06 21 May basic 1

August 17, 2023

0.1 20 May

0.2 Python Basic - 1

Q.1. What are keywords in python? Using the keyword library, print all the python keywords. keywords are reserved words that have special meanings and purposes within the language. These keywords cannot be used as identifiers (variable names, function names, class names, etc.) because they are already predefined and have specific functionalities.

[1]:
```
import keyword

all_keywords = keyword.kwlist
print(all_keywords)
```

```
['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']
```

Q.2. What are the rules to create variables in python? the rules are very simple: - They must start with a letter (a-z, A-Z) or an underscore (_). - You cannot start a variable name with a digit (0-9) - You cannot use Python keywords mentioned (all_keywords = keyword.kwlist) -

Q.3. What are the standards and conventions followed for the nomenclature of variables in python to improve code readability and maintainability? In Python, variable names must adhere to certain rules and conventions. Here are the main rules for creating variables in Python:

1. **Valid characters**: Variable names can include letters (both lowercase and uppercase), digits, and underscores. They must start with a letter (a-z, A-Z) or an underscore (_). You cannot start a variable name with a digit (0-9).
2. **Case sensitivity**: Python is case-sensitive, which means that variables with different letter cases are considered distinct. For example, `myVariable`, `MyVariable`, and `myvariable` are all different variables.
3. **Reserved keywords**: You cannot use Python keywords as variable names since they have predefined meanings in the language.
4. **No spaces or special characters**: Variable names cannot contain spaces or special characters like !, @, #, $, %, etc.

5. **Length**: Variable names can be of any length, but it's a good practice to keep them concise and descriptive.

6. **Naming conventions**: While not enforced by the language, there are some naming conventions in Python to make code more readable. For example:

   - Use lowercase letters for variable names (e.g., `my_variable`).
   - Use underscores to separate words in a variable name (e.g., `student_name`, `total_score`).
   - Avoid using single-letter variable names (except in specific cases like loops).
   - For constants, use all uppercase letters with underscores (e.g., `PI`, `MAX_VALUE`).

[2]:
```
# Here are some valid examples of variable names:

name = "John"
age = 25
total_score = 98.5
is_student = True
```

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

[3]:
```
# there will be error SyntaxError invalid syntax
# example
if = 1
```

```
  Cell In[3], line 3
    if = 1
       ^
SyntaxError: invalid syntax
```

Q.5. For what purpose def keyword is used? The `def` keyword in Python is used to define functions

[4]:
```
def fun1():
    pass
```

Q.6. What is the operation of this special character ''? The special character '', known as the backslash, is used as an escape character in Python and many other programming languages. It allows you to include special characters in strings that would otherwise be difficult to represent directly. examples: - `\\` for a single backslash. - `\"` for a double quote - `\t` horizontal tab.

Q.7. Give an example of the following conditions: (i) Homogeneous list (ii) Heterogeneous set (iii) Homogeneous tuple

(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 containing integers:

[5]:
```
homogeneous_list = [1, 2, 3, 4, 5]
print(homogeneous_list)
```

```
[1, 2, 3, 4, 5]
```

(ii) Heterogeneous set A heterogeneous set is a set that contains elements of different data types. Here's an example of a heterogeneous set containing integers, strings, and floats:

[6]:
```
heterogeneous_set = {1, "hello", 3.14}
print(heterogeneous_set)
```

```
{1, 3.14, 'hello'}
```

(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 containing strings:

[7]:
```
homogeneous_tuple = ("apple", "banana", "orange")
print(homogeneous_tuple)
```

```
('apple', 'banana', 'orange')
```

Q.8. Explain the mutable and immutable data types with proper explanation & examples. Mutable data types are those whose values can be modified after creation. This means you can change the elements or content of a mutable object without creating a new object. Lists, dictionaries, and sets are examples of mutable data types.

[8]:
```
mutable_list = [1, 2, 3, 4]
print("Original List:", mutable_list)

# Modifying the list
mutable_list[0] = 100
mutable_list.append(5)
print("Modified List:", mutable_list)
```

```
Original List: [1, 2, 3, 4]
Modified List: [100, 2, 3, 4, 5]
```

[ ]:
```

```

Immutable data types are those whose values cannot be changed after creation. Once you create an immutable object, you cannot modify its content. Examples of immutable data types in Python include strings, tuples, and numbers (integers, floats, etc.).

[9]:
```
immutable_string = "Hello"
print("Original String:", immutable_string)

# Attempting to modify the string (will result in an error)
immutable_string[0] = "W"
```

```
Original String: Hello
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[9], line 5
      2 print("Original String:", immutable_string)
      4 # Attempting to modify the string (will result in an error)
----> 5 immutable_string[0] = "W"

TypeError: 'str' object does not support item assignment
```

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

[10]:
```
rows = 5

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

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

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

[11]:
```
rows = 5
i= 1
while i < rows+1:
    print(" " * (i -1), end="")
    print("|" * (2 *(rows-i) + 1))
    i += 1
```

```
|||||||||
 |||||||
  |||||
   |||
    |
```

[ ]:

[ ]: