Python Programming Session 3

Seyyed Ali Shohadaalhosseini

TOPICS Python Fundamentals and Programming

- ❖ Values, Variables, operators

Values or Data

The main and brain of everythings in programming is **value,** Especially when it’s a real data.

All we want to do is working with our value in our program to reach the goals and solutions for our problem.

We want to use values to make a program which finds the problems’ solution.

Data type

A value can have any type, but we always know its’ type actually we determine value type.

In python we have following data type built-in by default:

Text Type: str	Sequence Type: list, tuple, range	Set Types: set, frozenset
Numeric Type: int, float, complex	Mapping Type: dict	Boolean Type: bool

Binary Types: bytes, bytearray, memoryview

Example of each data type #1

Example	Data Type
“ Hello World “	str
23	int
20.0	float
1j	complex
[“Ali”, “Mohammad”, “Naghi”]	list
(“name”, “Family”)	tuple
range(6)	rang

Example of each data type #2

Example	Data Type
{“name” : “Ali”, “age” : 15}	dict
{“name”, “Ali”, “age”, 15}	set
frozenset({ “name”, “Ali”, “age”, 15 })	frozenset
True or False	bool
b”Hello”	bytes
bytearray(5)	bytearray
memoryview(bytes(5))	memoryview

Variables

We can not use these data and values directly in our program, So how we can use data in our program? **Variables.**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Base on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, float, string, and other type of value.

Assigning Values to Variables

So by now we have understand what are values and variables, and how we can use variables to use values in our program. Now the question is how can we assign values to a variables ?

In python we don't need explicit declaration for variables to reserve memory space. The declaration happens automatically when you assign a value to a variable. And, to assign a value to a variables in python, we use a single equal sign (=) .

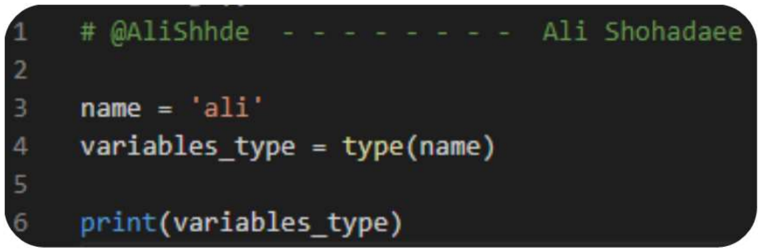
The operand on the left of the = operator is the name of the variable and the operand to the right of the = is the value stored in the variable.

Example: **Name = 'Ali'**

How to understand the type of the variables

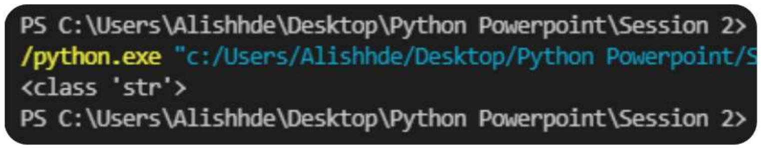
If you want to know the types of the variable, you can figure this out by by using the *type()* method.

Example: **Code**



```python
# @AliShhde - - - - - - - -  Ali Shohadaee

name = 'ali'
variables_type = type(name)

print(variables_type)
```

Output



```
PS C:\Users\Alishhde\Desktop\Python Powerpoint\Session 2>
/python.exe "c:/Users/Alishhde/Desktop/Python Powerpoint/S
<class 'str'>
PS C:\Users\Alishhde\Desktop\Python Powerpoint\Session 2>
```

Some Notation

★ Single or Double Quotes
  ○ String variables can be declared either by using single or double quotes.
★ Case-Sensitive
  ○ Variable names are case-sensitive
★ If you want to see the length of the variable, you can use the ***len()*** method

