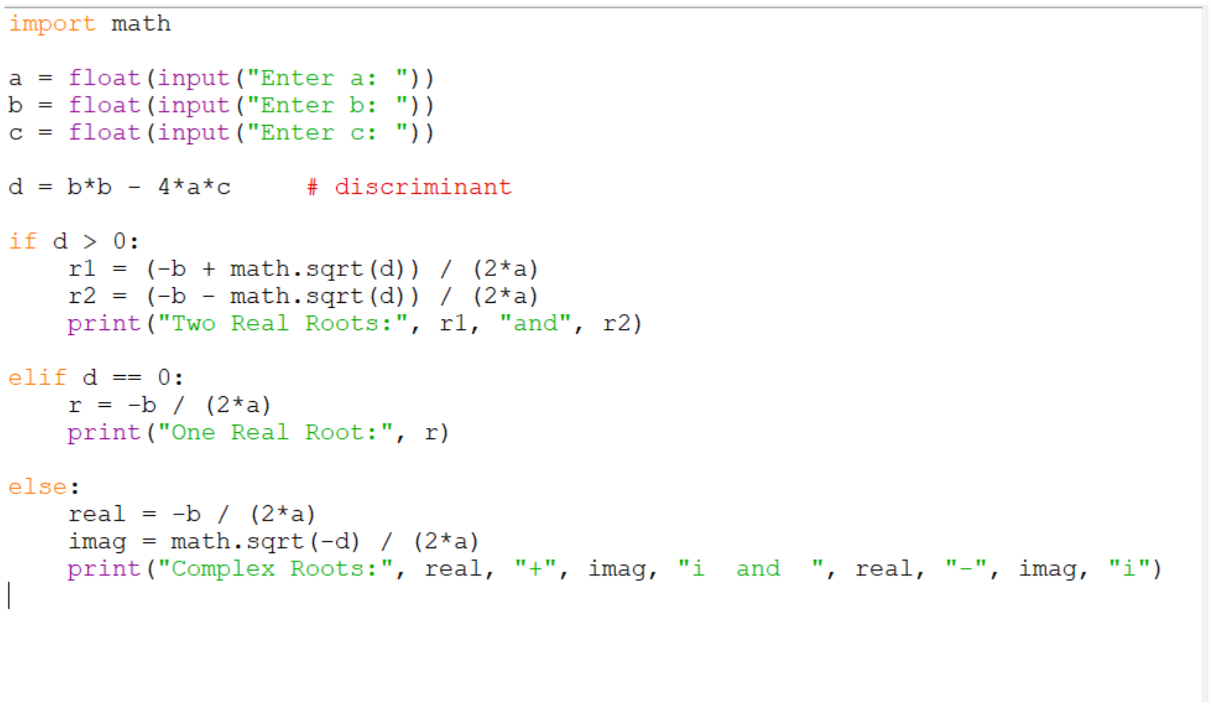
1. Write a program to find the roots of a quadratic equation



```python
import math

a = float(input("Enter a: "))
b = float(input("Enter b: "))
c = float(input("Enter c: "))

d = b*b - 4*a*c     # discriminant

if d > 0:
    r1 = (-b + math.sqrt(d)) / (2*a)
    r2 = (-b - math.sqrt(d)) / (2*a)
    print("Two Real Roots:", r1, "and", r2)

elif d == 0:
    r = -b / (2*a)
    print("One Real Root:", r)

else:
    real = -b / (2*a)
    imag = math.sqrt(-d) / (2*a)
    print("Complex Roots:", real, "+", imag, "i  and  ", real, "-", imag, "i")
```

```python
import math

a = float(input("Enter a: "))
b = float(input("Enter b: "))
c = float(input("Enter c: "))

d = b*b - 4*a*c     # discriminant

if d > 0:
    r1 = (-b + math.sqrt(d)) / (2*a)
    r2 = (-b - math.sqrt(d)) / (2*a)
    print("Two Real Roots:", r1, "and", r2)

elif d == 0:
    r = -b / (2*a)
    print("One Real Root:", r)

else:
    real = -b / (2*a)
    imag = math.sqrt(-d) / (2*a)
    print("Complex Roots:", real, "+", imag, "i  and  ", real, "-", imag, "i")
```

```
Enter a: 2
Enter b: 3
Enter c: 4
Complex Roots: -0.75 + 1.1989578808281798 i  and  -0.75 - 1.1989578808281798 i
```

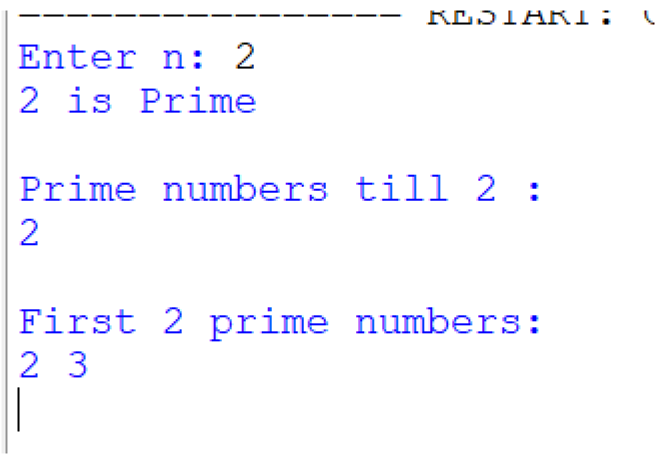
2. Write a program to accept a number ‘n’ and

a. Check if ’n’ is prime

b. Generate all prime numbers till ‘n’

c. Generate first ‘n’ prime numbers This program may be done using functions

```
def is_prime(n):
    if n < 2:
        return False
    else:
        for i in range(2, n):
            if n % i == 0:
                return False
        return True


# Take input
n = int(input("Enter n: "))

# (a) Check if n is prime
if is_prime(n):
    print(n, "is Prime")
else:
    print(n, "is NOT Prime")


# (b) Generate all prime numbers till n
print("\nPrime numbers till", n, ":")
for i in range(1, n + 1):
    if is_prime(i):
        print(i, end=" ")

print()


# (c) Generate first n prime numbers
print("\nFirst", n, "prime numbers:")
count = 0
num = 2

while count < n:
    if is_prime(num):
        print(num, end=" ")
        count += 1
    num += 1
```

