuple after updation :  ('Mike', 'El', 'Lucas', 'Will', 'Sheriff Hopper')

Delete tuple
Enter index of element to delete from tuple: 2
Tuple after deleting element:  ('Mike', 'El', 'Will', 'Sheriff Hopper')
PS E:\AssignmentCodes\Python\College Practicals>
```

Conclusion: Thus, we performed various operations on tuples

Practical no. 10

FS19CO042

Aim: Write a Python program to read keyboard input & print it to the screen.

Problem statements:

- Read your name, contact number, percentage from keyboard and store them in appropriate variables (use type conversion if needed). Print all stored data.

Theory:

- **print()**
  print function is used to print content to the console.

- **input()**
  input function is used to take input from user and returns a string.
  It accepts an optional parameter of type string which is prompted to user.
  For example, inputStr = input("Enter some data")

- **Type conversion functions:**

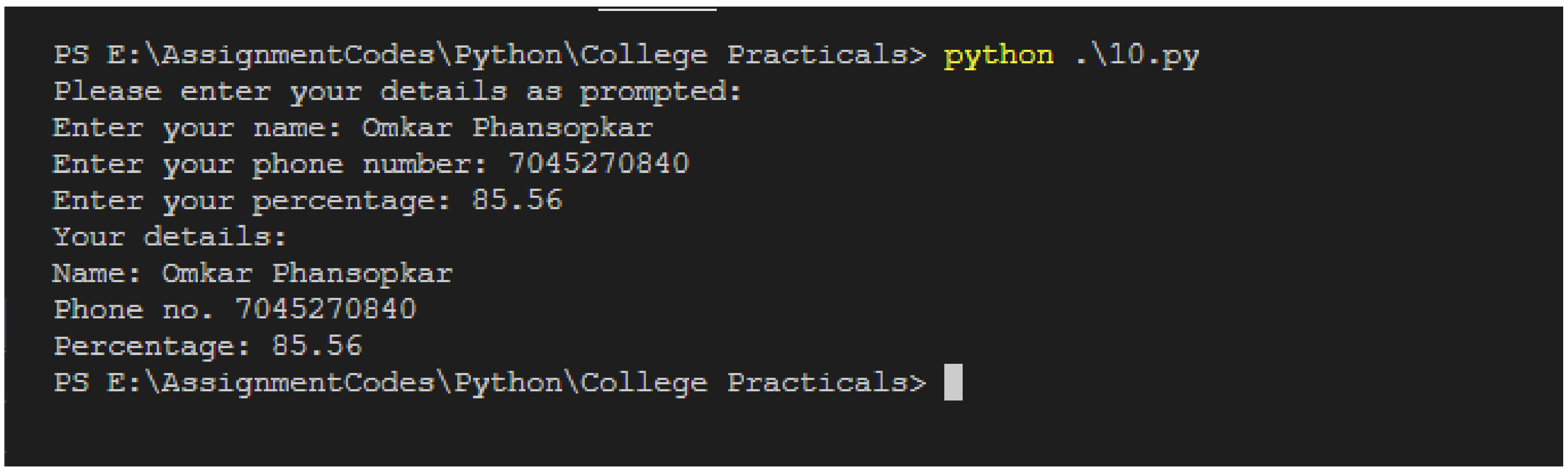
  These functions are used to convert variables to a specific type.
  These are very useful when accepting input and converting them to desired datatype.
  Some examples:

  - **int(x)**
    Converts x variable to an integer and returns it.
  - **float(x)**
    Converts x variable to a float type and returns it.
  - **List(x)**
    Converts variable to list, for example a string to list of characters in it.

Code and outputs:

```
print("Please enter your details as prompted: ")


name = input("Enter your name: ")
phone = int(input("Enter your phone number: "))
percentage = float(input("Enter your percentage: "))


print("Your details: ")
print("Name:", name)
print("Phone no.", phone)
print("Percentage:", percentage)
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python .\10.py
Please enter your details as prompted:
Enter your name: Omkar Phansopkar
Enter your phone number: 7045270840
Enter your percentage: 85.56
Your details:
Name: Omkar Phansopkar
Phone no. 7045270840
Percentage: 85.56
PS E:\AssignmentCodes\Python\College Practicals>
```

Conclusion: Thus, we took inputs, converted and stored them in proper formats, and later printed them.

Practical no. 11

FS19CO042

Aim: Write a python program to display a message to screen

Problem statements:

Print your biodata, including your name, Enrollment number, residential address, email id, contact number etc. to console.

Theory:

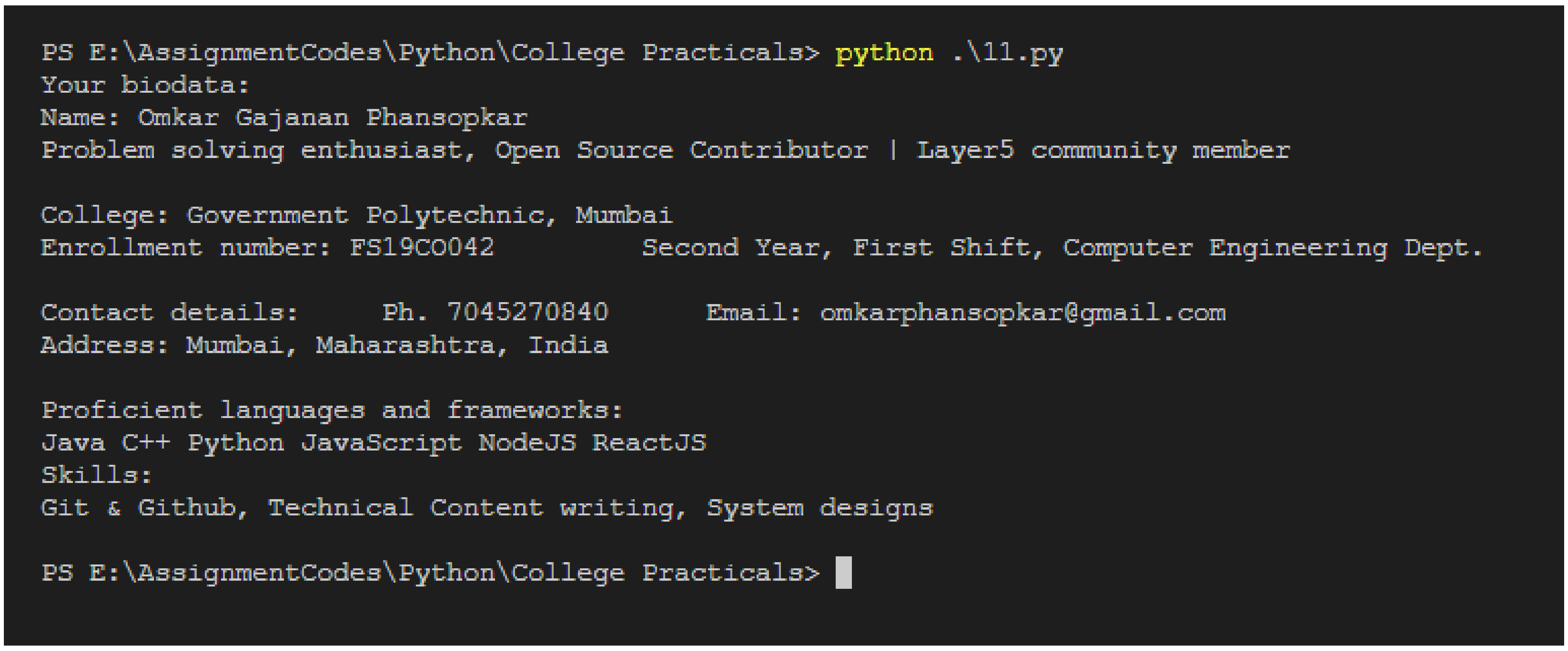
- **print()**
  Print function is used to print a string or variable to console.

- **print("string output", end="endler")**
  Print function outputs endline after string by default, to change this, end parameter can be passed, its value will be used at the end instead of newline character.

Code and outputs:

```
print("Your biodata: ")
print("Name: Omkar Gajanan Phansopkar")
print("Problem solving enthusiast, Open Source Contributor | Layer5 community member", end="\n\n")

print("College: Government Polytechnic, Mumbai")
print("Enrollment number: FS19CO042        Second Year, First Shift, Computer Engineering Dept.",end="\n\n")

print("Contact details:     Ph. 7045270840      Email: omkarphansopkar@gmail.com")
print("Address: Mumbai, Maharashtra, India", end="\n\n")

print("Proficient languages and frameworks: ")
print("Java C++ Python JavaScript NodeJS ReactJS")
print("Skills: ")
print("Git & Github, Technical Content writing, System designs", end="\n\n")
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python .\11.py
Your biodata:
Name: Omkar Gajanan Phansopkar
Problem solving enthusiast, Open Source Contributor | Layer5 community member

College: Government Polytechnic, Mumbai
Enrollment number: FS19CO042         Second Year, First Shift, Computer Engineering Dept.

Contact details:     Ph. 7045270840      Email: omkarphansopkar@gmail.com
Address: Mumbai, Maharashtra, India

Proficient languages and frameworks:
Java C++ Python JavaScript NodeJS ReactJS
Skills:
Git & Github, Technical Content writing, System designs

PS E:\AssignmentCodes\Python\College Practicals>
```

Conclusion: Thus, we displayed our resume successfully using Python.

Practical no. 12

FS19CO042

Aim: Write a python program to demonstrate use of looping statements

Problem statements:

1. Print all even numbers between 1 to 100
2. Create a simple list and iterate it with loop
3. Print pattern,

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

Theory:

- **Loops**

  Loops are used to repeat a portion of code for specified number of times or until the specified condition is met. It is used to perform repetitive tasks.

  Loop statements in python:

  - **While loop**

    Repeats a statement/group of statements until the given condition return True. The condition is checked before executing statements in body of loop. If the condition is false at very first iteration, loop body will never be executed.

    Example:
    ```
    i=1
    while i<6:
        print(i, end=” ”)
        i+=1
    ```
    Output: 1 2 3 4 5

  - **For loop**

    A for loop is generally used when number of iterations is known beforehand (either hardcoded or based on value of a variable). It is also used as an iterator for each element in a sequence eg. List tuple, set, string, etc.

    Example1:
    ```
    for x in range(3, 12, 2):
        print(x)
    ```
    Output: 3 5 7 9 11

    Example2:
    ```
    names = [“Jonas”, “Martha”, “Katharine”, “Bartosz”]
    for x in  names:
        print(x, end=” “)
    ```
    Output: Jonas Martha Katharine Bartosz

- **Nested loops**

  Loops can be nested inside another loop to achieve certain situations.
  In such loops, inner loop executes (whatever no. of its iterations) every time outer loop iterates.
  i.e.outerloop:

```
    --------
    innerloop:
        --------
        --------
```

Example:

```
for x in range(1, 3):
    for y in range(1, x):
        print(y, end=" ")
    print()
```

Output:

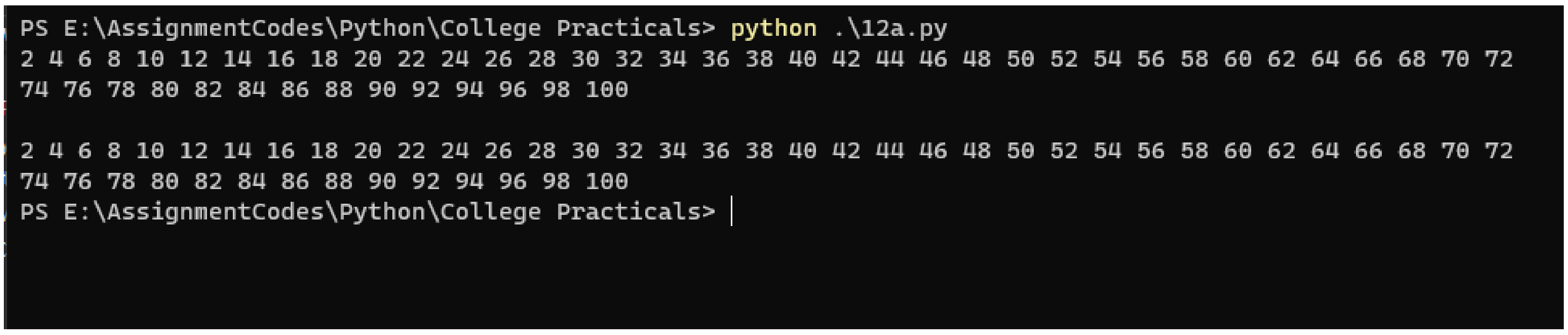
```
1
1 2
1 2 3
```

Code and outputs:

1. Print all even numbers between 1 to 100

```
# Method1. Using for loop
for i in range(2,101,2):
   print(i,end=" ")
print()
print()

# Method2. Using while loop
n = 2;
while(n<= 100):
   print(n,end=" ")
   n += 2
print()
```

Output:

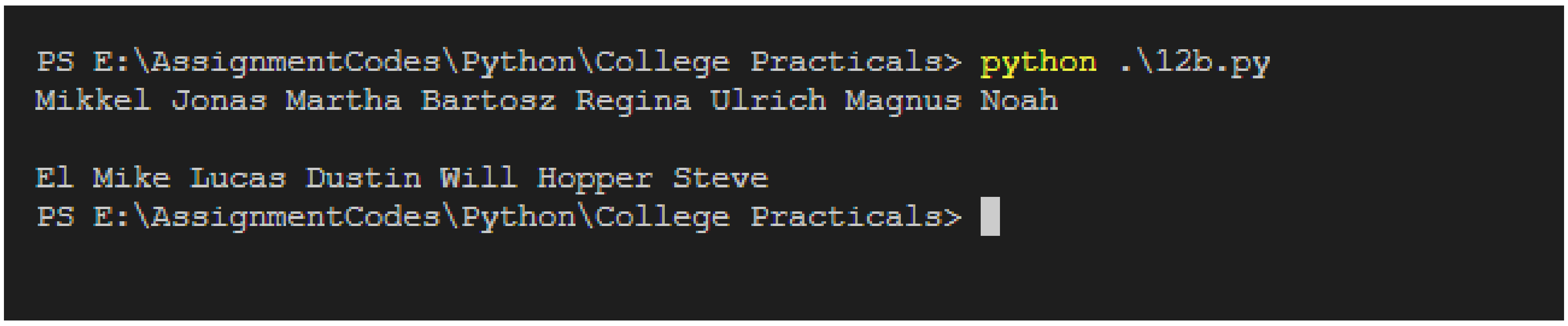


```
PS E:\AssignmentCodes\Python\College Practicals> python .\12a.py
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72
74 76 78 80 82 84 86 88 90 92 94 96 98 100

2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72
74 76 78 80 82 84 86 88 90 92 94 96 98 100
PS E:\AssignmentCodes\Python\College Practicals>
```

2. Create a simple list and iterate it with loop.

