1.1. Python Program for Arithmetic Operations:

```
# Store input numbers:
num1 = input('Enter first number: ')
num2 = input('Enter second number: ')

# Add two numbers
sum = float(num1) + float(num2)

# Subtract two numbers
min = float(num1) - float(num2)

# Multiply two numbers
mul = float(num1) * float(num2)

# Divide two numbers
div = float(num1) / float(num2)

# Display the sum
print('The sum of {0} and {1} is {2}'.format(num1, num2, sum))

# Display the subtraction
print('The subtraction of {0} and {1} is {2}'.format(num1, num2, min))

# Display the multiplication
print('The multiplication of {0} and {1} is {2}'.format(num1, num2, mul))

# Display the division
print('The division of {0} and {1} is {2}'.format(num1, num2, div))
```

Output:

```
Enter first number: 20
Enter second number: 10
The sum of 20 and 10 is 30.0
The subtraction of 20 and 10 is 10.0
The multiplication of 20 and 10 is 200.0
The division of 20 and 10 is 2.0
```

1.2. Python Program for Arithmetic Operations:

```
def add_numbers(num1, num2):
    return float(num1) + float(num2)

def subtract_numbers(num1, num2):
    return float(num1) - float(num2)

def multiply_numbers(num1, num2):
    return float(num1) * float(num2)

def divide_numbers(num1, num2):
    return float(num1) / float(num2)

def perform_operations(num1, num2):
    # Display the sum
    print('The sum of {0} and {1} is {2}'.format(num1, num2, add_numbers(num1,
num2)))
```

```
    # Display the subtraction
    print('The subtraction of {0} and {1} is {2}'.format(num1, num2,
subtract_numbers(num1, num2)))

    # Display the multiplication
    print('The multiplication of {0} and {1} is {2}'.format(num1, num2,
multiply_numbers(num1, num2)))

    # Display the division
    print('The division of {0} and {1} is {2}'.format(num1, num2,
divide_numbers(num1, num2)))

if __name__ == "__main__":
    # Store input numbers:
    num1 = input('Enter first number: ')
    num2 = input('Enter second number: ')

    # Perform operations recursively
    perform_operations(num1, num2)
```

Output:
Enter first number: 20
Enter second number: 10
The sum of 20 and 10 is 30.0
The subtraction of 20 and 10 is 10.0
The multiplication of 20 and 10 is 200.0
The division of 20 and 10 is 2.0

- **In this version, I've replaced the direct calculations with function calls.**
- **The functions 'add_numbers', 'subtract_numbers', 'multiply_numbers', and 'divide_numbers' handle the respective operations.**
- **The 'perform_operations' function calls these functions and displays the results. This structure allows for recursion, although in this specific case, recursion isn't necessary and a more straightforward iterative approach could be used.**

2. Python Program for Area of Triangle:

```
# Python Program for Area of Triangle:
b = float(input('Enter the base: '))
h= float(input('Enter the height: '))

# calhculate the area
area = 1/2*b*h
print('The area of the triangle is %0.2f' % area)
```

Output:
Enter the base: 10
Enter the height: 5
The area of the triangle is 25.00

3.1. Python program to swap two variables:

```
P = int(input("Please enter value for P: "))
Q = int(input("Please enter value for Q: "))

# To swap the value of two variables
# we will user third variable which is a temporary variable
temp_1 = P
P = Q
Q = temp_1

print("The Value of P after swapping: ", P)
print("The Value of Q after swapping: ", Q)
```

Output:
Please enter value for P: 20
Please enter value for Q: 10
The Value of P after swapping: 10
The Value of Q after swapping: 20

4. Python Program to check if a Number is odd or even:

Odd and Even numbers:

If you divide a number by 2 and it gives a remainder of 0 then it is known as even number, otherwise an odd number.

Even number examples: 2, 4, 6, 8, 10, etc.

Odd number examples: 1, 3, 5, 7, 9 etc.

```
num = int(input("Enter a number: "))
if (num % 2) == 0:
   print("{0} is Even number".format(num))
else:
   print("{0} is Odd number".format(num))
```

Output:
Enter a number: 10
10 is Even number

OR

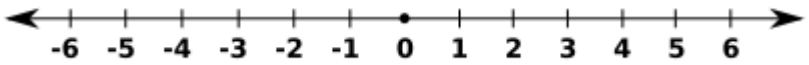
Output:
Enter a number: 15
**   15 s Odd number**

5. Python Program to check if a Number is Positive, Negative or Zero

Here, we will learn to write a Python program through which we can check if the given number is positive, zero, or negative.