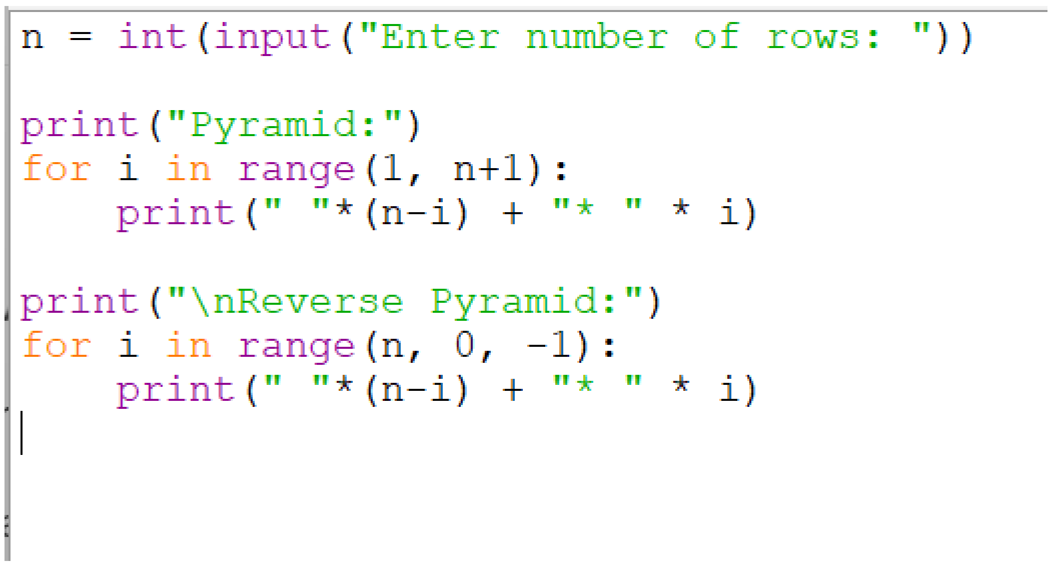


```
----------------- RESTART: (
Enter n: 2
2 is Prime

Prime numbers till 2 :
2

First 2 prime numbers:
2 3
```

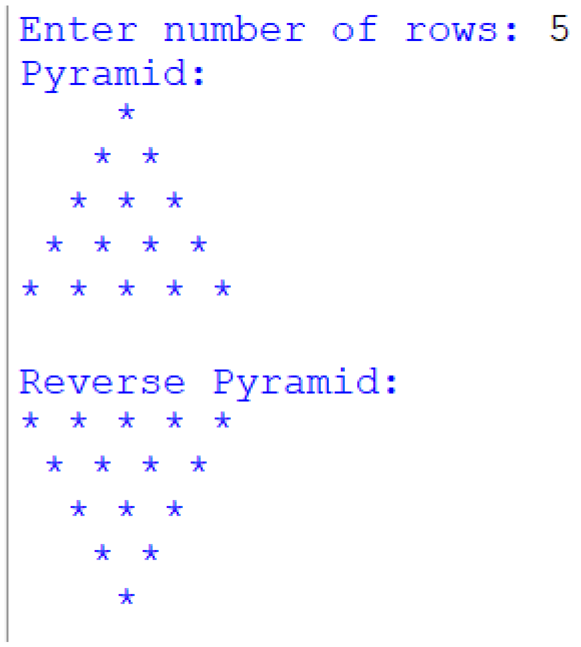
```
# Function to check if a number is prime (your easy version)
def is_prime(n):
    if n < 2:
        return False
    else:
        for i in range(2, n):
            if n % i == 0:
                return False
        return True


# Take input
n = int(input("Enter n: "))

# (a) Check if n is prime
if is_prime(n):
    print(n, "is Prime")
else:
    print(n, "is NOT Prime")


# (b) Generate all prime numbers till n
print("\nPrime numbers till", n, ":")
for i in range(1, n + 1):
    if is_prime(i):
        print(i, end=" ")

print()


# (c) Generate first n prime numbers
print("\nFirst", n, "prime numbers:")
count = 0
```

```
num = 2

while count < n:
    if is_prime(num):
        print(num, end=" ")
        count += 1
    num += 1
```

2. Write a program to create a pyramid of the character ‘*’ and a reverse pyramid



```python
n = int(input("Enter number of rows: "))

print("Pyramid:")
for i in range(1, n+1):
    print(" "*(n-i) + "* " * i)

print("\nReverse Pyramid:")
for i in range(n, 0, -1):
    print(" "*(n-i) + "* " * i)
```



