20 May

Python Basic - 1

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

Ans: Keywords in Python are reserved words that have predefined meanings and are used to perform specific tasks or operations in the Python language. These keywords cannot be used as variable names or identifiers because they have a special significance in the Python syntax.

To print all the Python keywords, you can make use of the keyword library in Python. Here's an example code snippet that demonstrates how to print all the Python keywords:

```
python
import keyword

# Get all the Python keywords
all_keywords = keyword.kwlist

# Print each keyword
for keyword in all_keywords:
    print(keyword)
```

When you run this code, it will display all the Python keywords in the console output. The current list of Python keywords (as of Python 3.9) includes:

```
plaintext
Copy code
False
None
True
and
as
assert
async
await
    break
```

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

Ans: In Python, variables are used to store and manipulate data. Here are the rules for creating variables in Python:

Variable Names:

Variable names can contain letters (a-z, A-Z), digits (0-9), and underscores (_).

They must start with a letter or an underscore. However, it's recommended to start with a lowercase letter.

Variable names are case-sensitive, meaning "myVar" and "myvar" are treated as different variables.

Python keywords cannot be used as variable names.

Naming Conventions:

It's a common convention to use lowercase letters and underscores to separate words in variable names (e.g., my_variable).

Avoid using special characters like !, @, #, $, %, etc. in variable names.

Assignment:

Variables are created when you assign a value to them using the assignment operator (=).

For example: my_variable = 10
Data Types:

Python is a dynamically typed language, so you don't need to explicitly specify the data type of a variable.
The data type of a variable is determined automatically based on the value assigned to it.
You can reassign a variable to a different data type later in your code.
Variable Scope:

Variables have a scope, which determines where they can be accessed and used.
Variables defined inside a function have local scope and can only be accessed within that function.
Variables defined outside any function have global scope and can be accessed throughout the program.
If a variable is defined with the same name inside a function, it creates a new local variable that shadows the global variable.
Here are some examples of valid variable names in Python:

python
Copy code
age = 25
name = "John"
my_var = 10
total_students = 100
It's important to follow these rules and conventions to create meaningful and readable variable names in your Python programs.

Q.3. What are the standards and conventions followed for the nomenclature of variables in python to improve code readability and maintainability?
Ans: In Python, there are certain standards and conventions followed for variable naming to improve code readability and maintainability. The most commonly used standards are:

Snake Case:

Use lowercase letters and underscores to separate words in variable names.
Example: my_variable, total_students, user_name
Descriptive Names:

Use descriptive and meaningful names for variables to make your code more readable.
Avoid using single-letter variable names unless they are used for simple loop counters.
Example: num_students instead of n, first_name instead of fn
Avoid Reserved Words:

Do not use Python keywords and built-in function names as variable names.
Example: list, str, for, if, etc.
Constants:

For variables that store constant values, use uppercase letters and underscores to separate words.
Example: MAX_COUNT, PI, DEFAULT_VALUE
Singular vs. Plural:

Use singular nouns for variables that represent a single item, and plural nouns for variables

that represent collections or multiple items.
Example: student, students, car, cars
Class Names:

Use CamelCase for class names (capitalize the first letter of each word without underscores).
Example: ClassName, MyClass, UserData
Private Variables:

By convention, use a single underscore prefix for private variables that should not be accessed or modified directly outside the class.
Example: _private_variable
Constants:

For variables that should not be modified, use all uppercase letters with underscores to separate words.
Example: MAX_SIZE, DEFAULT_COLOR

It's important to follow these naming conventions to make your code more readable and maintainable. Consistency in variable naming across your codebase will help others (including yourself) understand and work with your code more easily

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

Ans: If a keyword is used as a variable name in Python, it will result in a syntax error. Python keywords are reserved words that have predefined meanings and are used to perform specific tasks or operations in the Python language. These keywords cannot be used as variable names or identifiers because they have a special significance in the Python syntax.

When a keyword is used as a variable name, Python will raise a SyntaxError indicating that the keyword cannot be used as an identifier. Here's an example that demonstrates this:

```
python
for = 10  # Using "for" as a variable name

# Output:
# SyntaxError: invalid syntax
```

In the above example, we are trying to use the keyword "for" as a variable name, which is not allowed. Python raises a SyntaxError with the message "invalid syntax" because it detects the improper usage of the keyword.

