

Lecture#4 Python Programming

Python Variables: How to Define/Declare String Variable Types

What is a Variable in Python?

A Python variable is a reserved memory location to store values. In other words, a variable in a python program gives data to the computer for processing.

Python Variable Types

Every value in Python has a datatype. Different data types in Python are Numbers, List, Tuple, Strings, Dictionary, etc. Variables in Python can be declared by any name or even alphabets like a, aa, abc, etc.

In this lecture, we will learn,

- How to Declare and use a Variable
- Re-declare a Variable
- Concatenate Variables
- Local & Global Variables
- Delete a variable

How to Declare and use a Variable

Let see an example. We will define variable in Python and declare it as “a” and print it.

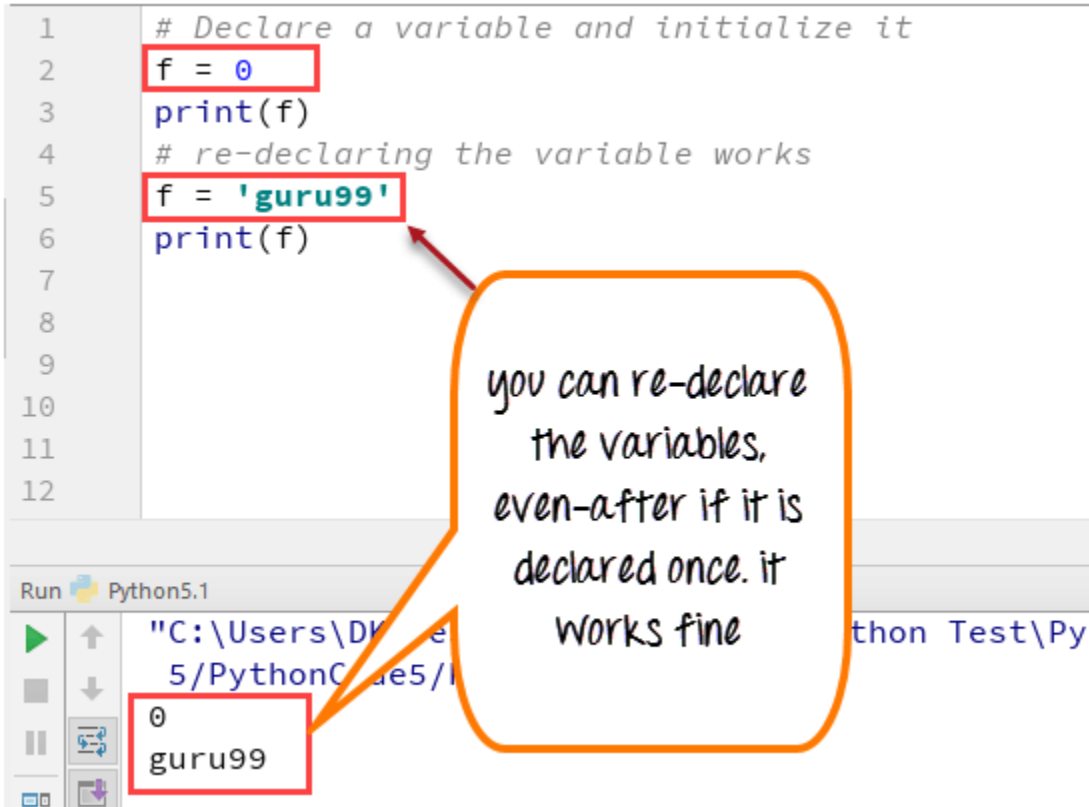
```
a=100  
print (a)
```

Re-declare a Variable

You can re-declare Python variables even after you have declared once.

Here we have Python declare variable initialized to f=0.

Later, we re-assign the variable f to value "guru99"



The screenshot shows a Python IDE with a code editor and a console. The code editor contains the following code:

```
1 # Declare a variable and initialize it
2 f = 0
3 print(f)
4 # re-declaring the variable works
5 f = 'guru99'
6 print(f)
```

The lines `f = 0` and `f = 'guru99'` are highlighted with red boxes. A speech bubble points to the second line with the text: "you can re-declare the variables, even-after if it is declared once. it works fine". The console shows the output of the code:

```
Run Python5.1
"C:\Users\...e
5/PythonCode5/l
0
guru99
```

The output shows the value 0 on the first line and guru99 on the second line, indicating that the variable f was successfully re-declared.

Python 2 Example

```
# Declare a variable and initialize it
f = 0
print f
# re-declaring the variable works
f = 'guru99'
print f
```

Python 3 Example

```
# Declare a variable and initialize it
f = 0
print(f)
# re-declaring the variable works
f = 'guru99'
print(f)
```

Python String Concatenation and Variable

Let's see whether you can concatenate different data types like string and number together. For example, we will concatenate "Guru" with the number "99".

Unlike Java, which concatenates number with string without declaring number as string, while declaring variables in Python requires declaring the number as string otherwise it will show a `TypeError`



The screenshot shows a Python IDE with a code editor and a console window. In the code editor, line 10 contains the code `print("guru"+99)`. A red box highlights this line with the text `#ERROR: different types cannot be combined`. The console window shows the output of the program, which is `guru99`. A red box highlights the error message in the console: `TypeError: must be str, not int`. A speech bubble points to this error message with the text: "it shows type error as #\"99\" is not declared as string".

```
6 print(f)
7
8
9 #ERROR: different types cannot be combined
10 print("guru"+99)
11
12
```

Run Python5.1

"C:\Users\DK\Desktop\Python code\PythonCode5\PythonCode5\Python5.1.py"

Traceback (most recent call last):

0

File "C:/Users/DK/Desktop/Python code/PythonCode5/PythonCode5/Python5.1.py", line 10, in <module>

guru99

print("guru"+99)

TypeError: must be str, not int

it shows type error as #\"99\" is not declared as string

For the following code, you will get undefined output –

```
a="Guru"  
b = 99  
print a+b
```

Once the integer is declared as string, it can concatenate both “Guru” + str(“99”) = “Guru99” in the output.

```
a="Guru"  
b = 99  
print(a+str(b))
```

Python Variable Types: Local & Global

There are two types of variables in Python, Global variable and Local variable. When you want to use the same variable for rest of your program or module you declare it as a global variable, while if you want to use the variable in a specific function or method, you use a local variable while Python variable declaration.