Positive Numbers: A number is said to be positive if the number has a value greater than 0, like 1, 2, 3, 5, 7, 9, 11, 13, etc. All the natural numbers are positive numbers.

Negative Numbers: If a given number has a value less than 0 like -1, -2, -3, -5, -7, -9, -11, -13, etc., then we can say that the given number is a negative number. Only rational and integer type numbers can have negative values or numbers.




```
def NumberCheck(a):
    # Checking if the number is positive
    if a > 0:
        print("Number given by you is Positive")

    # Checking if the number is negative
    elif a < 0:
        print("Number given by you is Negative")

    # Else the number is zero
    else:
        print("Number given by you is zero")

# Taking number from user
a = float(input("Enter a number as input value: "))
# Printing result
NumberCheck(a)
```

Output:

```
Enter a number as input value: 14
Number given by you is Positive
                OR
Enter a number as input value: -12
Number given by you is Negative
                OR
Enter a number as input value: 0
Number given by you is zero
```

Explanation:

- We have used a nested if else condition in the program to check the number.
- When the user gives an input number, the program will first check if the value of the number is greater than 0 (if yes, it will print Positive and the program ends),
- Otherwise it will check if the value is less than 0 (if yes, it will print Negative and the program ends), and
- At last it will print that number is 0 if the number entered is 0.

6. Python program to display calendar:

It is simple in python programming to display calendar. To do so, you need to import the calendar module which comes with Python.

```
import calendar

# Enter the month and year
yy = int(input("Enter year: "))
mm = int(input("Enter month: "))

# display the calendar
print(calendar.month(yy, mm))
```

Output:

```
Enter year: 2022
Enter month: 10

     October 2022
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
```

7. Python Program to Find the Square Root:

```
# Python Program to Find the Square Root
n=int(input('Enter no.:  '))
def sqr_root():
    return n**0.5
print(sqr_root())
```

```
Output:
Enter no.:  81
9.0
```

8. Python Program to Check Prime Number:

We will write a program here in which we will check that a given number is a prime number or not.

Prime numbers:
If the natural number is greater than 1 and having no positive divisors other than 1 and the number itself etc.
For example: 3, 7, 11 etc. are prime numbers.

Composite number:
Other natural numbers that are not prime numbers are called composite numbers.
For example: 4, 6, 9 etc. are composite numbers.

```
# A default function for Prime checking conditions
def PrimeChecker(a):
    # Checking that given number is more than 1
    if a > 1:
        # Iterating over the given number with for loop
        for j in range(2, int(a/2) + 1):
            # If the given number is divisible or not
            if (a % j) == 0:
                print(a, "is not a prime number")
                break
        # Else it is a prime number
        else:
            print(a, "is a prime number")
    # If the given number is 1
    else:
        print(a, "is not a prime number")


# Taking an input number from the user
a = int(input("Enter an input number:"))

# Printing result
PrimeChecker(a)
```

Output:

```
Enter an input number: 13
13 is a prime number
          OR
Enter an input number: 20
20 is not a prime number
```

Explanation:

- We have used nested if else condition to check if the number is a prime number or not.
- First, we have checked if the given number is greater than 1 or not. If it is not greater than 1, then the number will directly come to the else part and print 'not a prime number.'
- Now, the number will enter into for loop where we perform Iteration from 2 to number/2. Then, we used nested if else condition inside for loop. If the number is completely divisible by 'i' then, it is not a prime number; else, the number is a prime number.

9. Python Program to Find the Factorial of a Number:

What is factorial?

Factorial is a non-negative integer. It is the product of all positive integers less than or equal to that number you ask for factorial. It is denoted by an exclamation sign (!).

Example:

- n! = n* (n-1) * (n-2) *.....1
- 4! = 4x3x2x1 = 24

```
num = int(input("Enter a number: "))
factorial = 1
if num < 0:
   print(" Factorial does not exist for negative numbers")
elif num == 0:
   print("The factorial of 0 is 1")
else:
   for i in range(1,num + 1):
       factorial = factorial*i
   print("The factorial of",num,"is",factorial)
```

Output:

```
Enter a number: 5
The factorial of 5 is 120
```

10. Python Program to print all Prime Numbers in an Interval:

A prime number is a natural number which is greater than 1 and has no positive divisor other than 1 and itself, such as 2, 3, 5, 7, 11, 13, and so on.

The user is given two integer numbers, lower value, and upper value. The task is to write the Python program for printing all the prime numbers between the given interval (or range)

.

