Task (1): What the difference between library and framework?

The technical difference between a framework and library lies in a term called inversion of control.

When you use a library, you are in charge of the flow of the application. You are choosing when and where to call the library. When you use a framework, the framework is in charge of the flow. It provides some places for you to plug in your code, but it calls the code you plugged in as needed.

Frameworks and libraries are both code written by someone else that helps you perform some common tasks in a less verbose way.

A framework inverts the control of the program. It tells the developer what they need. A library doesn't. The programmer calls the library where and when *they* need it.

a library is a collection of pre-written code that provides specific functionality and can be used to extend your code, while a framework is a more comprehensive software environment that provides a structure and tools for building applications in a specific domain.

Task (2): What is the difference between function and method?

Function: A function is a self-contained block of code that performs a specific task or calculation. It is a reusable piece of code that can be called from other parts of a program. Functions can take input parameters, perform operations, and return a result. In Python, functions are defined using the def keyword.

Example:

```
def square(number):
    return number ** 2
```

In this case, square is a function that takes a parameter called number and returns the square of that number.

Method: A method, on the other hand, is a function that is associated with an object or a class. It is similar to a function, but it is defined within the context of a specific object or class and can access and operate on the object's internal data.

Methods are typically used in object-oriented programming (OOP), where objects are instances of classes that encapsulate data and behaviour. Methods are defined within the class definition and are accessed through objects or class instances.

Example:

```
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_area(self):
        return 3.14159 * self.radius ** 2
```

In this case, calculate_area is a method of the Circle class. It is defined within the class and operates on the radius attribute of the class instance.

Task (3): How to make the user input only String without numbers?

```
while True:
    text = input("Enter a string: ")
    if text.replace(" ", "").isalpha():
        break
    else:
        print("Invalid input. Please enter a string with only letters and spaces."
)

print("You entered:", text)
```

OR we can use something which is like regex in C++:

```
import re

pattern = r'\b[A-Za-z]+\b'  # Example regex pattern to match words

text = "Hello, World! This is a sample text."

matches = re.findall(pattern, text)
print(matches)
```

In this example, the re.findall() function is used to find all matches of the regex pattern in the given text. The regular expression pattern r'\b[A-Za-z]+\b' searches for one or more alphabetic words.

The re module in Python provides several other functions and methods for regex operations, such as re.search(), re.match(), re.sub(), and more. These functions allow you to perform more advanced regex operations like searching for specific patterns, replacing text, or extracting portions of strings.

Task (4): How to access an element in set?

We can use that way:

```
my_set = {1, 2, 3, 4, 5}
my_list = list(my_set)

print(my_list[0])  # Accessing the first element of the list
```

Task (5): How to make do while loop by python?

Python doesn't have a built-in do-while loop like some other programming languages, but you can achieve similar functionality using a while loop with a condition that is initially True. Inside the loop, you can check the desired condition and use the break statement to exit the loop when needed. Here's an example:

```
while True:
    # Code to be executed
    # ...

    # Condition check
    if condition:
        break
```