Q.1. What are keywords in python? Using the keyword library, print all the python keywords.

- Keywords are reserved words in Python that have a special meaning to the interpreter. They cannot be used as variable names or function names. To print all the Python keywords, you can use the `keyword` module.
- The `keyword` module contains a list of all the Python keywords. To print the list of keywords, you can use the `print()` function.

```
import keyword

print(keyword.kwlist)
```

- Output :

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

Q.2. What are the rules to create variables in python?

- To create a variable in Python, you must first assign a value to it. This can be done using the assignment operator (=). For example, to create a variable called "my_variable" and assign it the value "10", you would write:

```
my_variable = 10
```

- Once you have created a variable, you can use it in your code by referencing its name. For example, to print the value of the variable "my_variable", you would write:

```
print(my_variable)
```

- There are a few rules that you must follow when creating variables in Python. First, variable names must start with a letter or underscore (_). Second, variable names cannot contain spaces. Third, variable names cannot be the same as keywords in Python. For a full list of keywords, see the Python documentation.

Q.3. What are the standards and conventions followed for the nomenclature of variables in python to improve code readability and maintainability?

- Use descriptive variable names that clearly indicate what the variable is used for.
- Use lower case letters for variable names, and separate words with underscores (_).
- Avoid using abbreviations or acronyms in variable names.
- Avoid using names that are too long or too short.
- Avoid using names that are similar to other names in your code.

Q.4. What will happen if a keyword is used as a variable name?

- If you try to use a keyword as a variable name, Python will throw an error. For example, if you try to create a variable called "for", Python will give you the following error:

  SyntaxError: invalid syntax

- This is because keywords are reserved words that have special meaning in Python. They cannot be used as variable names, function names, or class names.

Q.5. For what purpose def keyword is used?

- The def keyword is used to define a function in Python. A function is a block of code that can be reused in different parts of your program. To define a function, you use the following syntax:

  ```
  def function_name(parameters):
  ```

- where:
- `function_name` is the name of the function.
- `parameters` are the arguments that the function takes.
- `code block` is the code that the function executes.

- Once you have defined a function, you can call it by using its name and passing in the arguments. For example, if you have a function called `add_two_numbers` that takes two numbers as arguments, you can call it like this

  ```
  add_two_numbers(1, 2)
  ```

- This will call the `add_two_numbers` function and pass in the values 1 and 2 as arguments. The function will then add the two numbers together and return the result

Q.6. What is the operation of this special character '\'?

- The backslash character (\) is used to escape special characters in Python. This means that it can be used to tell the interpreter that a character should be interpreted literally, even if it would normally have a special meaning. For example, the backslash character can be used to escape the newline character (\n), which is used to end a line of code. Here are some examples of how the backslash character can be used:

- To escape the newline character:

  ```
  print("This is a line of code.\nThis is another line of code.")
  ```

- To escape the tab character:

  print("This is a line of code.\tThis is another line of code.")

- To escape the backslash character itself:

  print("This is a line of code.\\This is another line of code.")

- The backslash character can also be used to escape other special characters, such as the double quote character (\") and the single quote character (\'). Here are some examples of how the backslash character can be used to escape special characters: * To escape the double quote character:

  print("This is a line of code.\"This is another line of code.")

- To escape the single quote character:
  print("This is a line of code.'This is another line of code.")

- The backslash character can be a useful tool for writing Python code. It can be used to escape special characters and to make your code more readable.

Q.7. Give an example of the following conditions:

(i) Homogeneous list:

- A homogeneous list is a list that contains elements of the same type. For example, the following list is a homogeneous list of integers:

- H_list = [1, 2, 3, 4, 5]

(ii) Heterogeneous set:

- A heterogeneous set is a set that contains elements of different types. For example, the following set is a heterogeneous set of integers, strings, and floats:

- H_set = {1, '2', 3.0, '4', 5}

(iii) Homogeneous tuple:

- A homogeneous tuple is a tuple that contains elements of the same type. For example, the following tuple is a homogeneous tuple of integers:

- H_tuple = (1, 2, 3, 4, 5)

Q.8. Explain the mutable and immutable data types with proper explanation & examples.

- there are two types of data types: mutable and immutable. Mutable data types can be changed after they are created, while immutable data types cannot.
- Some of the most common mutable data types are lists, dictionaries, and sets. Lists are ordered collections of objects, dictionaries are unordered collections of key-value pairs, and sets are unordered collections of unique objects.
- Some of the most common immutable data types are numbers, strings, and tuples. Numbers are used to represent numeric values, strings are used to represent text values, and tuples are used to represent ordered collections of objects.

- Here is an example of how to create a mutable list:
  my_list = [1, 2, 3, 4, 5]

- This list can be changed by adding, removing, or rearranging elements. For example, the following code will add the number 6 to the end of the list: my_list.append(6)

- Here is an example of how to create an immutable tuple:
  my_tuple = (1, 2, 3, 4, 5)

- This tuple cannot be changed after it is created. If you try to add, remove, or rearrange elements, you will get an error. Here is a table that summarizes the differences between mutable and immutable data types:

Data type	Mutable	Immutable
List	Yes	No
Dictionary	Yes	No
Set	No	Yes
Tuple	No	Yes
String	No	Yes

Q.9. Write a code to create the given structure using only for loop.

*

***

*****

****

****

- for i in range(10):
    if i%2 == 1 :
      for j in range(i):
        print("*",end=" ")
      print()

Q.10. Write a code to create the given structure using while loop.

|||||||||

|||||||

|||||

|||

|

- i = 0
  while i<10:
    if i%2 == 1:
      print("| "* i)
    i += 2