```
ST = ["El", "Mike", "Lucas", "Dustin", "Will", "Hopper", "Steve"]
Dark = ["Mikkel", "Jonas", "Martha", "Bartosz", "Regina", "Ulrich", "Magnus", "Noah"]

# Iterating using for loop
for character in Dark:
   print(character, end=" ")
print()
print()
# Iterating using indexed for loop
for x in range(len(ST)):
       print(ST[x], end=" ")
print()
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python .\12b.py
Mikkel Jonas Martha Bartosz Regina Ulrich Magnus Noah

El Mike Lucas Dustin Will Hopper Steve
PS E:\AssignmentCodes\Python\College Practicals>
```

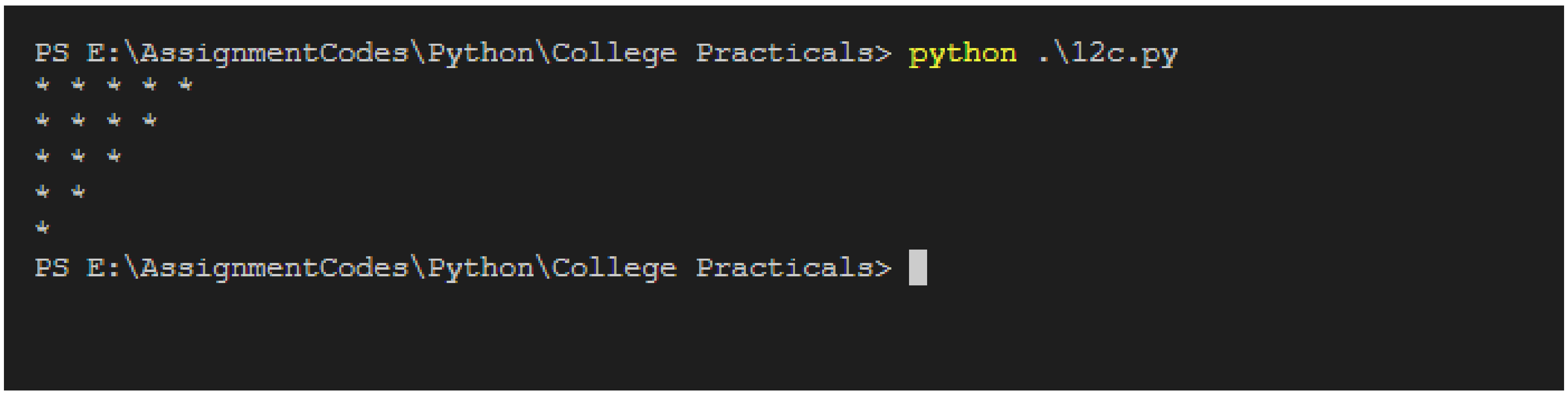
3. Print pattern,

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

Code:

```
n = 5
for i in range(n):
   for j in range(n-i):
      print("*",end=" ")
   print("")
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals> python .\12c.py
* * * * *
* * * *
* * *
* *
*
PS E:\AssignmentCodes\Python\College Practicals> 
```

<u>Conclusion:</u> Thus we understood and implemented looping statements in Python.

Practical no. 13 **FS19CO042**

Aim: Write a python program to demonstrate the use of built-in packages & user defined packages.

Problem statements:

1. Use any two built in packages
2. Create and use user defined packages

Theory:

- **Built in packages**

  Python provides related built in functions packed together inside a package which are provided by default with Python interpreter and can be directly imported.

  Examples of built-in packages: math, random, zipfile, gzip, etc.

  Examples for importing packages:

```
import math               # import whole module accessible as math.pi, math.sqrt, etc.
from math import sqrt     # import specific function
from math import *        # import all functions
```

- **Creating and using user defined packages**

  Create a new folder which is later used as package name and create file named __init__.py inside that folder. Now we can put as many python files(modules) in this folder.

  Examples for accessing these modules:

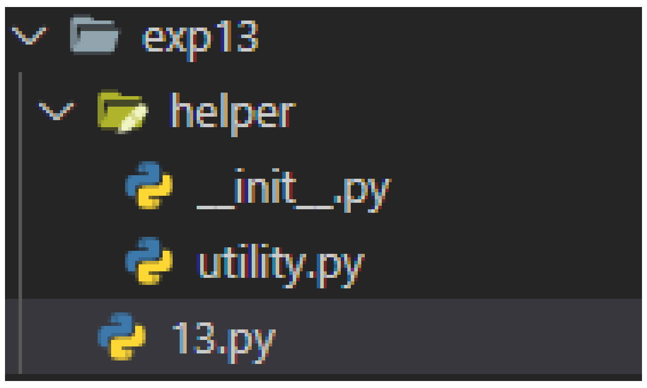
```
from abc import moduleName
from abc.moduleName import functionName
```

Code and outputs:

1. Use any two built in packages

2. Create an use user defined packages

Directory structure:



