**Practical no. 1**

**FS19CO042**

**Aim:** Install & configure Python IDE

**Theory:**

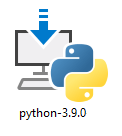
**Step 1: Download Python 3.9**

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

****

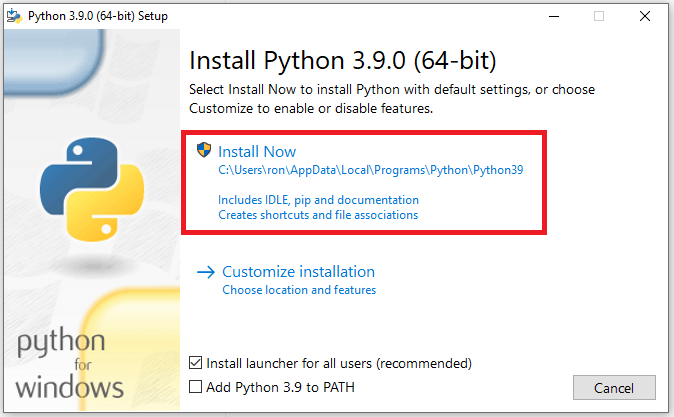
**Step 2: Run the .exe file**

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

****

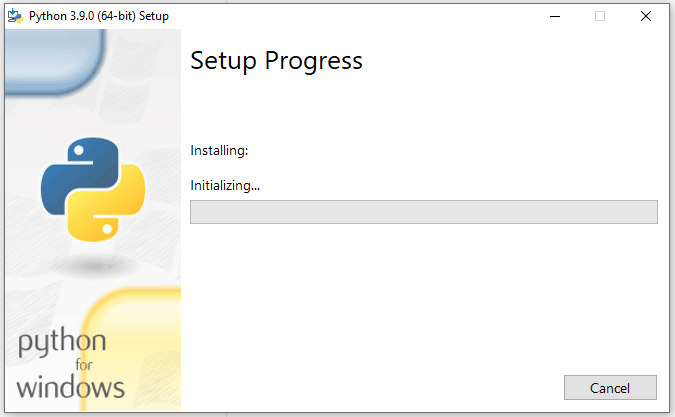
**Step 3: Install Python 3.9**

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

****

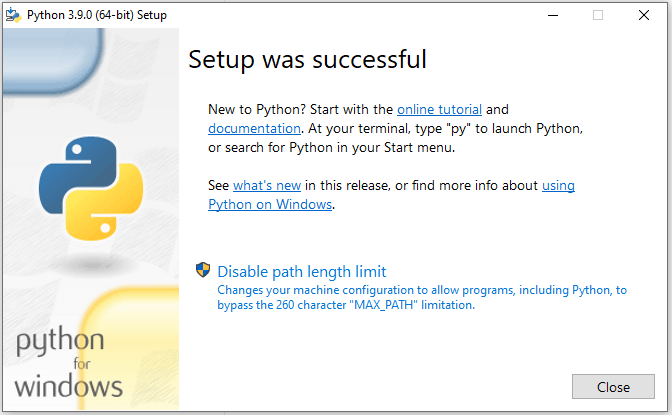
**Note that depending on your needs, you may also check the box to add Python to the Path.**

**Your installation should now begin:**

****

**Setup Progress**

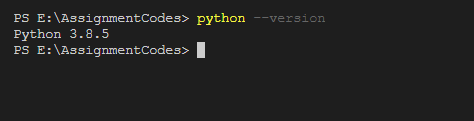
**After a short period of time, your setup would be completed:**

****

**Successful Setup**

**Congrats, you just installed Python on Windows!**

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

****

**Conclusion:** 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)** |

## 

## **Assignment operators :**

**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**](https://www.programiz.com/python-programming/string)**,** [**list**](https://www.programiz.com/python-programming/list)**,** [**tuple**](https://www.programiz.com/python-programming/tuple)**,** [**set**](https://www.programiz.com/python-programming/set) **and** [**dictionary**](https://www.programiz.com/python-programming/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)