Let's understand this Python variable types with the difference between local and global variables in the below program.

1. Let us define variable in Python where the variable “f” is global in scope and is assigned value 101 which is printed in output
2. Variable f is again declared in function and assumes local scope. It is assigned value “I am learning Python.” which is printed out as an output. This Python declare variable is different from the global variable “f” defined earlier
3. Once the function call is over, the local variable f is destroyed. At line 12, when we again, print the value of “f” is it displays the value of global variable f=101

```
1 # Declare a variable and initialize it
2 f = 101
3 print(f)
4
5 # Global vs. local variables in functions
6 def someFunction():
7     # global f
8     f = 'I am learning Python'
9     print(f)
10
11 someFunction()
12 print(f)
13
```

Run Python5.2

"C:\Users\DK\Desktop\Python code\Python Test\Python 5\PythonCode5\PythonCode5\Python5.2.py"

101
I am learning Python
101

f is a local variable declared inside the function.

Python 2 Example

```
# Declare a variable and initialize it
f = 101
print f

# Global vs. local variables in functions
def someFunction():
    # global f
    f = 'I am learning Python'
    print f

someFunction()
print f
```

Python 3 Example

```
# Declare a variable and initialize it
f = 101
print(f)
# Global vs. local variables in functions
def someFunction():
    # global f
    f = 'I am learning Python'
    print(f)
someFunction()
print(f)
```

While Python variable declaration using the keyword global, you can reference the global variable inside a function.

- 1. Variable “f” is global in scope and is assigned value 101 which is printed in output**
- 2. Variable f is declared using the keyword global. This is NOT a local variable, but the same global variable declared earlier. Hence when we print its value, the output is 101**
- 3. We changed the value of “f” inside the function. Once the function call is over, the changed value of the variable “f” persists. At line 12, when we again, print the value of “f” is it displays the value “changing global variable”**

The screenshot shows a Python IDE with the following code:

```
1 f = 101;
2 print(f)
3
4 # Global vs.local variables in functions
5 def someFunction():
6     global f
7     print(f)
8     f = "changing global variable"
9
10 someFunction()
11 print(f)
```

Annotations in the image:

- Red circle 1 points to line 2 (`print(f)`).
- Red circle 2 points to line 6 (`global f`).
- Red circle 3 points to line 11 (`print(f)`).
- A yellow highlight is on line 8 (`f = "changing global variable"`).
- An orange callout bubble points to line 8 with the text: "We are now accessing and changing the global variable f."
- A green dashed line connects line 2 to line 11.
- A green dashed line connects line 8 to line 11.

The output console shows the following:

```
Run Python5.3
"C:\Users\DK\Desktop\Python code\Python Test\Python 5\Pyt
5/PythonCode5/Python5.3.py"
101
101
changing global variable
```

Python 2 Example

```
f = 101;
print f
# Global vs.local variables in functions
def someFunction():
    global f
    print f
    f = "changing global variable"
someFunction()
print f
```

Python 3 Example

```

f = 101;
print(f)
# Global vs.local variables in functions
def someFunction():
    global f
    print(f)
    f = "changing global variable"
someFunction()
print(f)

```

Delete a variable

You can also delete Python variables using the command `del` “variable name”.

In the below example of Python delete variable, we deleted variable `f`, and when we proceed to print it, we get error “variable name is not defined” which means you have deleted the variable.

The screenshot shows a Python IDE with the following code in the editor:

```

1  #Declare a variable and initialize it
2  f = 11;
3  print(f)
4
5
6
7  del f
8  print(f)
9

```

The code is executed, and the output window shows the following:

```

Run Python5.4
"C:\Users\DK\Desktop\Python code\Python
5/PythonCode5/Python5.4.py"
11
Traceback (most recent call last):
  File "C:/Users/DK/Desktop/Python code/
  print(f)
NameError: name 'f' is not defined

```

A red box highlights the `del f` and `print(f)` lines in the code editor. Another red box highlights the `NameError: name 'f' is not defined` message in the output window. A speech bubble points to the error message with the text: "Once you delete variable f and print f it will show this comment, which means your variable is now deleted".

Example of Python delete variable or Python clear variable :


```
f = 11;  
print(f)  
del f  
print(f)
```

Summary:

- Variables are referred to “envelop” or “buckets” where information can be maintained and referenced. Like any other programming language Python also uses a variable to store the information.
- Variables can be declared by any name or even alphabets like a, aa, abc, etc.
- Variables can be re-declared even after you have declared them for once
- Python constants can be understood as types of variables that hold the value which can not be changed. Usually Python constants are referenced from other files. Python define constant is declared in a new or separate file which contains functions, modules, etc.
- Types of variables in Python or Python variable types : Local & Global
- Declare local variable when you want to use it for current function
- Declare Global variable when you want to use the same variable for rest of the program
- To delete a variable, it uses keyword “del”.