20 May

Python Basic - 1

* 1. Q.1.What are keywords in python? Using the keyword library, print all the python keywords.
  2. Ans: In Python, keywords are reserved words that have specific meanings and purposes within the language. These keywords cannot be used as identifiers (such as variable names or function names) because they are already predefined with their respective functionalities. Python provides a library called keyword that allows you to work with and access the list of all keywords in the language.
  3. Here's an example of how you can use the keyword library to print all the Python keywords:
  4. Example:
  5. #Input
  6. import keyword
  7. all\_keywords = keyword.kwlist
  8. 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']

* 1. Q.2.What are the rules to create variables in python?
  2. Ans: In Python, there are a few rules to follow when creating variables:
  3. Variable names must start with a letter or an underscore (\_) character. They cannot start with a number or any other special character.
  4. Variable names can contain letters (both uppercase and lowercase), numbers, and underscores. They are case-sensitive, so "myVariable" and "myvariable" would be considered different variables.
  5. Variable names should be descriptive and meaningful, so they can convey the purpose or content of the data they represent. For example, instead of using generic names like "x" or "temp," it's better to use more descriptive names like "age," "username," or "total\_sales."
  6. Python keywords cannot be used as variable names. Keywords are reserved words that have a special meaning in Python. For example, "if," "while," "for," and "def" are all keywords and cannot be used as variable names.
  7. Variable names should not contain spaces. If you need to represent multiple words, you can use underscores between the words (e.g., "first\_name," "total\_sales").
  8. Q.3.What are the standards and conventions followed for the nomenclature of variables in python to improve code readability and maintainability?
  9. Ans: In Python, there are several standards and conventions followed for naming variables to enhance code readability and maintainability. Here are some commonly accepted practices:
  10. Use descriptive names: Variable names should be meaningful and describe the purpose or content of the variable. Avoid using single-character names or cryptic abbreviations that can make the code hard to understand.
  11. Follow snake\_case: Python convention suggests using lowercase letters with underscores (\_) to separate words in variable names. For example: my\_variable, user\_name, num\_of\_attempts.
  12. Use lowercase for most variables: By convention, most variable names (including function and method names) should be in lowercase. This helps differentiate variables from class names, which are typically written in CamelCase (discussed next).
  13. Capitalize class names: Class names should use CamelCase, which means starting each word with a capital letter and having no spaces or underscores. For example: MyClass, HttpRequest, CustomerOrder.
  14. Avoid reserved words: Avoid using Python's reserved keywords (e.g., if, for, while, class, etc.) as variable names, as they have predefined meanings in the language.
  15. Be consistent: Maintain consistency in your naming conventions throughout the codebase. This makes it easier for other developers to understand and work with your code.
  16. Use plural for collections: When naming variables that represent collections or sequences, use plural nouns. For example: users, items, countries.
  17. Avoid single leading underscores: Although a single leading underscore \_variable is allowed, it is typically used as a convention to indicate a variable or attribute that is intended for internal use within a class or module.
  18. Use uppercase for constants: Constants, which are variables whose values should not be changed, are conventionally written in uppercase letters with underscores. For example: MAX\_ATTEMPTS, PI, DEFAULT\_TIMEOUT.
  19. Avoid name clashes: Ensure that your variable names do not clash with names from imported modules or built-in functions to prevent unexpected behavior or conflicts.
  20. Q.4.What will happen if a keyword is used as a variable name?
  21. Ans: If a keyword is used as a variable name in a programming language, it would typically result in a syntax error or compilation error. Keywords are reserved words in programming languages that have predefined meanings and are used to define the language's syntax and structure. They cannot be used as variable names or identifiers.
  22. When you attempt to use a keyword as a variable name, the programming language's compiler or interpreter will recognize it as a reserved word and raise an error. This error indicates that you have violated the language's syntax rules by using a reserved word in a context where it is not allowed.
  23. Q.5.For what purpose def keyword is used?
  24. Ans: In Python, the def keyword is used to define a function. It is followed by the name of the function, parentheses that may contain parameters, and a colon. The function definition is then followed by an indented block of code that makes up the function's body.
  25. Q.6.What is the operation of this special character ‘\’?
  26. Ans: The special character '' is known as the backslash or escape character. It is used in various programming languages and systems to perform certain operations or indicate special sequences of characters. Here are some common uses of the backslash character:
  27. Escape sequences: The backslash is often used to introduce escape sequences, which represent special characters or control codes. For example, '\n' represents a newline character, '\t' represents a tab character, and '\' represents a literal backslash.
  28. Path separation: In file systems, the backslash is commonly used as a path separator in Windows-based systems. For example, "C:\Program Files" is a path that indicates the "Program Files" directory on the C drive.
  29. Character escaping: In some programming languages, the backslash can be used to escape special characters within strings. For example, if you want to include a double quote within a string literal enclosed in double quotes, you can escape it like this: "He said, "Hello!"".
  30. Regular expressions: In many programming languages, the backslash is used to escape special characters or define character classes within regular expressions. It allows you to match characters that have special meaning in regular expressions, such as '\*', '+', '?', etc.
  31. Line continuation: In some programming languages, the backslash can be used to continue a line of code onto the next line. This is often used for code readability when a line of code is too long to fit on a single line.
  32. Q.7.Give an example of the following conditions:

1. Homogeneous list
2. Heterogeneous set
3. Homogeneous tuple

Ans:

(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 in Python:

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

In this example, the list numbers contains only integers, making it a homogeneous list.

(ii) Heterogeneous set:

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

my\_set = {1, 2.5, "Hello", True}

In this example, the set my\_set contains an integer (1), a float (2.5), a string ("Hello"), and a boolean (True). Since the elements have different data types, it is a heterogeneous set.

(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 in Python:

fruits = ("apple", "banana", "orange")

In this example, the tuple fruits contains only strings representing different types of fruits. Therefore, it is a homogeneous tuple.

* 1. Q.8.Explain the mutable and immutable data types with proper explanation & examples.
  2. Ans:
  3. In programming, mutable and immutable are terms used to describe the behavior of data types in relation to their ability to be modified or changed after they are created. Let's explore these concepts with proper explanations and examples:
  4. Mutable Data Types:
  5. Mutable data types are those that can be modified or changed after they are created. This means you can add, remove, or update elements or properties of the object without creating a new object. Examples of mutable data types include lists, sets, and dictionaries in Python.
  6. Example:
  7. # Creating a mutable list
  8. my\_list = [1, 2, 3]
  9. print(my\_list) # Output: [1, 2, 3]
  10. # Modifying the list
  11. my\_list.append(4)
  12. print(my\_list) # Output: [1, 2, 3, 4]
  13. my\_list[0] = 5
  14. print(my\_list) # Output: [5, 2, 3, 4]
  15. In the example above, we create a list my\_list and then modify it by appending elements and updating existing elements. The list remains the same object, and its contents change.
  16. Immutable Data Types:
  17. Immutable data types are those that cannot be modified or changed after they are created. This means that once an immutable object is created, its value cannot be altered. Examples of immutable data types include strings, numbers (integers, floats), and tuples in Python.
  18. Example:
  19. # Creating an immutable string
  20. my\_string = "Hello"
  21. print(my\_string) # Output: Hello
  22. # Trying to modify the string
  23. my\_string += " World"
  24. print(my\_string) # Output: Hello World
  25. # Creating an immutable tuple
  26. my\_tuple = (1, 2, 3)
  27. print(my\_tuple) # Output: (1, 2, 3)
  28. # Trying to modify the tuple
  29. my\_tuple[0] = 4 # Raises an error
  30. In the example above, we create an immutable string my\_string and an immutable tuple my\_tuple. When we try to modify the string, a new string is created with the combined value, while the original string remains unchanged. However, when we try to modify the tuple, it raises an error because tuples are immutable.
  31. The choice between mutable and immutable data types depends on the requirements of your program. Immutable objects are useful in situations where you want to ensure that the value remains constant and prevent unintended modifications. Mutable objects, on the other hand, provide flexibility for modifying the data in-place.
  32. It's important to note that even though an object itself may be immutable, it can contain mutable elements. For example, a tuple is an immutable object, but it can contain mutable objects like lists within it. In such cases, the immutability applies to the tuple itself, but the mutable objects it contains can still be modified.
  33. Q.9.Write a code to create the given structure using only for loop.

\*

\*\*\*

\*\*\*\*\*

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

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

Ans:

n = 5

for i in range(n):

for j in range(n - i - 1):

print(" ", end="")

for k in range(2 \* i + 1):

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

print()

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

|||||||||

|||||||

|||||

|||

|

Ans:

row = 5

while row >= 1:

spaces = 5 - row

while spaces >= 1:

print(" ", end="")

spaces -= 1

col = 2 \* row - 1

while col >= 1:

print("|", end="")

col -= 1

print()

row -= 1