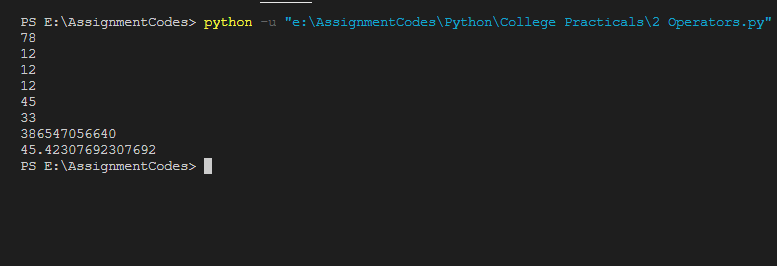
print(a&b)

print(a<<b)

d = a+b/c

print(d)

**Output:**



**Conclusion:** 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()](https://www.w3schools.com/python/ref_math_acos.asp) | Returns the arc cosine of a number |
| [math.acosh()](https://www.w3schools.com/python/ref_math_acosh.asp) | Returns the inverse hyperbolic cosine of a number |
| [math.asin()](https://www.w3schools.com/python/ref_math_asin.asp) | Returns the arc sine of a number |
| [math.asinh()](https://www.w3schools.com/python/ref_math_asinh.asp) | Returns the inverse hyperbolic sine of a number |
| [math.atan()](https://www.w3schools.com/python/ref_math_atan.asp) | Returns the arc tangent of a number in radians |
| [math.atan2()](https://www.w3schools.com/python/ref_math_atan2.asp) | Returns the arc tangent of y/x in radians |
| [math.atanh()](https://www.w3schools.com/python/ref_math_atanh.asp) | Returns the inverse hyperbolic tangent of a number |
| [math.ceil()](https://www.w3schools.com/python/ref_math_ceil.asp) | Rounds a number up to the nearest integer |
| [math.comb()](https://www.w3schools.com/python/ref_math_comb.asp) | Returns the number of ways to choose k items from n items without repetition and order |
| [math.copysign()](https://www.w3schools.com/python/ref_math_copysign.asp) | Returns a float consisting of the value of the first parameter and the sign of the second parameter |
| [math.cos()](https://www.w3schools.com/python/ref_math_cos.asp) | Returns the cosine of a number |
| [math.cosh()](https://www.w3schools.com/python/ref_math_cosh.asp) | Returns the hyperbolic cosine of a number |
| [math.degrees()](https://www.w3schools.com/python/ref_math_degrees.asp) | Converts an angle from radians to degrees |
| [math.dist()](https://www.w3schools.com/python/ref_math_dist.asp) | Returns the Euclidean distance between two points (p and q), where p and q are the coordinates of that point |
| [math.erf()](https://www.w3schools.com/python/ref_math_erf.asp) | Returns the error function of a number |
| [math.erfc()](https://www.w3schools.com/python/ref_math_erfc.asp) | Returns the complementary error function of a number |
| [math.exp()](https://www.w3schools.com/python/ref_math_exp.asp) | Returns E raised to the power of x |
| [math.expm1()](https://www.w3schools.com/python/ref_math_expm1.asp) | Returns Ex - 1 |
| [math.fabs()](https://www.w3schools.com/python/ref_math_fabs.asp) | Returns the absolute value of a number |
| [math.factorial()](https://www.w3schools.com/python/ref_math_factorial.asp) | Returns the factorial of a number |
| [math.floor()](https://www.w3schools.com/python/ref_math_floor.asp) | Rounds a number down to the nearest integer |
| [math.fmod()](https://www.w3schools.com/python/ref_math_fmod.asp) | Returns the remainder of x/y |
| [math.frexp()](https://www.w3schools.com/python/ref_math_frexp.asp) | Returns the mantissa and the exponent, of a specified number |
| [math.fsum()](https://www.w3schools.com/python/ref_math_fsum.asp) | Returns the sum of all items in any iterable (tuples, arrays, lists, etc.) |
| [math.gamma()](https://www.w3schools.com/python/ref_math_gamma.asp) | Returns the gamma function at x |
| [math.gcd()](https://www.w3schools.com/python/ref_math_gcd.asp) | Returns the greatest common divisor of two integers |
| [math.hypot()](https://www.w3schools.com/python/ref_math_hypot.asp) | Returns the Euclidean norm |
| [math.isclose()](https://www.w3schools.com/python/ref_math_isclose.asp) | Checks whether two values are close to each other, or not |
| [math.isfinite()](https://www.w3schools.com/python/ref_math_isfinite.asp) | Checks whether a number is finite or not |
| [math.isinf()](https://www.w3schools.com/python/ref_math_isinf.asp) | Checks whether a number is infinite or not |
| [math.isnan()](https://www.w3schools.com/python/ref_math_isnan.asp) | Checks whether a value is NaN (not a number) or not |
| [math.isqrt()](https://www.w3schools.com/python/ref_math_isqrt.asp) | Rounds a square root number downwards to the nearest integer |
| [math.ldexp()](https://www.w3schools.com/python/ref_math_ldexp.asp) | Returns the inverse of [math.frexp()](https://www.w3schools.com/python/ref_math_frexp.asp) which is x \* (2\*\*i) of the given numbers x and i |
| [math.lgamma()](https://www.w3schools.com/python/ref_math_lgamma.asp) | Returns the log gamma value of x |
| [math.log()](https://www.w3schools.com/python/ref_math_log.asp) | Returns the natural logarithm of a number, or the logarithm of number to base |
| [math.log10()](https://www.w3schools.com/python/ref_math_log10.asp) | Returns the base-10 logarithm of x |
| [math.log1p()](https://www.w3schools.com/python/ref_math_log1p.asp) | Returns the natural logarithm of 1+x |
| [math.log2()](https://www.w3schools.com/python/ref_math_log2.asp) | Returns the base-2 logarithm of x |
| [math.perm()](https://www.w3schools.com/python/ref_math_perm.asp) | Returns the number of ways to choose k items from n items with order and without repetition |
| [math.pow()](https://www.w3schools.com/python/ref_math_pow.asp) | Returns the value of x to the power of y |
| [math.prod()](https://www.w3schools.com/python/ref_math_prod.asp) | Returns the product of all the elements in an iterable |
| [math.radians()](https://www.w3schools.com/python/ref_math_radians.asp) | Converts a degree value into radians |
| [math.remainder()](https://www.w3schools.com/python/ref_math_remainder.asp) | Returns the closest value that can make numerator completely divisible by the denominator |
| [math.sin()](https://www.w3schools.com/python/ref_math_sin.asp) | Returns the sine of a number |
| [math.sinh()](https://www.w3schools.com/python/ref_math_sinh.asp) | Returns the hyperbolic sine of a number |
| [math.sqrt()](https://www.w3schools.com/python/ref_math_sqrt.asp) | Returns the square root of a number |
| [math.tan()](https://www.w3schools.com/python/ref_math_tan.asp) | Returns the tangent of a number |
| [math.tanh()](https://www.w3schools.com/python/ref_math_tanh.asp) | Returns the hyperbolic tangent of a number |
| [math.trunc()](https://www.w3schools.com/python/ref_math_trunc.asp) | Returns the truncated integer parts of a number |

