# Assignment - 6

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 special meaning and are part of the language syntax. These keywords cannot be used as identifiers (e.g., variable names or function names) because they are reserved for specific purposes in Python.

import keyword # Get all the Python keywords

keywords = keyword.kwlist # Print all the keywords

for kw in keywords:

print(kw)

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

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

Variable Name Rules:

Variable names can contain letters (both uppercase and lowercase), digits, and underscores (\_).

Variable names must begin with a letter (a-z, A-Z) or an underscore (\_). They cannot begin with a digit.

Variable names are case-sensitive. For example, myVar, myvar, and MyVar are different variables.

Variable names cannot be the same as Python keywords or reserved words. For example, you cannot name a variable if or for because they are reserved for specific purposes in Python.

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, following consistent naming conventions for variables improves code readability and maintainability. The PEP 8 style guide provides recommendations for variable naming conventions. Here are some standard practices:

Use descriptive names: Choose variable names that accurately describe the data they represent. This makes it easier for others (and yourself) to understand the purpose of the variable.

Use lowercase letters: Variable names should generally be in lowercase, with words separated by underscores (snake\_case). For example: my\_variable, user\_name, num\_iterations.

Avoid single-letter names: Except for generic loop variables (i, j, k), avoid single-letter variable names. Use descriptive names that convey the meaning of the variable.

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. Keywords are reserved words in the Python language and cannot be used as identifiers (such as variable names, function names, or class names) because they have special meanings and are part of the language syntax.

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

Ans. n Python, the def keyword is used to define functions. Functions are blocks of reusable code that perform a specific task when called. The def keyword marks the beginning of the function definition.

Ex:

def function\_name(parameters):

"""docstring"""

# Function body

# Statements

return value

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

Ans. In Python, the backslash (\) character is known as the escape character. It is used to escape special characters in strings and represent characters that are difficult to type directly. The backslash is followed by another character to create an escape sequence, which has a special meaning interpreted by Python.

Here are some common uses of the backslash (\) in Python:

Escape Sequences: The backslash is used to create escape sequences to represent special characters in strings. For example:

\n: Newline

\t: Tab

\': Single quote

\": Double quote

\\: Backslash itself

Multiline Strings: The backslash can be used to create multiline strings by escaping the newline character. This is useful when you want to break a long string into multiple lines for readability.

Unicode Characters: The backslash can be used to represent Unicode characters using their hexadecimal or Unicode code points.

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

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

Ans.

homogeneous\_list = [1, 2, 3, 4, 5],

heterogeneous\_set = {1, 'hello', 3.14},

homogeneous\_tuple = ('apple', 'banana', 'orange')

* 1. Explain the mutable and immutable data types with proper explanation & examples.

Ans. In Python, data types are classified as either mutable or immutable based on whether their values can be changed after they are created. Understanding the difference between mutable and immutable data types is essential for writing efficient and bug-free code.

Here's an explanation of mutable and immutable data types with examples:

Mutable Data Types:

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

This means that you can change the internal state or contents of the data type without creating a new object.

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

Example:

# Mutable list

my\_list = [1, 2, 3]

my\_list.append(4) # Modifying the list by adding an element

print(my\_list) # Output: [1, 2, 3, 4]

# Mutable dictionary

my\_dict = {'a': 1, 'b': 2}

my\_dict['c'] = 3 # Modifying the dictionary by adding a key-value pair

print(my\_dict) # Output: {'a': 1, 'b': 2, 'c': 3}

Immutable Data Types:

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

This means that once an immutable object is created, its state cannot be modified.

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

Example: # Immutable integer

my\_int = 10

# Attempting to modify the integer will result in an error

# my\_int += 1 # This will raise an error

# Immutable string

my\_str = "Hello"

# Attempting to modify the string will result in an error

# my\_str += " World" # This will raise an error

* 1. Write a code to create the given structure using only for loop.

\*

\*\*\*

\*\*\*\*\*

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

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

Ans.

rows = 5 # Number of rows in the structure

# Outer loop to iterate over each row

for i in range(rows):

# Inner loop to print '\*' for each column

for j in range(i + 1):

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

# Move to the next line after printing '\*' for the current row

print()

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

|||||||||

|||||||

|||||

|||

|

Ans.

rows = 5 # Number of rows in the structure

# Initial number of pipes

pipes = rows \* 2 - 1

# Outer loop to iterate over each row

while rows >= 1:

# Inner loop to print '|' for each column

i = 1

while i <= pipes:

print('|', end='')

i += 1

# Decrease the number of pipes for the next row

pipes -= 2

# Move to the next line after printing '|' for the current row

print()

# Move to the next row

rows -= 1