To resolve this issue, you should choose a different variable name that does not conflict with Python keywords. It is recommended to follow the naming conventions and use descriptive names to make your code more readable and avoid naming conflicts with reserved keywords.

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

Ans: The def keyword in Python is used to define a function. It is followed by the name of the function and a set of parentheses, which may contain any parameters that the function accepts. The function definition is then followed by a colon (:), and the body of the function is indented below it.

Here's an example of a function definition using the def keyword:

python
```
def greet(name):
    print("Hello, " + name + "!")

    greet("Alice")
```

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

Ans: The special character \ is known as the backslash or escape character in Python. It has several uses and operations:

Escape sequences: The backslash is used to introduce escape sequences in strings. Escape sequences allow you to include special characters within strings that are otherwise difficult to represent directly. For example, you can use \' to include a single quote within a string enclosed in single quotes, or use \" to include a double quote within a string enclosed in double quotes.

python
```
print('I\'m learning Python.')  # Output: I'm learning Python.
    print("She said, \"Hello!\"")  # Output: She said, "Hello!"
```

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

(i) Homogeneous list
(ii) Heterogeneous set
(iii) Homogeneous tuple

Ans: (i) Homogeneous list:

A homogeneous list is a list in which all elements have the same data type. Here's an example of a homogeneous list containing integers:

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

In this example, all elements of the numbers list are integers. It is a homogeneous list because all the elements have the same data type.

(ii) Heterogeneous set:

A heterogeneous set is a set in which elements can have different data types. Here's an example of a heterogeneous set:

python
```
my_set = {1, 'apple', 3.14, True}
```

In this example, the my_set set contains elements of different data types, including an integer (1), a string ('apple'), a float (3.14), and a boolean (True). It is a heterogeneous set because the elements have different data types.

(iii) Homogeneous tuple:

A homogeneous tuple is a tuple in which all elements have the same data type. Here's an example of a homogeneous tuple containing strings:

python
```
fruits = ('apple', 'banana', 'cherry', 'date')
```

In this example, all elements of the fruits tuple are strings. It is a homogeneous tuple because all the elements have the same data type.

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

Ans: In Python, data types are classified as either mutable or immutable. The distinction refers to whether an object's value can be changed after it is created.

Immutable Data Types:

Immutable data types cannot be modified once they are created. If you attempt to modify an immutable object, a new object is created instead. Examples of immutable data types in Python include:

int (integer)
float (floating-point number)
bool (boolean)
str (string)
tuple

Here's an example to illustrate the immutability of strings:

python

```
name = "Alice"
print(name)  # Output: Alice

# Attempting to modify the string
name[0] = "B"  # Raises TypeError: 'str' object does not support item assignment
```

In this example, the string "Alice" is stored in the variable name. When we attempt to modify the first character of the string, an error occurs because strings are immutable. Instead, a new string would need to be created to reflect the desired change.

Mutable Data Types:

Mutable data types, on the other hand, can be modified after they are created. Changes to a mutable object directly modify the object itself, without creating a new object. Examples of mutable data types in Python include:

list
dict (dictionary)
set

Here's an example showcasing the mutability of lists:

python

```
numbers = [1, 2, 3, 4]
print(numbers)  # Output: [1, 2, 3, 4]

# Modifying the list
numbers.append(5)
print(numbers)  # Output: [1, 2, 3, 4, 5]

numbers[0] = 10
print(numbers)  # Output: [10, 2, 3, 4, 5]
```

In this example, the list numbers is initially created with the values [1, 2, 3, 4]. We can modify the list by appending a new element or by changing the value of an existing element. The list itself is mutable, allowing in-place modifications.

The distinction between mutable and immutable data types is essential because it impacts how objects are handled and how memory is utilized. Immutable objects provide advantages like immutability guarantees and safer sharing of data across multiple references, while mutable objects offer flexibility for dynamic modifications.

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

Ans: rows = 5 # Number of rows in the structure

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

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

Ans: rows = 5 # Number of rows in the structure

```
i = rows
while i >= 1:
    j = 1
    while j <= i:
        print("|", end="")
        j += 1
    print()
    i -= 1
```