To print all the prime numbers between the given intervals, the user has to follow the following steps:

Step 1: Loop through all the elements in the given range.

Step 2: Check for each number if it has any factor between 1 and itself.

Step 3: If yes, then the number is not prime, and it will move to the next number.

Step 4: If no, it is the prime number, and the program will print it and check for the next number.

Step 5: The loop will break when it is reached to the upper value.

```
# First, we will take the input:
lower_value = int(input("Please, Enter the Lowest Range Value: "))
upper_value = int(input("Please, Enter the Upper Range Value: "))

print("The Prime Numbers in the range are: ")
for number in range(lower_value, upper_value + 1):
    if number > 1:
        for i in range(2, number):
            if (number % i) == 0:
                break
        else:
            print(number)
```

```
Output:
Please, Enter the Lowest Range Value: 30

Please, Enter the Upper Range Value: 90

The Prime Numbers in the range are:

31

37

41

43

47

53

59

61

67

71

73

79

83

89
```

10. Python Program to check if a Number is a Palindrome

This is a Python Program to check whether a given number is a palindrome.

Problem Description
The program takes a number and checks whether it is a palindrome or not.

Problem Solution
1. Take the value of the integer and store in a variable.
2. Transfer the value of the integer into another temporary variable.
3. Using a while loop, get each digit of the number and store the reversed number in another variable.
4. Check if the reverse of the number is equal to the one in the temporary variable.
5. Print the final result.
6. Exit.

Program/Source Code:
Here is source code of the Python Program to check whether a given number is a palindrome. The program output is also shown below.

```
n=int(input("Enter number:"))
temp=n
rev=0
while(n>0):
    dig=n%10
    rev=rev*10+dig
    n=n//10
```

```
if(temp==rev):
    print("The number is a palindrome!")
else:
    print("The number isn't a palindrome!")
```

Program Explanation

1. User must first enter the value of the integer and store it in a variable.
2. The value of the integer is then stored in another temporary variable.
3. The while loop is used and the last digit of the number is obtained by using the modulus operator.
4. The last digit is then stored at the one's place, second last at the ten's place and so on.
5. The last digit is then removed by truly dividing the number with 10.
6. This loop terminates when the value of the number is 0.
7. The reverse of the number is then compared with the integer value stored in the temporary variable.
8. If both are equal, the number is a palindrome.
9. If both aren't equal, the number isn't a palindrome.
10. The final result is then printed.

11. String Palindrome Program in Python

This is a Python Program to check if a string is a palindrome or not.

Problem Description

The program takes a string and checks if a string is a palindrome or not.

Problem Solution

1. Take a string from the user and store it in a variable.
2. Reverse the string using string slicing and compare it back to the original string.
3. Print the final result
4. Exit.

Program/Source Code

Here is source code of the Python Program to check if a string is a palindrome or not. The program output is also shown below.

```
string=input(("Enter a letter:"))
if(string==string[::-1]):
    print("The letter is a palindrome")
else:
    print("The letter is not a palindrome")
```

Program Explanation

1. User must enter a string and store it in a variable.
2. The string is reversed using string slicing and it is compared back to the non-reversed string.

3. If they both are equal, the strings are palindromes.
4. If they aren't equal, the strings aren't palindromes.
6. The final result is printed.

12. Python Program to Calculate Grade of a Student:

This is a Python Program to take in the marks of 5 subjects and display the grade.

Problem Description:

The program takes in the marks of 5 subjects and displays the grade.

Problem Solution:

1. Take in the marks of 5 subjects from the user and store it in different variables.
2. Find the average of the marks.
3. Use an else condition to decide the grade based on the average of the marks.
4. Exit.

```
print("Enter Marks Obtained in 5 Subjects: ")
markOne = int(input())
markTwo = int(input())
markThree = int(input())
markFour = int(input())
markFive = int(input())

tot = markOne+markTwo+markThree+markFour+markFive
avg = tot/5

if avg>=91 and avg<=100:
   print("Your Grade is A1")
elif avg>=81 and avg<91:
   print("Your Grade is A2")
elif avg>=71 and avg<81:
   print("Your Grade is B1")
elif avg>=61 and avg<71:
   print("Your Grade is B2")
elif avg>=51 and avg<61:
   print("Your Grade is C1")
elif avg>=41 and avg<51:
   print("Your Grade is C2")
elif avg>=33 and avg<41:
   print("Your Grade is D")
elif avg>=21 and avg<33:
   print("Your Grade is E1")
```

```
elif avg>=0 and avg<21:
   print("Your Grade is E2")
else:
   print("Invalid Input!")
```

Output:

Enter Marks Obtained in 5 Subjects:
90
80
85
84
80
Your Grade is A2

13. Python Program for simple interest

Simple interest formula is given by: Simple Interest = (P x T x R)/100 Where, P is the principal amount T is the time and R is the rate

Examples:

EXAMPLE1:

Input: P = 10000

R = 5

T = 5

Output: 2500.0

We need to find simple interest on Rs. 10,000 at the rate of 5% for 5 Units of time.