```
exp13
  helper
    __init__.py
    utility.py
  13.py
```

Files:

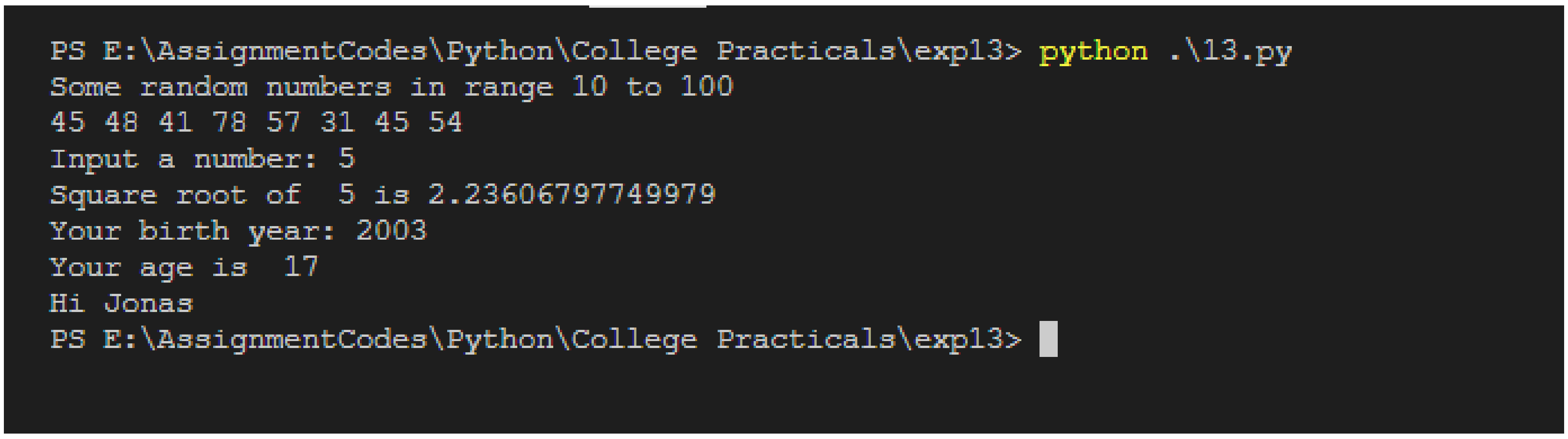
Utility.py

```
def  sayHello(name):
   print("Hi",name)
def calculateAge(year):
   return 2020-year
```

13.py

```
# Importing whole package
import random
print("Some random numbers in range 10 to 100")
for x in range(8):
    print(random.randint(15, 100), end=" ")
print()
# Importing specific function
from math import sqrt
num = int(input("Input a number: "))
print("Square root of ", num, "is", sqrt(num))

# Importing and using User defined packages
from helper import utility
birthYear = int(input("Your birth year: "))
print("Your age is ", utility.calculateAge(birthYear))

from helper.utility import sayHello
sayHello("Jonas")
```

Output:



```
PS E:\AssignmentCodes\Python\College Practicals\exp13> python .\13.py
Some random numbers in range 10 to 100
45 48 41 78 57 31 45 54
Input a number: 5
Square root of  5 is 2.23606797749979
Your birth year: 2003
Your age is  17
Hi Jonas
PS E:\AssignmentCodes\Python\College Practicals\exp13>
```

<u>**Conclusion:**</u> Thus, we understood how to use packages (built in as well as user defined).