**Variables**

* Variables are nothing but reserved memory locations to store values
* Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory

# Variables Definition and Initialization

* Python variables do not need explicit declaration to reserve memory space
* The declaration happens automatically when you assign a value to a variable
* You can change the value of a variable in your program at any time, and Python will always keep track of its current value.

#!/usr/bin/python

counter = 100 # An integer assignment

miles = 1000.0 # A floating point

name = "John" # A string

print counter

print miles

print name

Output:

100

1000.0

John

# Multiple Assignment

Single value to several variables simultaneously

a = b = c = 1

Multiple value to multiple variables simultaneously

a, b, c = 3, 2.56, "john"

#!/usr/bin/python

a = b = c = 1

print a

print b

print c

print "\n"

a, b, c = 3, 2.56, "john"

print a

print b

print c

Output:

1

1

1

3

2.56

john

# Global and Local Variables

**Global variables** defined and declared outside a function and we need to use them inside a function.

#!/usr/bin/python

# This function uses global variable s

def f():

print s

# Global scope

s = "I love Geeksforgeeks"

f()

Output:

I love Geeksforgeeks

If a variable with same name is defined inside the scope of function as well then it will print the value given inside the function only and not the global value.

#!/usr/bin/python

# This function has a variable with name same as s.

def f():

s = "Me too."

print s

# Global scope

s = "I love Geeksforgeeks"

f()

print s

Output:

Me too.

I love Geeksforgeeks

**what will happen, if we change the value of s inside of the function f()?**

#!/usr/bin/python

def f():

print s

# This program will NOT show error if we comment below line.

s = "Me too."

print s

# Global scope

s = "I love Geeksforgeeks"

f()

print s

Runtime Errors:

Traceback (most recent call last):

File "/home/580423c254cc0aef0e80d3b1ddec63fa.py", line 12, in <module>

f()

File "/home/580423c254cc0aef0e80d3b1ddec63fa.py", line 3, in f

print s

UnboundLocalError: local variable 's' referenced before assignment

Any variable which is changed or created inside of a function is local, if it hasn’t been declared as a global variable. To tell Python, that we want to use the global variable, we have to use the keyword “global”,

We only need to use global keyword in a function if we want to do assignments / change them.

**global is not needed for printing and accessing. Why?**

Python “assumes” that we want a local variable due to the assignment to s inside of f(), so the first print statement throws this error message. Any variable which is changed or created inside of a function is local, if it hasn’t been declared as a global variable.

#!/usr/bin/python

def f():

global s

print s

# This program will NOT show error if we comment below line.

s = "Me too."

print s

# Global scope

s = "I love Geeksforgeeks"

f()

print s

Output:

I love Geeksforgeeks

Me too.

Me too.

# Print Single and Multiple variable

**For multiple variable**:

* “print variable” prints the variables without any brackets ‘()’ and splitted by a space
* “print(variable)” prints the variables with brackets ‘()’ and splitted by a coma ‘,’ so it’s treated as a tuple

#!/usr/bin/python

print 1

print (1)

print ((1))

print "\n"

print 1, 2

print (1, 2)

print ((1, 2))

Output:

1

1

1

1 2

(1, 2)

(1, 2)

**Note:** In Python 3.0, the print statement is changed to print() function. Below are equivalent codes in Python 3.0.

#!/usr/bin/python

#print 1 # SyntaxError: Missing parentheses in call to 'print'

print (1)

print ((1))

print("\n")

#print 1, 2 # SyntaxError: Missing parentheses in call to 'print'

print (1, 2)

print ((1, 2))

Output:

1

1

1 2

(1, 2)

# Swap two variables in one line

In Python, there is a simple and syntactically neat construct to swap variables

“x, y = y, x”

#!/usr/bin/python

x = 3

y = 5

print x

print y

x, y = y, x

print x

print y

Output:

3

5

5

3