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

Answer:-

In Python, keywords are reserved words that have special meanings and purposes within the language. These words cannot be used as identifiers (variable names, function names, etc.) because they are already predefined with specific functionality. You can use the keyword library in Python to access and print all the Python keywords.

```
import keyword

all_keywords = keyword.kwlist
print(all_keywords)
```

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?
Answer:-
In Python, variables are used to store data and are created following certain rules and conventions. Here are the rules to create variables in Python:

- Variable names must start with a letter (a-z, A-Z) or an underscore (_) character.
- The first character of the variable name cannot be a digit (0-9).
- Variable names can only contain alphanumeric characters (a-z, A-Z, 0-9) and underscores (_). Special characters and spaces are not allowed.
- Variable names are case-sensitive, meaning "myVar" and "myvar" are considered different variables.
- Python keywords (reserved words) cannot be used as variable names.

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

Answer:-

In Python, there are several standards and conventions for naming variables that aim to improve code readability and maintainability. Following these conventions makes it easier for other developers to understand your code and for you to understand code written by others. Here are some widely accepted naming conventions for variables in Python:

Snake_case: Use lowercase letters for variable names, and separate words with underscores. For example: user_name, total_count, first_name.

Descriptive Names: Choose meaningful and descriptive names for variables. This helps to understand the purpose and content of the variable without the need for extensive comments. For example: num_students instead of n, user_list instead of ul.

Avoid One-letter Variables (except for counters): One-letter variable names like i, j, and k are commonly used as loop counters, but try to avoid using them for other purposes as they lack context and may not be informative.

Avoid Reserved Keywords: Do not use Python's reserved keywords as variable names. For example, if, else, while, for, def, class, import, etc., are reserved keywords.

Q.4. What will happen if a keyword is used as a variable name?
Answer:-
If you use a keyword (reserved word) as a variable name in Python, you will encounter a syntax error.

Q.5. For what purpose def keyword is used?
Answer:-
The def keyword in Python is used to define a user-defined function. Functions are blocks of code that perform a specific task or set of tasks. By defining functions, you can encapsulate a piece of code, give it a name, and then call that code using the function name whenever needed.

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

The special character \ (backslash) in Python is known as the escape character. It has special meaning and is used to escape certain characters, creating escape sequences in strings and other contexts. When you use the backslash before a character, it changes the interpretation of that character. Here are some common escape sequences:

\': Single Quote - Use \' to represent a single quote within a single-quoted string.
\": Double Quote - Use \" to represent a double quote within a double-quoted string.
\\: Backslash - Use \\ to represent a single backslash within a string.
\n: Newline - Represents a newline character, used for line breaks.
\t: Tab - Represents a tab character.
\r: Carriage Return - Represents a carriage return, used in some systems along with \n for line breaks.
\b: Backspace - Represents a backspace character.
\uXXXX: Unicode Escape - Represents a Unicode character with the hexadecimal code XXXX.
\UXXXXXXXX: Unicode Escape - Represents a Unicode character with the hexadecimal code XXXXXXXX.
\xXX: Hexadecimal Escape - Represents a character with the hexadecimal code XX.

Q.7. Give an example of the following conditions:
(i) Homogeneous list
(ii) Heterogeneous set
(iii) Homogeneous tuple
Answer:-

here are examples for each condition:

(i) Homogeneous List:
A homogeneous list is a list that contains elements of the same data type. In Python, lists can store elements of different data types, but in a homogeneous list, all elements are of the same type.

Example of a homogeneous list containing integers:
homogeneous_list_int = [1, 2, 3, 4, 5]

(ii) Heterogeneous Set:
A heterogeneous set is a set that contains elements of different data types. In Python, sets can store elements of different data types, and duplicate elements are automatically removed.

Example of a heterogeneous set containing various data types:
heterogeneous_set = {1, "Hello", 3.14, (1, 2, 3)}

(iii) Homogeneous Tuple:
A homogeneous tuple is a tuple that contains elements of the same data type. Tuples are similar to lists but are immutable, meaning their elements cannot be changed once created.

Example of a homogeneous tuple containing strings:
homogeneous_tuple_str = ("apple", "banana", "orange")

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

In Python, data types can be classified as either mutable or immutable based on whether their values can be changed after they are created.

Mutable Data Types:
Mutable data types are objects whose values can be modified after creation. When you modify a mutable object, you are changing its contents or elements while keeping the same memory location. This means that any variable or reference pointing to the object will reflect the changes.

Examples of mutable data types in Python are:

1. Lists
2. Dictionaries
3. Sets

Immutable Data Types:
Immutable data types, on the other hand, are objects whose values cannot be changed after creation. When you modify an immutable object, you are actually creating a new object with the modified value. The original object remains unchanged, and any variable or reference pointing to it will still have the original value.

Examples of immutable data types in Python are:

1. Integers
2. Floats
3. Strings
4. Tuples

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

```
    *
   ***
  *****
 *******
*********
```

Answer:-

```
n = 5

for i in range(1, n + 1):
    for j in range(n - i):
        print(" ", end="")
    for k in range(2 * i - 1):
        print("*", end="")
    print()
```

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

```
|||||||||
 |||||||
  |||||
   |||
    |
```

Answer:-

```
n=5
i = 1
while i<=n:
    j=1
    while j<i:
        print(" ",end="")
        j+=1
    k= 1
    while k<= 2*(n-i)+1:
        print("|", end="")
        k+=1
    print()
    i+=1
```