- **created by**: Sudip Ghimire
- **URL**: https://www.sudipghimire.com.np
- **GitHub**: https://github.com/ghimiresdp


go to course contents

# Chapter 2.1: Python Variables and Constants

**Table of Contents**


- Chapter 2.1: Python Variables and Constants
  - Variables
  - Constants
  - Python Keywords


## Variables

let us suppose we have a statement as follows:

```
x = 5
```

here `x` is an identifier. It is also called as a variable since it's value can be changed while executing.

The above statement says set the value `5` to the variable `x` somewhere in a memory. **Python Memory Manager** manages the location where the values are stored.

There are few rules to create a variable name which are as follows:

1. Python identifiers can start with alphabetical characters.

   ```
   # Example
   name = "John Doe"
   age = 20
   ```

2. They can start with underscore `_` character.

   variables starting with `_` are generally used as protected attributes.

   ```
   # Example
   _name = "John Doe"
   _age = 20
   ```

3. Variables can not start with numeric characters but can exist in between or at the end.

```
# Example
# 1name = "John Doe"       # incorrect identifier name
name1 = "John Doe"       # correct syntax
na1me = "John Doe"       # correct syntax
```

4. We can't use special characters like `space tab + -` etc.

```
# The following are not allowed
"""
name one = "John Doe"       # incorrect identifier name
name+one = "John Doe"       # incorrect identifier name
name-one = "John Doe"       # incorrect identifier name
"""
# instead
```

we can use underscore character to separate 2 words name_one = "John Doe"

````
5. We use `snake_case` for variable definition

Even though python supports CamelCase identifier name
it is generally not recommended to use. PEP recommends using `snake_case`
identifiers over other type.



Some examples of identifiers supported by python are as follows

```python
name = 'cow'            # valid
_name = 'cow'           # valid
name1 = 'cow'           # valid
name_1 = 'cow'          # valid
# name 1 = 'cow'        # invalid
Name = 'cow'            # valid but not recommended by PEP
first_name = 'John'     # valid
firstName = 'John'      # valid but not recommended by PEP
FirstName = 'John'      # valid but not recommended by PEP
````

## Constants

There are no constants on python however we use UPPERCASE identifier to make developer know that the value shall never be changed.

```
PI = 3.1415
PROJECT_NAME = 'Python Notes'
```
```

```
PROJECT_VERSION = '1.0.0'
```

## Python Keywords

Python keywords are reserved words that cannot be used as variable names function names constants or any other identifiers.

Some of the keywords that are used in python are as follows:

| Keyword | Description |
|---|---|
| `False` | A boolean operator |
| `None` | Represents `null` value |
| `True` | A boolean oprator |
| `and` | A logical operator |
| `as` | Used to create an alias |
| `assert` | Used for testing for the right or wrong statement |
| `async` | Used for performing asynchronous operation |
| `await` | Used for getting the result of an async operation |
| `break` | Used to get out of the loop |
| `class` | Used to define the class |
| `continue` | Used to skip current iteration |
| `def` | Used to declare a function |
| `del` | Used to de-allocate the object from the memory |
| `elif` | An alternative statement for `if` |
| `else` | An altermative statement for `if` |
| `except` | Used to catch an exception |
| `finally` | Used to catch an exception |
| `for` | used to loop across the iterable |
| `from` | used to import specific part of the module |
| `global` | used to declare a global variable |
| `if` | used to create a conditional branching |
| `import` | used to import a module |
| `in` | an associative identifier |
| `is` | an identity operator |

| Keyword | Description |
| --- | --- |
| `lambda` | used to create an anonymous function |
| `nonlocal` | used to create a variable of parent's scope |
| `not` | a logical operator |
| `or` | a logical operator |
| `pass` | used as an empty body of a code block |
| `raise` | used to raise an exception |
| `return` | used to return a value from the function |
| `try` | used to try a statement before it raises an error |
| `while` | used to initialize a loop |
| `with` | used to simplify statements |
| `yield` | used to generate values to perform lazy execution |