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

- Keywords in Python are reserved words that have specific meanings and purposes within the language. These words cannot be used as variable names or other identifiers. They are an essential part of the Python syntax and define the structure and behavior of the language.

To print all the Python keywords, you can use the `keyword` module as follows:

```python

import keyword

all\_keywords = keyword.kwlist

print(all\_keywords)

```

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

In Python, the rules for creating variables are as follows:

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

- Variable names must start with a letter or an underscore, not with a digit.

- Variable names are case-sensitive. For example, `myVariable`, `myvariable`, and `MYVARIABLE` are treated as different variables.

- Variable names cannot be a Python keyword or reserved word (e.g., `if`, `else`, `while`, etc.).

- Avoid using special characters like `@`, `#`, `$`, etc., in variable names.

Q.3. What are the standards and conventions followed for the nomenclature of variables in Python to improve code readability and maintainability?

The standards and conventions followed for naming variables in Python are outlined in PEP 8, the official Python style guide. Here are some key points:

- Variable names should be in lowercase, and words should be separated by underscores (snake\_case).

- For constants, variable names should be in all uppercase with words separated by underscores (e.g., `MAX\_LENGTH`, `PI`, etc.).

- Use descriptive names that indicate the purpose or content of the variable (e.g., `counter`, `total\_amount`, etc.).

- Avoid using single characters as variable names, except for simple loop counters (`i`, `j`, `k`, etc.).

- When using acronyms in variable names, capitalize them as per their normal convention (e.g., `HTTPServer`, not `HttpServer`).

- For class names, use CamelCase (also known as PascalCase), where each word starts with an uppercase letter (e.g., `MyClass`, `UserData`, etc.).

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

If a keyword is used as a variable name, it will raise a syntax error. Python's lexer recognizes keywords as reserved words, and they cannot be used as identifiers or variable names. For example:

```python

for = 5 # This will raise a syntax error because "for" is a keyword.

```

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

The `def` keyword in Python is used to define user-defined functions. Functions are blocks of reusable code that perform a specific task or set of tasks. The `def` keyword is followed by the function name, a set of parentheses `()`, and a colon `:`. The function code is indented below the `def` statement.

Here's an example of defining a simple function using the `def` keyword:

```python

def greet(name):

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

greet("Alice") # Output: Hello, Alice!

greet("Bob") # Output: Hello, Bob!

```

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

In Python, the special character `\` (backslash) is known as the escape character. It is used to escape certain characters in strings and represents the beginning of an escape sequence. When `\` is followed by specific characters, it alters the meaning of those characters in the string. Some common escape sequences are:

- `\'`: Single quote

- `\"`: Double quote

- `\n`: Newline (line break)

- `\t`: Horizontal tab

- `\\`: Backslash itself (to escape the escape character)

For example:

```python

print("Hello, \"World\"!") # Output: Hello, "World"!

print('Hello, \'World\'!') # Output: Hello, 'World'!

print("Line 1\nLine 2") # Output: Line 1

# Line 2

```

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. For example:

```python

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

```

(ii) Heterogeneous set: A heterogeneous set is a set that contains elements of different data types. However, sets in Python only allow elements of hashable data types, so the elements need to be of immutable types. For example:

```python

heterogeneous\_set = {1, 'hello', 3.14, (1, 2, 3)}

```

(iii) Homogeneous tuple: A homogeneous tuple is a tuple that contains elements of the same data type. Tuples are immutable, and once created, their elements cannot be changed. For example:

```python

homogeneous\_tuple = (10, 20, 30, 40, 50)

```

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

In Python, data types can be categorized as mutable and immutable based on whether their values can be changed after creation.

- Mutable Data Types: Mutable data types allow their values to be modified after creation. This means you can change the elements or content of the data type without creating a new instance of it. Examples of mutable data types are lists, dictionaries, and sets.

```python

# Mutable list

my\_list = [1, 2, 3]

my\_list[0] = 10 # Modifying an element

print(my\_list) # Output: [10, 2, 3]

# Mutable dictionary

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

my\_dict['b'] = 20 # Modifying a value

print(my\_dict) # Output: {'a': 1, 'b': 20}

```

- Immutable Data Types: Immutable data types do not allow their values to be changed after creation. Once an immutable object is created, its state cannot be modified. Examples of immutable data types are integers, floats, strings, and tuples.

```python

# Immutable string

my\_string = "Hello"

# The following line will raise an error since strings are immutable.

my\_string[0] = 'h'

# Immutable tuple

my\_tuple = (1, 2, 3)

# The following line will raise an error since tuples are immutable.

my\_tuple[0] = 10

```

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

To create the pattern:

```

\*

\*\*\*

\*\*\*\*\*

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

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

```

You can use a for loop in Python to achieve this pattern:

```python

rows = 5

for i in range(1, rows + 1):

print("\*" \* (2 \* i - 1))

```

Output:

```

\*

\*\*\*

\*\*\*\*\*

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

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

```

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

To create the pattern:

```

|||||||||

|||||||

|||||

|||

|

```

You can use a while loop in Python to achieve this pattern:

```python

rows = 5

i = rows

while i > 0:

print("|" \* i)

i -= 1

```

Output:

```

|||||||||

|||||||

|||||

|||

|

```