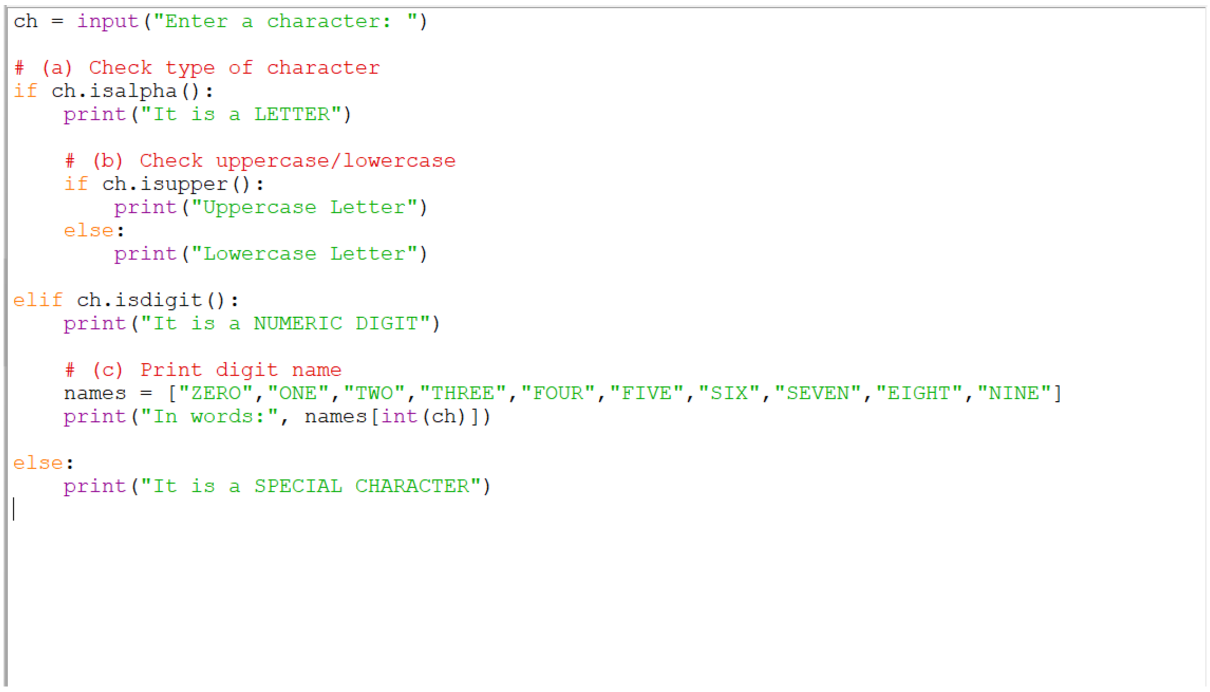
```
Enter number of rows: 5
Pyramid:
    * 
   * * 
  * * * 
 * * * * 
* * * * * 

Reverse Pyramid:
* * * * * 
 * * * * 
  * * * 
   * * 
    * 
```

```python
n = int(input("Enter number of rows: "))

print("Pyramid:")
for i in range(1, n+1):
    print(" "*(n-i) + "* " * i)

print("\nReverse Pyramid:")
for i in range(n, 0, -1):
    print(" "*(n-i) + "* " * i)
```

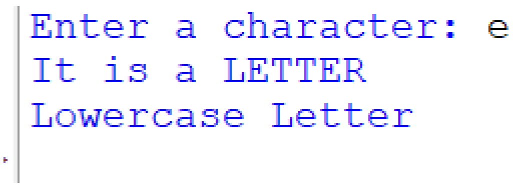
4. Write a program that accepts a character and performs the following:

a. print whether the character is a letter or numeric digit or a special character. b. if the character is a letter, print whether the letter is uppercase or lowercase

c. if the character is a numeric digit, prints its name in text (e.g., if input is 9, output is NINE)



```python
ch = input("Enter a character: ")

# (a) Check type of character
if ch.isalpha():
    print("It is a LETTER")

    # (b) Check uppercase/lowercase
    if ch.isupper():
        print("Uppercase Letter")
    else:
        print("Lowercase Letter")

elif ch.isdigit():
    print("It is a NUMERIC DIGIT")

    # (c) Print digit name
    names = ["ZERO","ONE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT","NINE"]
    print("In words:", names[int(ch)])

else:
    print("It is a SPECIAL CHARACTER")
```



```
Enter a character: e
It is a LETTER
Lowercase Letter
```

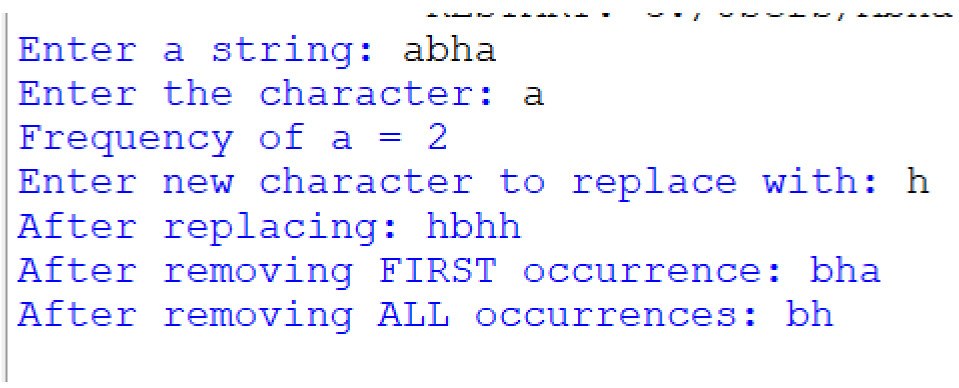
ch = input("Enter a character: ")

```
# (a) Check type of character
if ch.isalpha():
    print("It is a LETTER")

    # (b) Check uppercase/lowercase
    if ch.isupper():
        print("Uppercase Letter")
    else:
        print("Lowercase Letter")

elif ch.isdigit():
    print("It is a NUMERIC DIGIT")

    # (c) Print digit name
    names =
["ZERO","ONE","TWO","THREE","FOUR","FIVE","SIX","SEVEN","EIGHT","NI
NE"]
    print("In words:", names[int(ch)])

else:
    print("It is a SPECIAL CHARACTER")
```

5. Write a program to perform the following operations on a string
 a. Find the frequency of a character in a string.

b. Replace a character by another character in a string.
c. Remove the first occurrence of a character from a string.
d. Remove all occurrences of a character from a string.

```python
s = input("Enter a string: ")
ch = input("Enter the character: ")

# (a) Frequency of character
print("Frequency of", ch, "=", s.count(ch))

# (b) Replace a character by another
new_ch = input("Enter new character to replace with: ")
print("After replacing:", s.replace(ch, new_ch))

# (c) Remove first occurrence
print("After removing FIRST occurrence:", s.replace(ch, "", 1))

# (d) Remove all occurrences
print("After removing ALL occurrences:", s.replace(ch, ""))
```



```
Enter a string: abha
Enter the character: a
Frequency of a = 2
Enter new character to replace with: h
After replacing: hbhh
After removing FIRST occurrence: bha
After removing ALL occurrences: bh
```

