**20 May**

**Python Basic - 1**

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

Answer:

*Keywords in Python are reserved words that have predefined meanings and cannot be used as variable names or identifiers. These keywords are part of the Python language syntax and serve specific purposes in the code structure.*

*To print all the Python keywords, you can use the keyword module and its kwlist attribute. Here's an example:*

*import keyword*

*print(keyword.kwlist)*

*This code imports the keyword module and prints the kwlist attribute, which contains a list of all the Python keywords. When executed, it will display the following 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']*

*These are the 35 keywords in Python as of the current version (Python 3.9). It's important to note that keywords are case-sensitive, meaning False, None, and True are distinct from false, none, and true, which can be used as regular variable names.*

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

Answer:

*In Python, there are certain rules and conventions to follow when creating variables. Here are the key rules for variable creation in Python:*

*Variable Naming:*

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

*Variable names must start with a letter or an underscore. They cannot start with a digit.*

*Variable names are case-sensitive, meaning myVar and myvar are considered different variables.*

*Variable names should be descriptive and follow a naming convention (e.g., lowercase with underscores, also known as snake\_case).*

*Reserved Keywords:*

*Avoid using reserved keywords as variable names. These are special words in Python with predefined meanings and cannot be used as variable names. You can refer to the previous question for a list of Python keywords.*

*Valid Examples:*

*Valid variable names: my\_var, count, \_name, total\_1.*

*Invalid variable names: 123abc (starts with a digit), my-var (contains a hyphen), if (reserved keyword).*

*Variable Assignment:*

*Variables are created by assigning a value to them using the assignment operator (=).*

*The value can be of any data type, such as numbers, strings, lists, or other objects.*

*Variables do not need to be declared with a specific type. The type is inferred from the assigned value.*

*Here are a few examples of variable creation and assignment:*

*# Valid variable names and assignments*

*name = "John"*

*age = 25*

*is\_student = True*

*pi\_value = 3.14*

*# Invalid variable names*

*123abc = 10 # Invalid: Starts with a digit*

*my-var = "test" # Invalid: Contains a hyphen*

*if = 5 # Invalid: Reserved keyword*

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 followed for variable naming to improve code readability and maintainability. The most common conventions are outlined below:*

*Snake Case:*

*Variable names are typically written in all lowercase letters.*

*Words within the variable name are separated by underscores (\_).*

*Example: my\_variable, user\_name, total\_count.*

*Descriptive Names:*

*Variable names should be descriptive and reflect the purpose or meaning of the variable.*

*Use meaningful names that convey the intent and content of the variable.*

*Avoid using single character names (unless for specific purposes like loop variables).*

*Example: num\_items, customer\_name, is\_logged\_in.*

*Avoid Ambiguous Names:*

*Choose variable names that are clear and unambiguous.*

*Avoid using names that could be easily confused or misinterpreted.*

*For example, using names like l, O, or I can be confusing due to their similarity to the digits 1 and 0.*

*Avoid Reserved Keywords:*

*Do not use Python's reserved keywords as variable names.*

*Reserved keywords have special meanings in Python and cannot be used as variable names.*

*Example: if, for, while, def, etc.*

*Constants:*

*Constants (values that are not intended to be changed) are typically written in uppercase letters.*

*Multiple words in a constant name are separated by underscores (\_).*

*Example: MAX\_VALUE, PI, CONFIG\_FILE.*

*Consistency:*

*Maintain consistent naming conventions throughout your codebase.*

*Follow the same naming style for variables, functions, classes, and modules.*

*This helps improve code readability and makes it easier for others to understand your code.*

*While these conventions are widely followed, it's important to note that different projects and organizations may have their own specific naming conventions. It's always a good idea to adhere to the conventions used within your project or team to maintain consistency and readability in your code.*

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

Answer:

*If a keyword is used as a variable name in Python, it will result in a syntax error. Keywords in Python have predefined meanings and are reserved for specific purposes in the language. They cannot be used as variable names or identifiers.*

*When you attempt to use a keyword as a variable name, Python will raise a SyntaxError and point out the invalid usage. This error indicates that the keyword is not allowed in that context and needs to be replaced with a valid variable name.*

*For example, if you try to use the keyword if as a variable name, you would encounter a syntax error:*

*if = 10 # SyntaxError: invalid syntax*

*In this case, if is a keyword used for conditional statements in Python. Trying to assign a value to a variable with the name if violates the syntax rules, resulting in the syntax error.*

*To avoid such errors, it's important to choose meaningful and descriptive variable names that do not clash with Python's reserved keywords.*

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

Answer:

*The def keyword in Python is used to define a function. It is followed by the function name and a set of parentheses, which may include parameters for the function. The function definition is then followed by a colon (:) and an indented block of code that constitutes the body of the function.*

*Here's the general syntax of defining a function using the def keyword:*

*def function\_name(parameter1, parameter2, ...):*

*# Function body*

*# Code to be executed when the function is called*

*The def keyword marks the beginning of a function definition. It tells Python that you are defining a new function with the specified name and parameters.*

*Functions in Python allow you to encapsulate a set of instructions into a reusable block of code. They provide modularity and promote code organization by allowing you to break down a program into smaller, manageable pieces.*

*Once you define a function using def, you can call or invoke that function elsewhere in your code by using its name followed by parentheses, optionally passing arguments or values to the function parameters.*

*Here's an example of a simple function definition:*

*def greet(name):*

*print(f"Hello, {name}!")*

*greet("Alice") # Calling the function*

*In this example, the greet function takes a name parameter and prints a greeting message using that name. The function is defined using the def keyword, and later it is called with the argument "Alice", resulting in the output Hello, Alice!.*

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

Answer:

*The special character \ in Python is called the backslash. It is used as an escape character to indicate that the character following it has a special meaning or should be treated differently.*