**Code and outputs:**

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

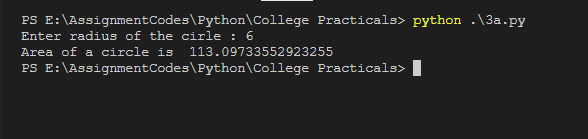
from math import pi

radius = int(input("Enter radius of the cirle : "))

area = pi \* radius \* radius

print("Area of a circle is ", area)

**Output:**



1. **Use any two built in math functions**

import math

x = 53.843

p = 15

q = 70

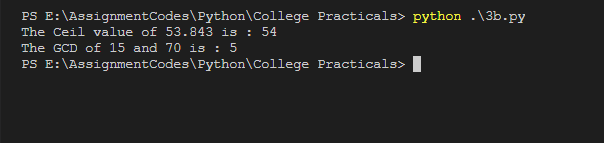
# returning the ceil value i.e. rounds no. to upper value

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:**

****

1. **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

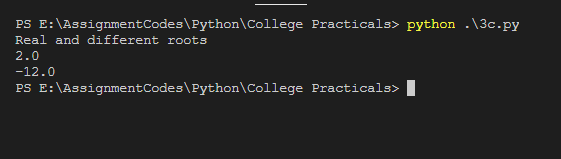
if a == 0:

print("Input correct quadratic equation")

else:

equationroots(a, b, c)

**Output:**

****

**Conclusion:** Thus we understood and used functions from math module

**Practical no. 4**

**Aim:** Write a Python program to perform following operations on list:

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

**Problem statements:**

Create a list of 5 strings and perform various operations like create, access, update, delete, etc

**Theory:**

* **List**

List are used to store multiple items in a single variable. Lists are one of 4 built-in data types in Python used to store collections of data.

Example:

listName =[“abc”, “cde”, “xyz”]

print(listName)

* **Creating list from list() constructor**

List constructor is used to create list from already defined variables like tuples, sets, etc.

Example:

tuple = (“abc”, “cde”, “efg”)

lst = list(tuple)

print(lst)

* Functions available for list

**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: '))

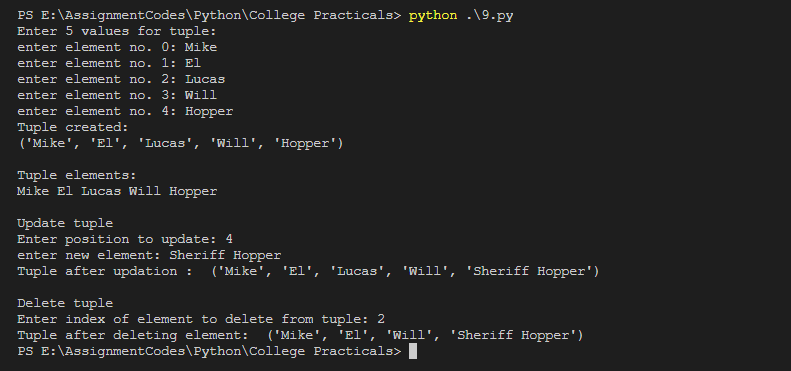
tempList = list(tup1)

tempList.pop(idx)

tup1=tuple(tempList)

print('Tuple after deleting element: ', tup1)

**Output:**



**Conclusion:** Thus, we performed various operations on tuples

**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: "))

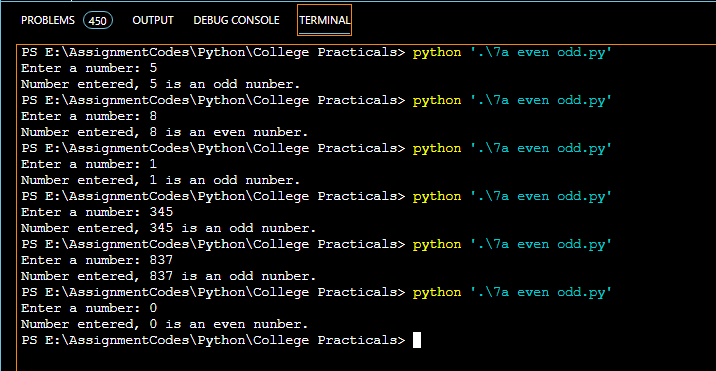
if number%2 == 0:

print("Number entered,", number, "is an even nunber.")

else:

print("Number entered,", number, "is an odd nunber.")

Output:



1. **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**

# 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")

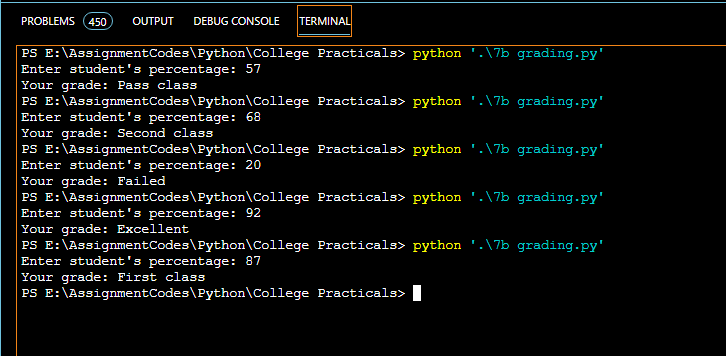
elif percentage >= 40:

print("Pass class")

else:

print ("Failed")

Output:



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)) # 12print(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:

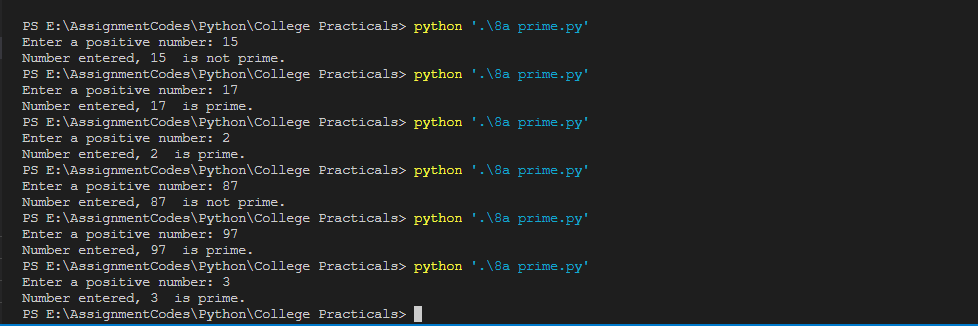
if isPrime(number):

print("Number entered,", number, " is prime.")

else:

print("Number entered,", number, " is not prime.")

**Output:**



1. **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))

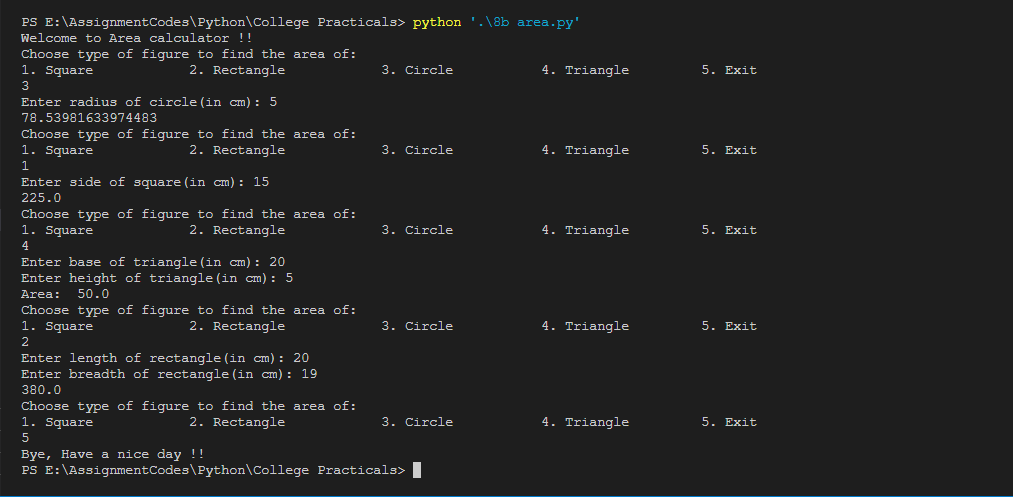
elif choice == 5:

print("Bye, Have a nice day !!")

break

else:

print("Invalid choice, try again")

**Output:**

**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:

1. 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: '))

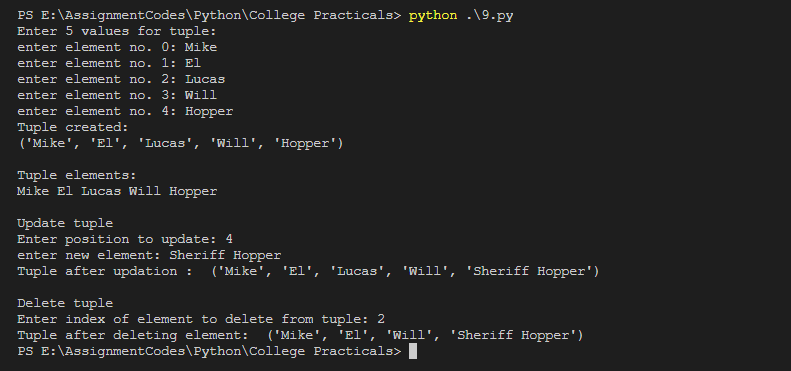
tempList = list(tup1)

tempList.pop(idx)

tup1=tuple(tempList)

print('Tuple after deleting element: ', tup1)

**Output:**



**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: "))

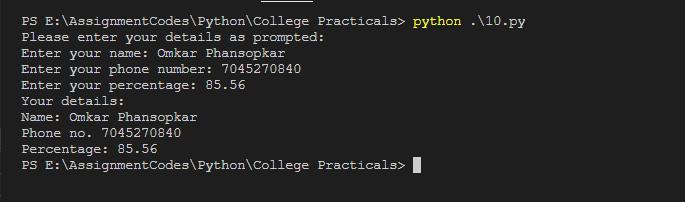
print("Your details: ")

print("Name:", name)

print("Phone no.", phone)

print("Percentage:", percentage)