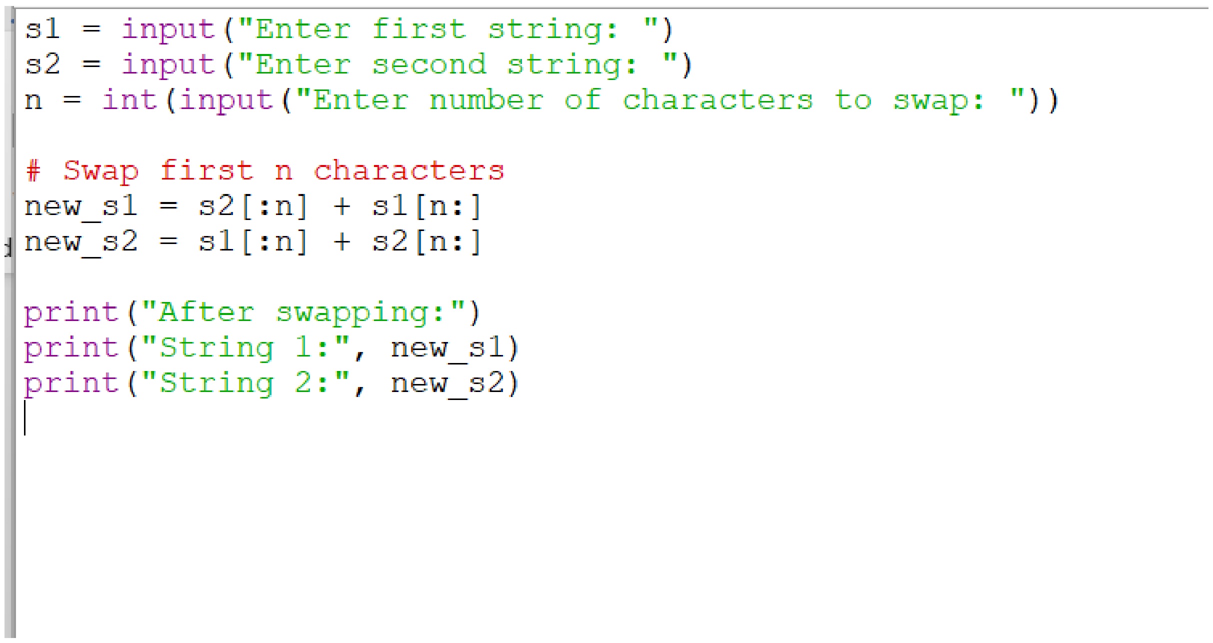
s = input("Enter a string: ")
ch = input("Enter the character: ")

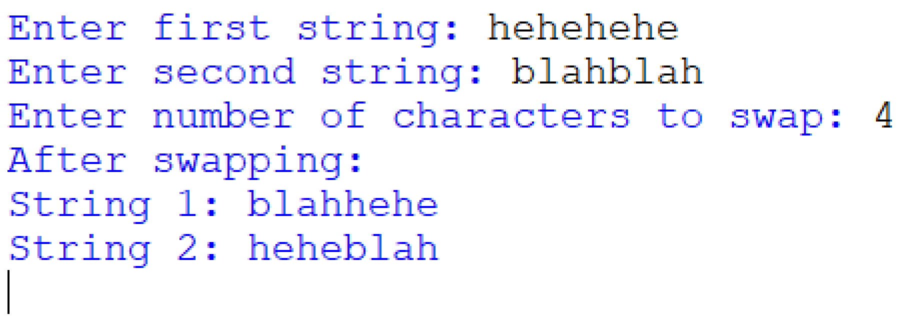
# (a) Frequency of character
print("Frequency of", ch, "=", s.count(ch))

# (b) Replace a character by another
new_ch = input("Enter new character to replace with: ")
print("After replacing:", s.replace(ch, new_ch))

# (c) Remove first occurrence
print("After removing FIRST occurrence:", s.replace(ch, "", 1))

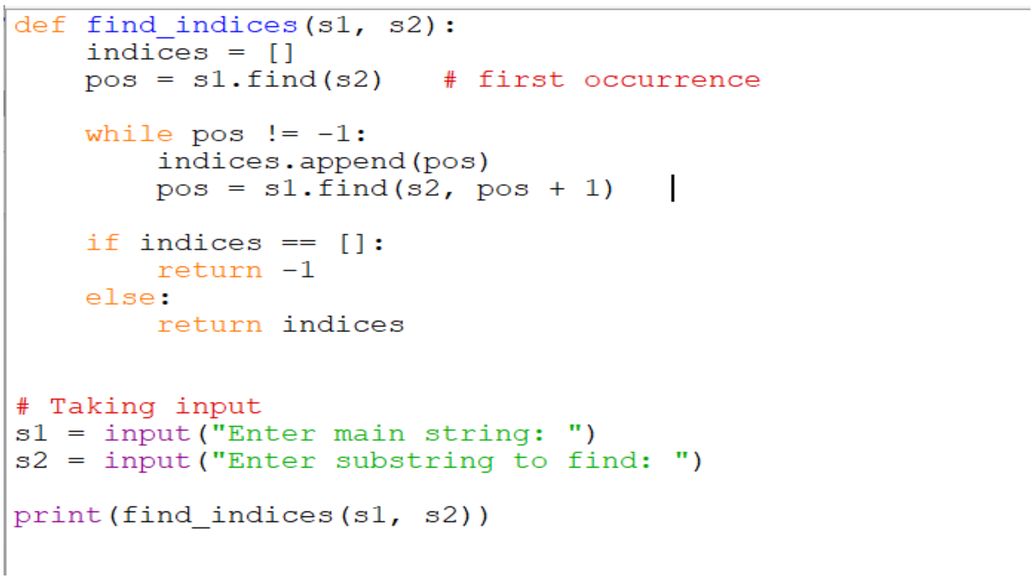
# (d) Remove all occurrences
print("After removing ALL occurrences:", s.replace(ch, ""))
6. Write a program to swap the first n characters of two strings.


```
s1 = input("Enter first string: ")
s2 = input("Enter second string: ")
n = int(input("Enter number of characters to swap: "))

# Swap first n characters
new_s1 = s2[:n] + s1[n:]
new_s2 = s1[:n] + s2[n:]

print("After swapping:")
print("String 1:", new_s1)
print("String 2:", new_s2)
```

