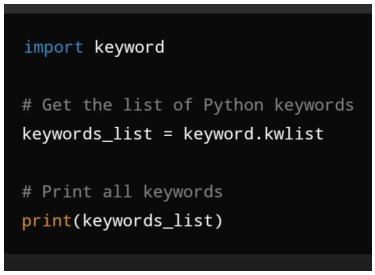
**Answer 1:-**

Keywords in Python
Keywords in Python are reserved words that have predefined meanings and cannot be used as variable names, function names, or any other identifiers. These keywords define the syntax and structure of the Python language.

Printing all Python Keywords
Python provides the keyword module to access the list of keywords. You can print all the keywords.



```
import keyword

# Get the list of Python keywords
keywords_list = keyword.kwlist

# Print all keywords
print(keywords_list)
```

**Answer 2:-**

**Rules for Creating Variables in Python:-**

When defining variables in Python, you must follow these rules:

- Variable names must start with a letter or an undersc(_)
- Variable names can only contain letters (a-z, A-Z), digits (0-9), and underscores (_).
- Variable names are case-sensitive
- Python keywords cannot be used as variable names
- Variable names should be meaningful and readable

**Answer 3:-**

**Conventions for Readability and Maintainability**

1. Use Descriptive and Meaningful Names
2. Use Snake Case (lower_case_with_underscores) for Variable Names
3. Constants Should Be in All Uppercase (UPPER_CASE_WITH_UNDERSCORES)
Constants are values that should not change throughout the program.
4. Avoid Single-Letter Variable Names (Except for Loop Counters)
5. Use Leading Underscore (_variable) for Internal or Private Variables
These variables indicate that they are for internal use and should not be accessed directly.
6. Use Double Leading Underscore (__variable) to Avoid Name Conflicts in Classes

A variable with __ at the beginning is name-mangled in classes to prevent accidental modification.

**Answer 4:-**

If you try to use a Python keyword as a variable name, Python will raise a SyntaxError because keywords have special meanings and are reserved for specific functionalities.

**How to Avoid This Issue?**

- Use a Different Variable Name
- Check for Python Keywords Before Naming a Variable

**Answer 5:-**

The def keyword in Python is used to define a function. Functions are reusable blocks of code that perform a specific task when called.

**Syntax:**

```
def function_name(parameters):
    """Optional docstring"""
    # Function body
    return value  # Optional
```

**Key Uses of def Keyword:-**

- Encapsulates reusable code
- Improves modularity and readability
- Supports parameters and return values
- Can include optional docstrings for documentation

**Answer 6:-**

The / operator in Python is the division operator. It is used to divide two numbers and returns a floating-point (decimal) result, even if both numbers are integers.

Usage of / (Division Operator)

```
result = 10 / 2
print(result)  # Output: 5.0
```

**Answer 7:-**

**Homogeneous list:-**

A homogeneous list in Python is a list where all elements are of the same data type. This ensures consistency and can make processing easier, especially in cases where operations require uniform data types.

**1. List of Integers**

numbers = [1, 2, 3, 4, 5]

**2. List of Strings**

fruits = ["apple", "banana", "cherry"]

**3. List of floats**

**prices = [10.5, 20.75, 30.0]**

**4. List of Boolean value**

flags = [True, False, True, False]

**Heterogeneous set:-**

A heterogeneous set is a collection of elements that are different in nature or type. In other words, the elements in the set do not share a common characteristic or belong to the same category.

**Example:-**

General Example: A set containing different types of objects, like S={apple,7,"hello",3.5}, which includes a fruit, a number, a string, and a decimal.

Heterogeneous sets are common in programming and real-life classifications, where mixed data types or different categories of items need to be grouped together.

**Homogeneous tuple:-**

A homogeneous tuple is a tuple in which all elements are of the same type or category. This means that every element in the tuple shares a common characteristic, such as being all integers, all strings, or all floats.

**Examples:**

**All integers**: (1,2,3,4,5)
**All strings**: ("apple","banana","cherry")
**All floats**: (10.5, 20.75, 50.90)

Homogeneous tuples are often used in programming when consistency in data type is required for operations or processing.

**Answer 8:-**

**1. Mutable Data Types**

- Mutable data types can be modified after creation.
- Changes can be made to elements without creating a new object in memory.

**Examples of mutable data types:-**

1. **List:**

```
my_list=[1, 2, 5, 3]
my_list[0]=10 # changing the first element
Print(my_list) # output:  [10, 2, 3]
```

2. **Dictionary**

```
my_dict={‘name’: ‘kushagra’ , ‘age’: 19}
my_dict[“age”]=20 # updating the age
print(my_dict) # output:
                    {‘name’: ‘laksh’ , ‘age’: 20}
```

**2. Immutable Data Types:-**

- Immutable data types cannot be changed after creation.
- Any modification results in a new object being created in memory.

1. **List**

```
my_str = "hello"
my_str[0] = 'H’

 # This will raise an error because strings are immutable
```

To modify a string, we must create a new one:

```
new_str = "H" + my_str[1:]
```

print(new_str)  # Output: "Hello”

2. **Tuple:**

my_tuple = (1, 2, 3)
my_tuple[0] = 10  # This will raise an error

**Answer 9:-**

```
k=5
for i in range(5):
    print(" "*(k-i) ,end="")
    for j in range(i+1):
        print("*" ,end="")
    print()
```

**Output:-**

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

=== Code Execution Successful ===
```

**Answer 10:-**

```
k=9
for i in range(9):
    print(" "*(k-i) ,end="")
    for j in range(i+1):
        print("|" ,end="")
    print()
```