Variables in python.

Variable are containers for storing data values.

Python has no commands to declaring a variable.

`Rules:`

- A variable name must start with `a letter` or the `underscore` character.
- A variable name cannot start with a number.
- A variable name can only contain `alpha-numeric` characters and underscores `(A-z, 0-9, and \_ )`
- Variable names are case-sensitive (age, Age and AGE are three different variables).
- A variable name cannot be any of the `Python keywords`.

`Example:`

```
# integer variable
x = 90;

# float variable
y = 9.90;

# boolean variable
isPython = True

# string variable
name = "Monty python flying circus"

# list
names = ["Dilip","Pradeep","Sanjeev","Arun"]

# tuple
skills = ("A","B","C","D")

# set
users = {"PK","DK","CK","AP"}

# dictionary
obj = {
    "name": "python",
    "author": "Guido van Rossum",
    "year": 1991
}

# Multiword variable can be written in `camelCase`, `PascalCase` or `kebab_case`.

# Many value to multiple variables
x,y,z = 10,'Hello world',34.6

print(x,y,z)
```

```
# One Value to Multiple Variables
a=b=c=10
print(a,b,c)
```

Variable Scope:

Variables in Python have a scope, which determines where they can be accessed or modified.

`Global Scope:` Variables defined outside any function or class are in the global scope and can be accessed anywhere within the module.

`Local Scope:` Variables defined inside a function are in the local scope and can only be accessed within that function.

Mutable vs Immutable:

`Immutable:` Variables whose values cannot be changed once assigned. Examples include integers, floats, strings, and tuples.

`Mutable:` Variables whose values can be changed after assignment. Examples include lists, dictionaries, and sets.

```
# Immutable example
x = 5
x = 10  # This creates a new integer object with value 10

# Mutable example
my_list = [1, 2, 3]
my_list.append(4)  # Modifies the existing list
```

Variable Reference:

Python variables are references to objects in memory. When you assign a variable, you're binding that variable name to the object. Multiple variables can refer to the same object.

```
a = [1, 2, 3]
b = a  # Both a and b now refer to the same list object

b.append(4)
print(a)  # Output: [1, 2, 3, 4], because a and b are referencing the same list
object
```

Casting:

If you want to specify the data type of a variable, this can be done with `casting`.

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

Deleting Variables:

You can delete a variable using the `del` statement.

```
x = 5
del x  # Deletes the variable x
```

Constants:

Python does not have constants in the strict sense, but variables that are intended to be constant are typically named in uppercase to indicate their intended immutability.

```
PI = 3.14
```

None:

None is a special constant in Python that represents the absence of a value.

```
x = None
```

Get the type:

```
x = 5
y = "John"
print(type(x)) # <class int>
print(type(y)) # <class string>
```

Global variable:

Variables that are created outside of a function are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

`Example:`

```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function.

`The global Keyword:`

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a `global` variable inside a `function`, you can use the `global` keyword.

```
x = 90;

def main():
    global x
    x=99;
    print(x)

print(x);
```

`Note:`