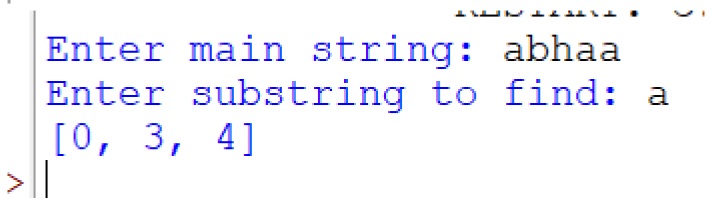

```
Enter first string: hehehehe
Enter second string: blahblah
Enter number of characters to swap: 4
After swapping:
String 1: blahhehe
String 2: heheblah
```

```
s1 = input("Enter first string: ")
s2 = input("Enter second string: ")
n = int(input("Enter number of characters to swap: "))

# Swap first n characters
new_s1 = s2[:n] + s1[n:]
new_s2 = s1[:n] + s2[n:]

print("After swapping:")
print("String 1:", new_s1)
print("String 2:", new_s2)
```

7. Write a function that accepts two strings and returns the indices of all the occurrences of the second string in the first string as a list. If the second string is not present in the first string then it should return -1.



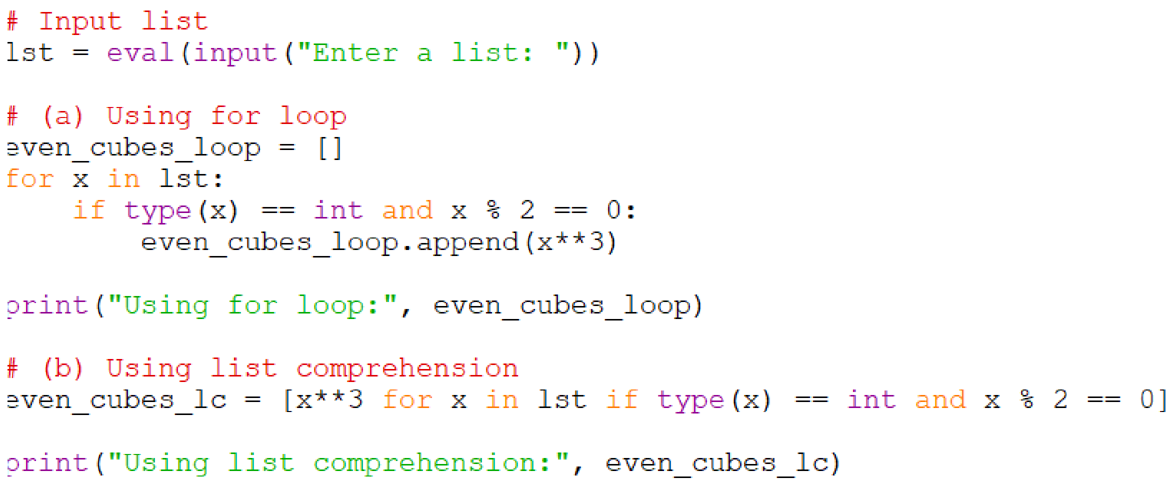
```
def find_indices(s1, s2):
    indices = []
    pos = s1.find(s2)    # first occurrence

    while pos != -1:
        indices.append(pos)
        pos = s1.find(s2, pos + 1)

    if indices == []:
        return -1
    else:
        return indices


# Taking input
s1 = input("Enter main string: ")
s2 = input("Enter substring to find: ")

print(find_indices(s1, s2))
```



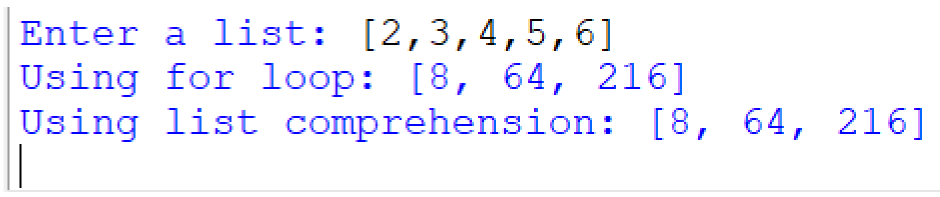
```
Enter main string: abhaa
Enter substring to find: a
[0, 3, 4]
>
```

```
def find_indices(s1, s2):
    indices = []
    pos = s1.find(s2)   # first occurrence

    while pos != -1:
        indices.append(pos)
        pos = s1.find(s2, pos + 1)

    if indices == []:
        return -1
    else:
        return indices

# Taking input
s1 = input("Enter main string: ")
s2 = input("Enter substring to find: ")

print(find_indices(s1, s2))
```

8. Write a program to create a list of the cubes of only the even integers appearing in the input list (may have elements of other types also) using the following:
 a. 'for' loop
 b. list comprehension



```python
# Input list
lst = eval(input("Enter a list: "))

# (a) Using for loop
even_cubes_loop = []
for x in lst:
    if type(x) == int and x % 2 == 0:
        even_cubes_loop.append(x**3)

print("Using for loop:", even_cubes_loop)

# (b) Using list comprehension
even_cubes_lc = [x**3 for x in lst if type(x) == int and x % 2 == 0]

print("Using list comprehension:", even_cubes_lc)
```



```
Enter a list: [2,3,4,5,6]
Using for loop: [8, 64, 216]
Using list comprehension: [8, 64, 216]
```

```
# Input list
lst = eval(input("Enter a list: "))

# (a) Using for loop
even_cubes_loop = []
for x in lst:
   if type(x) == int and x % 2 == 0:
      even_cubes_loop.append(x**3)

print("Using for loop:", even_cubes_loop)

# (b) Using list comprehension
even_cubes_lc = [x**3 for x in lst if type(x) == int and x % 2 == 0]

print("Using list comprehension:", even_cubes_lc)
```

9. Write a program to read a file and
 a. Print the total number of characters, words and lines in the file.

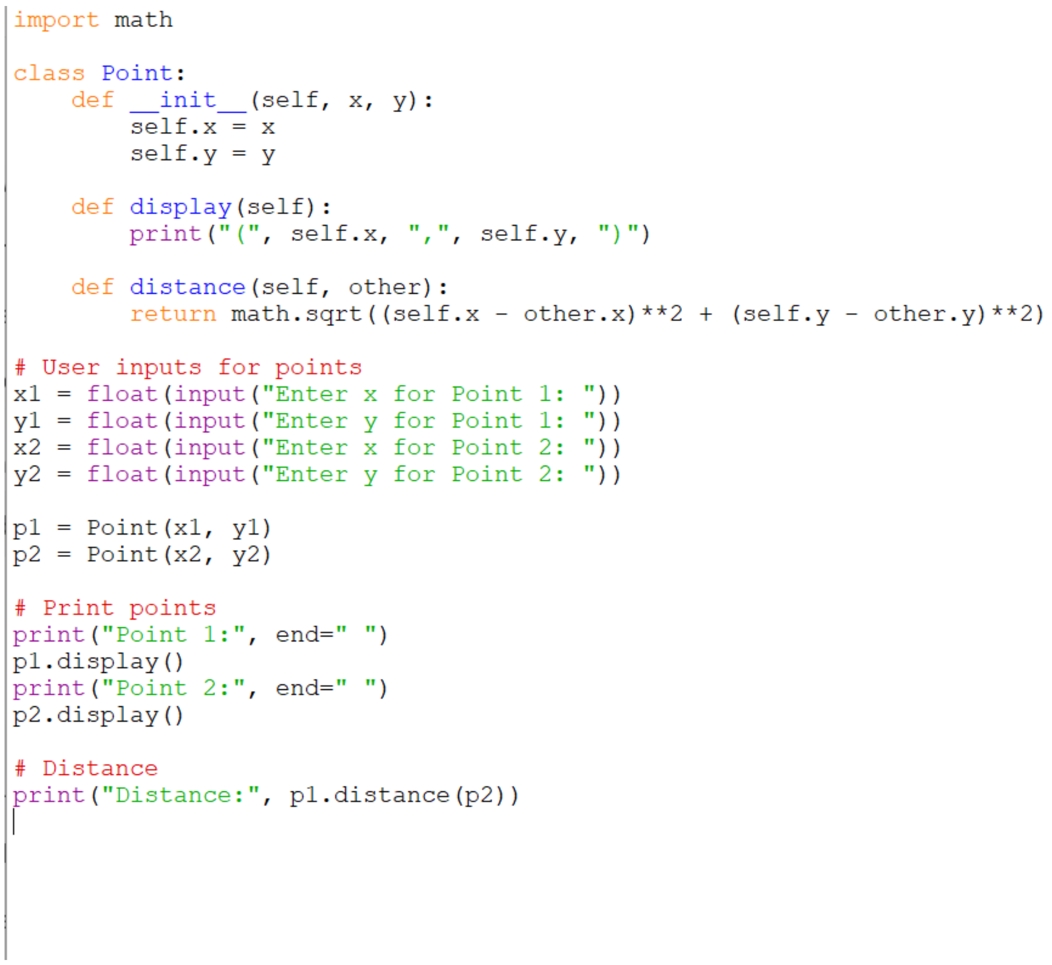
b. Calculate the frequency of each character in the file. Use a variable of dictionary type to maintain the count.
c. Print the words in reverse order.
d. Copy even lines of the file to a file named ‘File1’ and odd lines to another file named ‘File2’.

```
file = open("input.txt", "r")
lines = file.readlines()
file.close()

# (a) Total characters, words, lines
chars = sum(len(line) for line in lines)
words = sum(len(line.split()) for line in lines)
lines_count = len(lines)
print("Characters:", chars)
print("Words:", words)
print("Lines:", lines_count)

# (b) Character frequency
freq = {}
for line in lines:
    for ch in line.strip():
        freq[ch] = freq.get(ch, 0) + 1
print("\nCharacter Frequency:", freq)

# (c) Words in reverse
all_words = []
for line in lines:
    all_words += line.split()
print("\nWords in reverse:", all_words[::-1])

# (d) Copy even and odd lines
f1 = open("File1.txt", "w")  # Even lines
f2 = open("File2.txt", "w")  # Odd lines
for i in range(len(lines)):
    if (i+1) % 2 == 0:
        f1.write(lines[i])
    else:
        f2.write(lines[i])
f1.close()
f2.close()
print("\nEven lines -> File1.txt, Odd lines -> File2.txt")
```

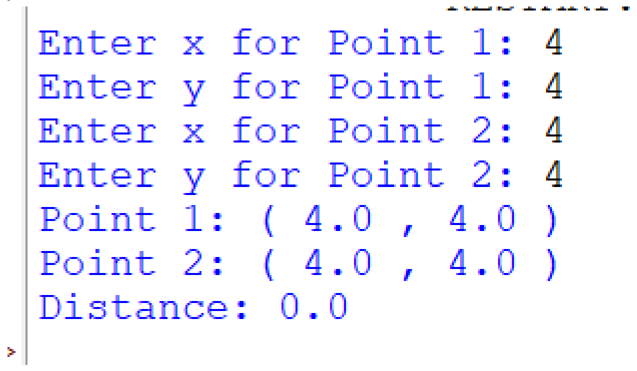
```
# Make sure "input.txt" exists in the same folder
file = open("input.txt", "r")
lines = file.readlines()
```

```
file.close()

# (a) Total characters, words, lines
chars = sum(len(line) for line in lines)
words = sum(len(line.split()) for line in lines)
lines_count = len(lines)
print("Characters:", chars)
print("Words:", words)
print("Lines:", lines_count)

# (b) Character frequency
freq = {}
for line in lines:
    for ch in line.strip():
        freq[ch] = freq.get(ch, 0) + 1
print("\nCharacter Frequency:", freq)

# (c) Words in reverse
all_words = []
for line in lines:
    all_words += line.split()
print("\nWords in reverse:", all_words[::-1])

# (d) Copy even and odd lines
f1 = open("File1.txt", "w")  # Even lines
f2 = open("File2.txt", "w")  # Odd lines
for i in range(len(lines)):
    if (i+1) % 2 == 0:
        f1.write(lines[i])
    else:
        f2.write(lines[i])
f1.close()
f2.close()
print("\nEven lines -> File1.txt, Odd lines -> File2.txt")
```

10. Write a program to define a class Point with coordinates x and y as attributes. Create relevant methods and print the objects. Also define a method distance to calculate the distance between any two point objects.