EXAMPLE 2:

Input: P = 3000, R = 7, T = 1

Output: 210.0

```
# Python3 program to find simple interest
# for given principal amount, time and
# rate of interest.

def simple_interest(p,t,r):
   print('The principal is', p)
   print('The time period is', t)
   print('The rate of interest is',r)

   si = (p * t * r)/100

   print('The Simple Interest is', si)
   return si

# Driver code
simple_interest(8, 6, 8)
```

Output:

The principal is 8
The time period is 6
The rate of interest is 8
The Simple Interest is 3.84

14. Python Program for compound interest

Let us discuss the formula for compound interest. The formula to calculate compound interest annually is given by:

$A = P (1 + R/100)^{T}$
Compound Interest = A – P

Where,
A is amount
P is the principal amount
R is the rate and
T is the time span

```
# Python3 program to find compound
# interest for given values.


def compound_interest(principal, rate, time):

    # Calculates compound interest
    Amount = principal * (pow((1 + rate / 100), time))
    CI = Amount - principal
    print("Compound interest is", CI)


# Driver Code
compound_interest(10000, 10.25, 5)
```

Output:

Compound interest is 6288.946267774416

15. Python program to copy all elements of one array into another array

In this program, we need to copy all the elements of one array into another. This can be accomplished by looping through the first array and store the elements of the first array into the second array at the corresponding position.

ALGORITHM:

- **STEP 1:** Declare and initialize an array.
- **STEP 2:** Declare another array of the same size as of the first one
- **STEP 3:** Loop through the first array from 0 to length of the array and copy an element from the first array to the second array that is arr1[i] = arr2[i].

PROGRAM:

```
#Initialize array
arr1 = [1, 2, 3, 4, 5];

#Create another array arr2 with size of arr1
arr2 = [None] * len(arr1);



 #Copying all elements of one array into another
for i in range(0, len(arr1)):
    arr2[i] = arr1[i];

#Displaying elements of array arr1
print("Elements of original array: ");
for i in range(0, len(arr1)):
    print(arr1[i]),

    print();

#Displaying elements of array arr2
print("Elements of new array: ");
for i in range(0, len(arr2)):
    print(arr2[i])
```

Output:

```
Elements of original array
1 2 3 4 5
Elements of new array:
1 2 3 4 5
```

16. Python program to print the sum of all elements in an array

In this program, we need to calculate the sum of all the elements of an array. This can be solved by looping through the array and add the value of the element in each iteration to variable sum.

1	2	3	4	5

Sum of all elements of an array is 1 + 2 + 3 + 4 + 5 = 15.

ALGORITHM:

STEP 1: Declare and initialize an array.

STEP 2: The variable sum will be used to calculate the sum of the elements. Initialize it to 0.

STEP 3: Loop through the array and add each element of the array to the variable sum as sum = sum + arr[i].

PROGRAM:

```
#Initialize array
arr = [1, 2, 3, 4, 5];
sum = 0;

#Loop through the array to calculate sum of elements
for i in range(0, len(arr)):
    sum = sum + arr[i];

print("Sum of all the elements of an array: " + str(sum));
```

Output:

```
Sum of all the elements of an array: 15
```

17. Python program to sort the elements of an array in ascending order

In this program, we need to sort the given array in ascending order such that elements will be arranged from smallest to largest. This can be achieved through two loops. The outer loop will select an element, and inner loop allows us to compare selected element with rest of the elements.

Original Array:

5	2	8	7	1

Array after sorting:

1	2	5	7	8

Elements will be sort in such a way that smallest element will appear on extreme left which in this case is 1. The largest element will appear on extreme right which in this case is 8.

ALGORITHM:

- **STEP 1:** Declare and initialize an array.
- **STEP 2:** Loop through the array and select an element.
- **STEP 3:** The inner loop will be used to compare the selected element from the outer loop with the rest of the elements of the array.
- **STEP 4:** If any element is less than the selected element then swap the values.
- **STEP 5:** Continue this process till entire array is sorted in ascending order.