**Output:**



**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")

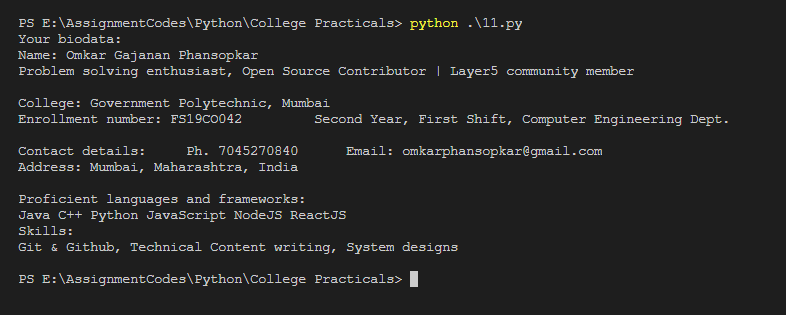
print("Proficient languages and frameworks: ")

print("Java C++ Python JavaScript NodeJS ReactJS")

print("Skills: ")

print("Git & Github, Technical Content writing, System designs", end="\n\n")

**Output:**

****

**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;

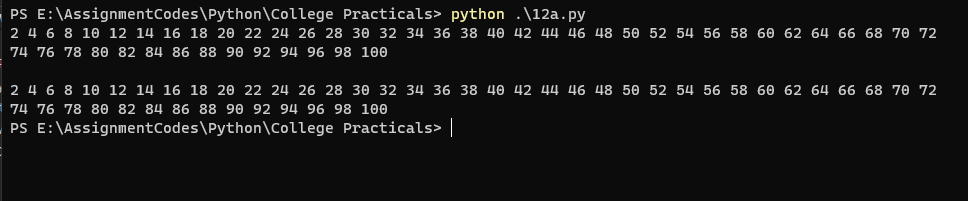
while(n<= 100):

print(n,end=" ")

n += 2

print()

**Output:**

****

1. **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()

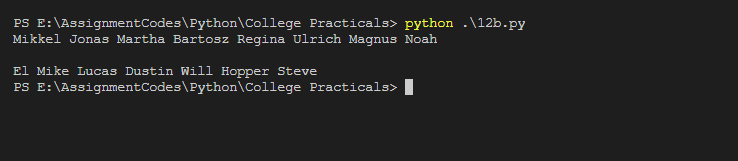
# Iterating using indexed for loop

for x in range(len(ST)):

print(ST[x], end=" ")

print()

**Output:**

****

1. **Print pattern,**

**\*\*\*\*\***

**\*\*\*\***

**\*\*\***

**\*\***

**\***

**Code:**

**n = 5**

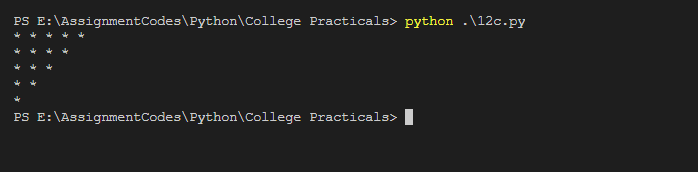
**for i in range(n):**

**for j in range(n-i):**

**print("\*",end=" ")**

**print("")**

**Output:**

****

**Conclusion:** 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:

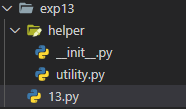
from abc import moduleName

from abc.moduleName import functionName

**Code and outputs:**

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

**Directory structure:**

****

**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

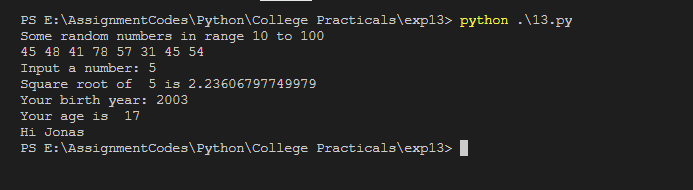
birthYear = int(input("Your birth year: "))

print("Your age is ", utility.calculateAge(birthYear))

from helper.utility import sayHello

sayHello("Jonas")

**Output:**

****

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