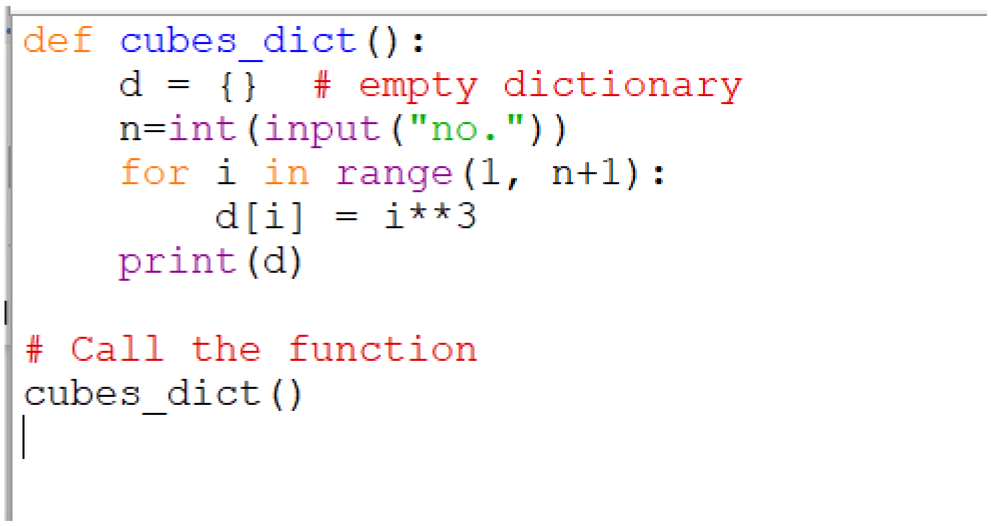
```python
import math

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def display(self):
        print("(", self.x, ",", self.y, ")")

    def distance(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)

# User inputs for points
x1 = float(input("Enter x for Point 1: "))
y1 = float(input("Enter y for Point 1: "))
x2 = float(input("Enter x for Point 2: "))
y2 = float(input("Enter y for Point 2: "))

p1 = Point(x1, y1)
p2 = Point(x2, y2)

# Print points
print("Point 1:", end=" ")
p1.display()
print("Point 2:", end=" ")
p2.display()

# Distance
print("Distance:", p1.distance(p2))
```



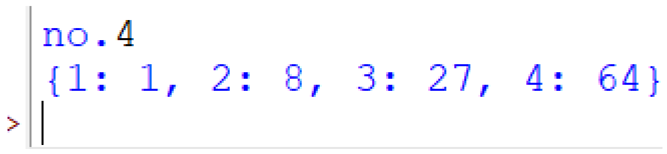
```
Enter x for Point 1: 4
Enter y for Point 1: 4
Enter x for Point 2: 4
Enter y for Point 2: 4
Point 1: ( 4.0 , 4.0 )
Point 2: ( 4.0 , 4.0 )
Distance: 0.0
>
```

```
import math

class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def display(self):
        print("(", self.x, ",", self.y, ")")

    def distance(self, other):
        return math.sqrt((self.x - other.x)**2 + (self.y - other.y)**2)

# User inputs for points
x1 = float(input("Enter x for Point 1: "))
y1 = float(input("Enter y for Point 1: "))
x2 = float(input("Enter x for Point 2: "))
y2 = float(input("Enter y for Point 2: "))

p1 = Point(x1, y1)
p2 = Point(x2, y2)

# Print points
print("Point 1:", end=" ")
p1.display()
print("Point 2:", end=" ")
p2.display()

# Distance
print("Distance:", p1.distance(p2))
```

11. Write a function that prints a dictionary where the keys are numbers between 1 and 5 and the values are cubes of the keys.



```
def cubes_dict():
    d = {}  # empty dictionary
    n=int(input("no."))
    for i in range(1, n+1):
        d[i] = i**3
    print(d)

# Call the function
cubes_dict()
```



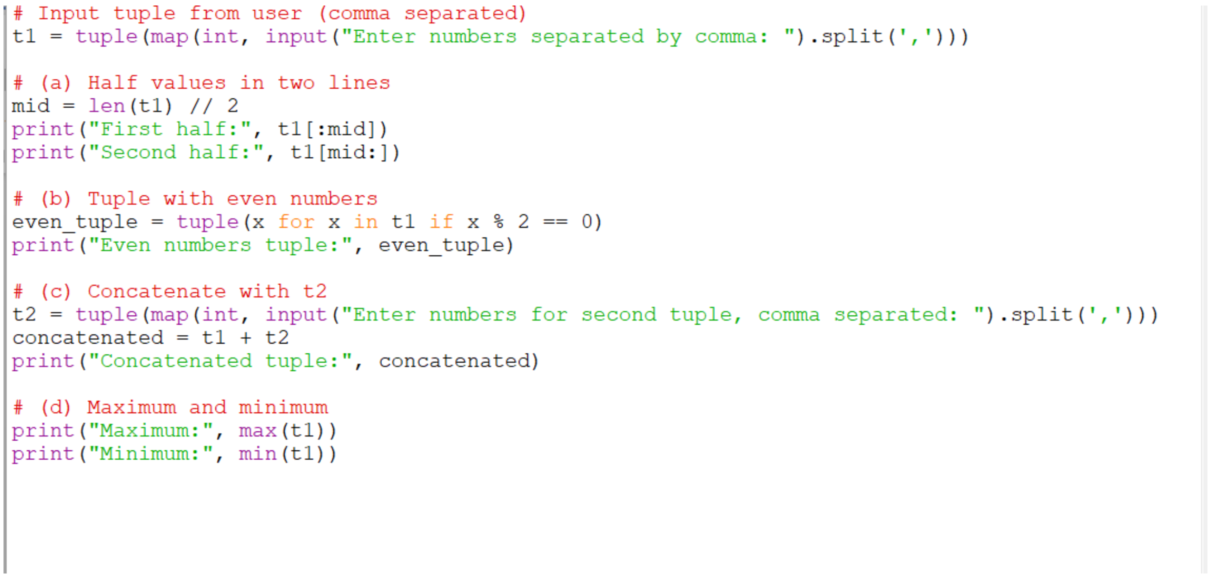
```
no.4
{1: 1, 2: 8, 3: 27, 4: 64}
>
```

```
def cubes_dict():
    d = {}  # empty dictionary
    n=int(input("no."))
    for i in range(1, n+):
        d[i] = i**3
    print(d)

# Call the function
cubes_dict()
```