PROGRAM:

```
#Initialize array
arr = [5, 2, 8, 7, 1];
temp = 0;
```

```
#Displaying elements of original array
print("Elements of original array: ");
for i in range(0, len(arr)):
    print(arr[i], end=" ");
    print();
#Sort the array in ascending order
for i in range(0, len(arr)):
    for j in range(i+1, len(arr)):
         if(arr[i] > arr[j]):
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
    print();
#Displaying elements of the array after sorting

print("Elements of array sorted in ascending order: ");
for i in range(0, len(arr)):
    print(arr[i], end=" ");
    print();
```

Output:

Elements of original array:
5 2 8 7 1
Elements of array sorted in ascending order:
1 2 5 7 8

18. Python program to sort the elements of an array in descending order

In this program, we need to sort the given array in descending order such that elements will be arranged from largest to smallest. This can be achieved through two loops. The outer loop will select an element, and inner loop allows us to compare selected element with rest of the elements.

Original Array:

5	2	8	7	1

Array after sorting:

8	7	5	2	1

Elements will be sort in such a way that largest element will appear on extreme left which in this case is 8. The smallest element will appear on extreme right which in this case is 1.

ALGORITHM:

- **STEP 1:** Declare and initialize an array.
- **STEP 2:** Loop through the array and select an element.

- **STEP 3:** Inner loop will be used to compare selected element from the outer loop with the rest of the elements of the array.
- **STEP 4:** If any element is greater than the selected element then swap the values.
- **STEP 5:** Continue this process till the entire list is sorted in descending order.

PROGRAM:

```
#Initialize array
arr = [5, 2, 8, 7, 1];
temp = 0;

#Displaying elements of original array
print("Elements of original array: ");
for i in range(0, len(arr)):
    print(arr[i]),

#Sort the array in descending order
for i in range(0, len(arr)):
    for j in range(i+1, len(arr)):
        if(arr[i] < arr[j]):
            temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
    print();

#Displaying elements of array after sorting
print("Elements of array sorted in descending order: ");
for i in range(0, len(arr)):
    print(arr[i]),
```

Output:

```
Elements of original array:
5 2 8 7 1
Elements of array sorted in descending order:
8 7 5 2 1
```

19. To append element in the list:

Python provides built-in methods to append or add elements to the list. We can also append a list into another list. These methods are given below.

- **append(elmt)** - It appends the value at the end of the list.
- **insert(index, elmt)** - It inserts the value at the specified index position.
- **extends(iterable)** - It extends the list by adding the iterable object.

Let's understand these methods by the following example.

1. append(elmt)

This function is used to add the element at the end of the list. The example is given below.

Example –

```
names = ["Joseph", "Peter", "Cook", "Tim"]

print('Current names List is:', names)

new_name = input("Please enter a name:\n")
names.append(new_name)  # Using the append() function

print('Updated name List is:', names)
```

Output:

```
Current names List is: ['Joseph', 'Peter', 'Cook', 'Tim']
Please enter a name:
Saquib
Updated name List is: ['Joseph', 'Peter', 'Cook', 'Tim', 'Saquib']
```

2. insert(index, elmt)

The insert() function adds the elements at the given an index position. It is beneficial when we want to insert element at a specific position. The example is given below.

Example –

```
list1 = [10, 20, 30, 40, 50]

print('Current Numbers List: ', list1)

el = list1.insert(3, 77)
print("The new list is: ",list1)

n = int(input("Enter a number to add to list:\n"))

index = int(input('Enter the index to add the number:\n'))

list1.insert(index, n)

print('Updated Numbers List:', list1)
```

Output:

```
Current Numbers List:  [10, 20, 30, 40, 50]
The new list is:  [10, 20, 30, 77, 40, 50]
Enter a number to add to list:
45
Enter the index to add the number:
1
Updated Numbers List: [10, 45, 20, 30, 77, 40, 50]
```

3. extend(iterable)

The extends() function is used to add the iterable elements to the list. It accepts iterable object as an argument. Below is the example of adding iterable element.

Example –

```
list1 = [10,20,30]
list1.extend(["52.10", "43.12" ])  # extending list elements
print(list1)
list1.extend((40, 30))  # extending tuple elements
print(list1)
list1.extend("Apple")  # extending string elements
print(list1)
```

Output:

```
[10, 20, 30, '52.10', '43.12']
[10, 20, 30, '52.10', '43.12', 40, 30]
[10, 20, 30, '52.10', '43.12', 40, 30, 'A', 'p', 'p', 'l', 'e']
```