*Here are some common uses of the backslash in Python:*

*Escape Sequences:*

*The backslash is used to create escape sequences for special characters.*

*Examples: \n for a newline, \t for a tab, \" for a double quote, \' for a single quote, \\ for a literal backslash, etc.*

*Multiline Statements:*

*The backslash can be used to split a long statement into multiple lines.*

*Example:*

*total = 10 + 20 + \*

*30 + 40 + \*

*50*

*Unicode Escape:*

*The backslash can be used to represent Unicode characters using escape sequences.*

*Example: \uXXXX to represent a Unicode character using its hexadecimal value.*

*Raw Strings:*

*The backslash can be used to create raw strings by prefixing a string literal with r.*

*Raw strings treat backslashes as literal characters, rather than escape characters.*

*Example: r"C:\path\to\file.txt" to represent a file path without interpreting the backslashes as escape sequences.*

*It's important to note that using a backslash in a string or character literal that does not have a valid escape sequence can result in a syntax error. In such cases, you may need to escape the backslash itself by using \\ to represent a literal backslash.*

*Overall, the backslash serves as an important tool for handling special characters, escape sequences, and formatting in Python strings.*

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

(i) Homogeneous list

(ii) Heterogeneous set

(iii) Homogeneous tuple

Answer:

*Here are examples of each condition:*

*(i) Homogeneous List:*

*A homogeneous list is a list that contains elements of the same data type. Here's an example of a homogeneous list of integers:*

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

*In this example, all the elements in the numbers list are integers.*

*(ii) Heterogeneous Set:*

*A heterogeneous set is a set that contains elements of different data types. Here's an example of a heterogeneous set:*

*my\_set = {1, 'apple', 3.14, True}*

*In this example, the set my\_set contains elements of different types, including an integer (1), a string ('apple'), a float (3.14), and a boolean (True).*

*(iii) Homogeneous Tuple:*

*A homogeneous tuple is a tuple that contains elements of the same data type. Here's an example of a homogeneous tuple of strings:*

*fruits = ('apple', 'banana', 'orange', 'grape')*

*In this example, all the elements in the fruits tuple are strings.*

*It's important to note that Python allows mixed data types in lists and tuples, but sets are designed to contain only unique elements, so they don't preserve the order or allow duplicates.*

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

Answer:

*In Python, data types can be categorized as either mutable or immutable based on whether their values can be changed after they are created. Here's an explanation of mutable and immutable data types along with examples:*

*Mutable Data Types:*

*Mutable data types are those whose values can be modified after they are created.*

*When you modify a mutable object, it changes its state without creating a new object.*

*Examples of mutable data types in Python include lists, sets, and dictionaries.*

*Example of a mutable data type: Lists*

*fruits = ['apple', 'banana', 'orange']*

*fruits.append('grape')*

*print(fruits) # Output: ['apple', 'banana', 'orange', 'grape']*

*In this example, the fruits list is mutable, and we can modify it by adding a new element using the append() method. The original list is changed, and a new element is added without creating a new list object.*

*Immutable Data Types:*

*Immutable data types are those whose values cannot be changed after they are created.*

*When you attempt to modify an immutable object, it results in the creation of a new object with the modified value.*

*Examples of immutable data types in Python include integers, floats, strings, and tuples.*

*Example of an immutable data type: Strings*

*message = "Hello"*

*new\_message = message + " World"*

*print(new\_message) # Output: Hello World*

*In this example, the message string is immutable. When we concatenate the string with " World", a new string object is created with the modified value. The original string remains unchanged.*

*Immutable data types are useful in scenarios where you want to ensure the integrity and consistency of the data, as their values cannot be accidentally modified. Mutable data types, on the other hand, allow you to modify the values in place, making them more flexible for situations that require frequent updates or modifications.*

*It's important to note that the mutability or immutability of a data type refers to the behavior of the objects of that type and not the variables themselves. Variables can be reassigned to reference different objects of the same or different types.*

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

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

Answer:

*The given structure resembles a pyramid pattern with increasing numbers of asterisks on each row. Here's the code to create that structure using a for loop:*

*rows = 5*

*for i in range(rows):*

*for j in range(i + 1):*

*print("\*", end="")*

*print()*

*Output:*

*\**

*\*\**

*\*\*\**

*\*\*\*\**

*\*\*\*\*\**

*Explanation:*

*The outer for loop iterates through the range of rows (in this case, 5), which determines the number of rows in the pyramid.*

*The inner for loop iterates through the range of i + 1, where i represents the current row number. This ensures that the number of asterisks printed on each row increases by one.*

*Within the inner loop, we print an asterisk (\*) without a newline character using print("\*", end="").*

*After printing the asterisks for each row, we use print() without any arguments to move to the next line and create a new row.*

*By executing this code, you will get the desired pyramid structure with increasing numbers of asterisks on each row*

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

|||||||||

|||||||

|||||

|||

|

Answer:

*The given structure resembles an inverted pyramid pattern with decreasing numbers of vertical bars on each row. Here's the code to create that structure using a while loop:*

*rows = 5*

*count = rows*

*while count >= 1:*

*print("|" \* count)*

*count -= 1*

*Output:*

*|||||||||*

*|||||||*

*|||||*

*|||*

*|*

*Explanation:*

*We initialize a variable count with the value of rows (in this case, 5).*

*The while loop continues executing as long as count is greater than or equal to 1.*

*In each iteration of the loop, we print count number of vertical bars (|) using the repetition operator \*.*

*After printing the vertical bars for each row, we decrement the value of count by 1.*

*The loop continues until count becomes less than 1, at which point the loop terminates.*

*By executing this code, you will get the desired inverted pyramid structure with decreasing numbers of vertical bars on each row using a while loop.*