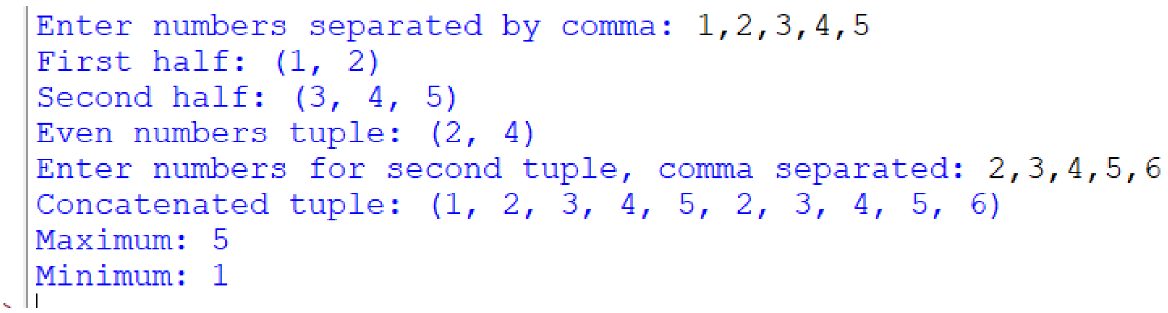
12. Consider a tuple t1=(1, 2, 5, 7, 9, 2, 4, 6, 8, 10). Write a program to perform following operations:
 a. Print half the values of the tuple in one line and the other half in the next line. b. Print another tuple whose values are even numbers in the given tuple.

c. Concatenate a tuple t2=(11,13,15) with t1.

d. Return maximum and minimum value from this tuple



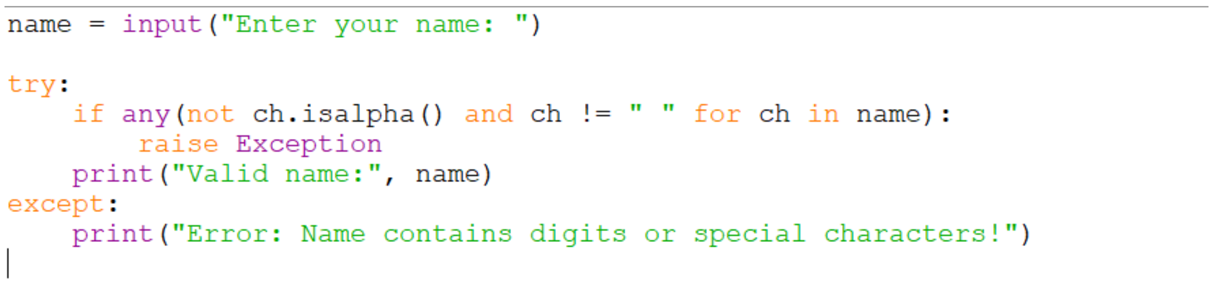
```
# Input tuple from user (comma separated)
t1 = tuple(map(int, input("Enter numbers separated by comma: ").split(',')))

# (a) Half values in two lines
mid = len(t1) // 2
print("First half:", t1[:mid])
print("Second half:", t1[mid:])

# (b) Tuple with even numbers
even_tuple = tuple(x for x in t1 if x % 2 == 0)
print("Even numbers tuple:", even_tuple)

# (c) Concatenate with t2
t2 = tuple(map(int, input("Enter numbers for second tuple, comma separated: ").split(',')))
concatenated = t1 + t2
print("Concatenated tuple:", concatenated)

# (d) Maximum and minimum
print("Maximum:", max(t1))
print("Minimum:", min(t1))
```



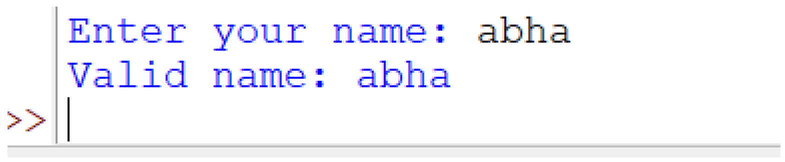
```
Enter numbers separated by comma: 1,2,3,4,5
First half: (1, 2)
Second half: (3, 4, 5)
Even numbers tuple: (2, 4)
Enter numbers for second tuple, comma separated: 2,3,4,5,6
Concatenated tuple: (1, 2, 3, 4, 5, 2, 3, 4, 5, 6)
Maximum: 5
Minimum: 1
```

```
# Input tuple from user (comma separated)
t1 = tuple(map(int, input("Enter numbers separated by comma: ").split(',')))

# (a) Half values in two lines
mid = len(t1) // 2
print("First half:", t1[:mid])
print("Second half:", t1[mid:])

# (b) Tuple with even numbers
even_tuple = tuple(x for x in t1 if x % 2 == 0)
print("Even numbers tuple:", even_tuple)

# (c) Concatenate with t2
t2 = tuple(map(int, input("Enter numbers for second tuple, comma separated:
").split(',')))
concatenated = t1 + t2
print("Concatenated tuple:", concatenated)

# (d) Maximum and minimum
print("Maximum:", max(t1))
print("Minimum:", min(t1))
```

13. Write a program to accept a name from a user. Raise and handle appropriate exception(s) if the text entered by the user contains digits and/or special characters.



```python
name = input("Enter your name: ")

try:
    if any(not ch.isalpha() and ch != " " for ch in name):
        raise Exception
    print("Valid name:", name)
except:
    print("Error: Name contains digits or special characters!")
```



```
Enter your name: abha
Valid name: abha
>>
```

```python
name = input("Enter your name: ")

try:
    if any(not ch.isalpha() and ch != " " for ch in name):
        raise Exception
    print("Valid name:", name)
except:
    print("Error: Name contains digits or special characters!")
```