```
'''## Q1 ## Write a Python function to check whether a number is
perfect or not. According to Wikipedia : In number theory,
            a perfect number is a positive integer that is equal to
the sum of its proper positive divisors, that is, the sum of its
            positive divisors excluding the number itself (also known
as its aliquot sum). Equivalently, a perfect number is a
            number that is half the sum of all of its positive
divisors (including itself).'''

def perfect_number(num):
    sum = 0
    for x in range(1, num):
        if num % x == 0:
            sum += x
    return sum == num
num = int(input("Enter a number to check if it is a perfect number:
"))
print(perfect_number(num))
```

```
Enter a number to check if it is a perfect number: 6
True
```

```
'''## Q2 ## Write a Python function that checks whether a passed
string is palindrome or not. Note: A palindrome is a word,
            phrase, or sequence that reads the same backward as
forward, e.g., madam or nurses run.'''

def isPalindrome(string):
    rev_string = string[::-1]

    if rev_string == string :
        print("The entered string is a Palindrome.")

    else:
        print("The entered string is not a Palindrome.")

string = str(input("Enter the string to check if it is a palindrome:
"))
isPalindrome(string)
```

```
Enter the string to check if it is a palindrome: racecar
The entered string is a Palindrome.
```

```
'''## Q3 ## Write a Python function that prints out the first n rows
of Pascal's triangle.
            Note: Each number is the two numbers above it added
together'''

from math import factorial
def pascal_triangle(n):
```

```
    for i in range(n):
        for j in range(n-1-i):
            print(" ", end = "")
        for k in range (i+1):
            print(factorial(i) // (factorial(i-k)*factorial(k)) ,
end=" ")     # nCr = n! / ((n-r)! * r!)
        print()

n = int(input("Enter the number of rows you want to be printed: "))
pascal_triangle(n)
```

```
Enter the number of rows you want to be printed: 5
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

```
'''## Q4 ## Write a Python function to check whether a string is a
pangram or not.
            Note: Pangrams are words or sentences containing every
letter of the alphabet at least once.
            example: "The quick brown fox jumps over the lazy dog"'''

def pangram(string):
    alphabet = "abcdefghijklmnopqrstuvwxyz"

    for buck in alphabet:
        if buck not in string:
            return False
        return True

user_string=str(input("Enter a sentence: "))

if pangram(user_string) == False:
    print("The entered sentence is not a pangram.")

else:
    print("The entered sentence is a pangram.")
```

```
Enter a sentence: qwertyuiop asdfghjkl zxcvbnm
The entered sentence is a pangram.
```

```
'''## Q5 ## Write a Python function that accepts a hyphen-separated
sequence of words as input
            and prints the words in a hyphen-separated sequence after
sorting them alphabetically.
            Sample Items : green-red-yellow-black-white
            Expected Result : black-green-red-white-yellow'''

input_string=str(input("Enter a hyphen separated sentence: "))
```

```
li = list(input_string.split("-"))
li.sort()

print("-".join(li))
```

```
Enter a hyphen separated sentence: green-red-yellow-black-white
black-green-red-white-yellow
```

```
'''## Q6 ## Write a Python function student_data () which will print
the id of a student (student_id).
        If the user passes an argument student_name or student_class
the function will print the
        student name and class.'''

def student_data(student_id, **kwargs):
    print(f"\nStudent ID: {student_id}")

    if 'student_name' in kwargs:
        print(f"Student Name: {kwargs['student_name']}")

    if 'student_name' and 'student_class' in kwargs:
            print(f"\nStudent Name: {kwargs['student_name']}")
            print(f"Student Class: {kwargs['student_class']}")


student_data(student_id='21107042', student_name='Shubham')

student_data(student_id='211070121', student_name='Monte Carlo',
student_class ='V')
```

```
Student ID: 21107042
Student Name: Shubham

Student ID: 211070121
Student Name: Monte Carlo

Student Name: Monte Carlo
Student Class: V
```

```
'''## Q7 ## Write a Python program to create two empty classes,
Student and Marks. Now create
            some instances and check whether they are instances of the
said classes or not. Also,
            check whether the said classes are subclasses of the
built-in object class or not.'''

class Student:
    pass
```

```
class Marks:
    pass
student1 = Student()
marks1 = Marks()
print("\n## Check whether the said are instances of the built-in
object class or not.")
print(isinstance(student1, Student))
print(isinstance(marks1, Student))
print(isinstance(marks1, Marks))
print(isinstance(student1, Marks))
print("\n## Check whether the said classes are subclasses of the
built-in object class or not.")
print(issubclass(Student, object))
print(issubclass(Marks, object))
```

```
## Check whether the said are instances of the built-in object class
or not.
True
False
True
False

## Check whether the said classes are subclasses of the built-in
object class or not.
True
True
```

```
'''## Q8 ## Write a Python class to find the three elements that sum
to zero from a set of n real numbers.
            Input array: [-25, -10, -7, -3, 2, 4, 8, 10]
            Output: [[-10, 2, 8], [-7, -3, 10]]'''

class py_solution:
    def threeSum(self, nums):
        nums, result, i = sorted(nums), [], 0

        while i < len(nums) - 2:
            j, k = i + 1, len(nums) - 1

            while j < k:

                if nums[i] + nums[j] + nums[k] < 0:
                    j += 1
                elif nums[i] + nums[j] + nums[k] > 0:
                    k -= 1
                else:
                    result.append([nums[i], nums[j], nums[k]])
                    j, k = j + 1, k - 1
```

```
                    while j < k and nums[j] == nums[j - 1]:
                        j += 1

                    while j < k and nums[k] == nums[k + 1]:
                        k -= 1
            i += 1
            while i < len(nums) - 2 and nums[i] == nums[i - 1]:
                i += 1

        return result

print(py_solution().threeSum([-25, -10, -7, -3, 2, 4, 8, 10]))
```

```
[[-10, 2, 8], [-7, -3, 10]]
```

```
'''## Q9 ## Write a Python class to find validity of a string of
parentheses, '(', ')', '{', '}', '[' and ']'.
            These brackets must be close in the correct order, for
example "()" and "()[]{}" are valid
            but "[)", "({[)]" and "{{{" are invalid.'''
class parantheses:
    def find(str):
        a= ['()', '{}', '[]']
        while any(i in str for i in a):
            for j in a:
                str = str.replace(j, '')
        return not str

string = input("Enter the sequence of parantheses: ")
if parantheses.find(s):
    print(string,"-","is a valid string of parentheses.")
else:
    print(string,"-","is an invalid string of parentheses.")
```

```
Enter the sequence of parantheses: []{}()
[]{}() - is a valid string of parentheses.
```