Variable Names

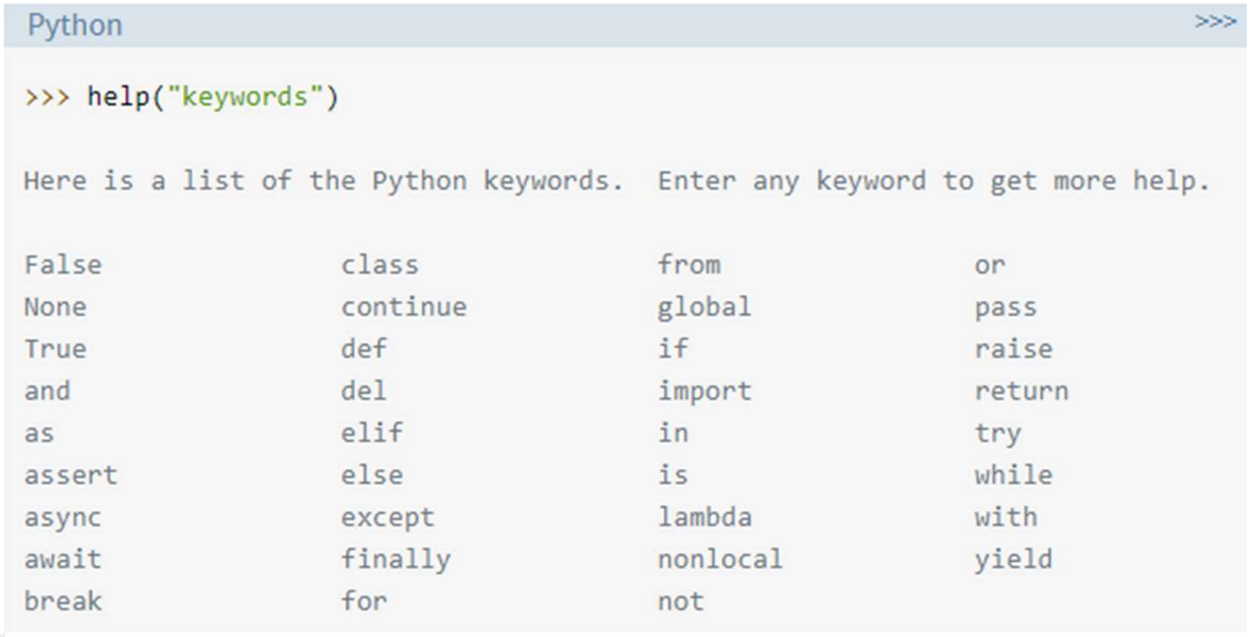
A variable can have a short name like: **x** or **y**, or a more descriptive name like: **age**, **carname**, **total_sum.**

Some rules for python variables:

- A variable name must start with a letter or the underscore _ character
- A variable name can only contain alpha-numeric characters and underscores (A -- z, 0-9 and _)
- A variable name can not start with a number
- Variable names are case-sensitive (age, Age and AGe are three different variables.
- We cannot use python's keyword's name as a variable.

Python keywords

- We can not use these name as a variable name.



Python >>>

```
>>> help("keywords")

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not
```

Multi Words Variable Name

Variable name with more than one word can be difficult to read. There are several techniques you can use to make them more readable:

1. **Camel Case**

Each word, except the first, starts

with a capital letter.

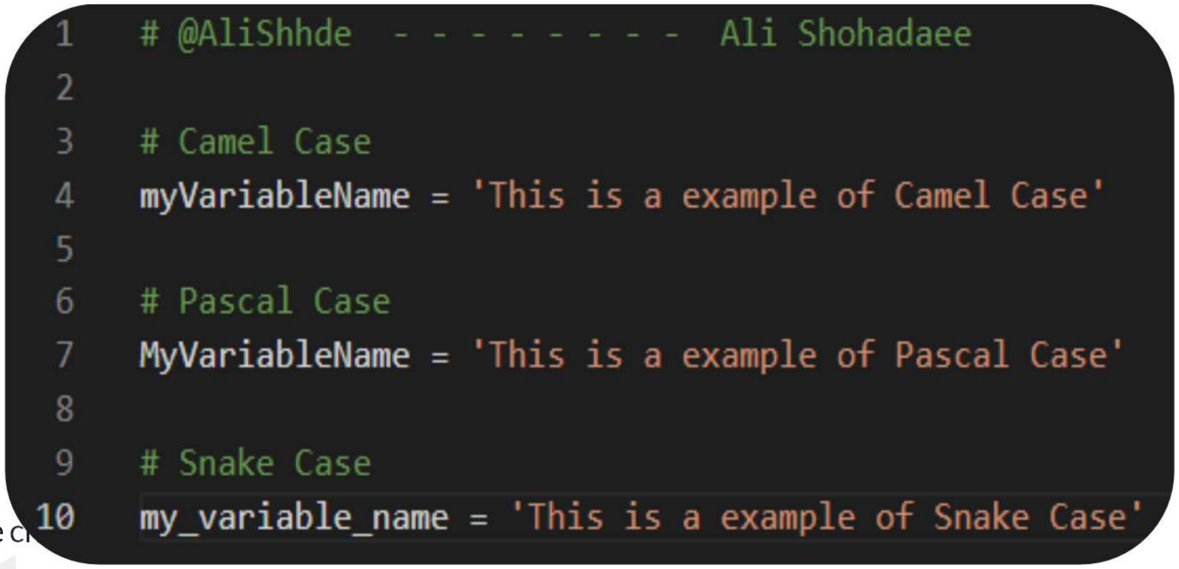
1. **Pascal Case**

Each word starts with a capital

letter.

1. **Snake Case**

Each word is separated by an underscore c

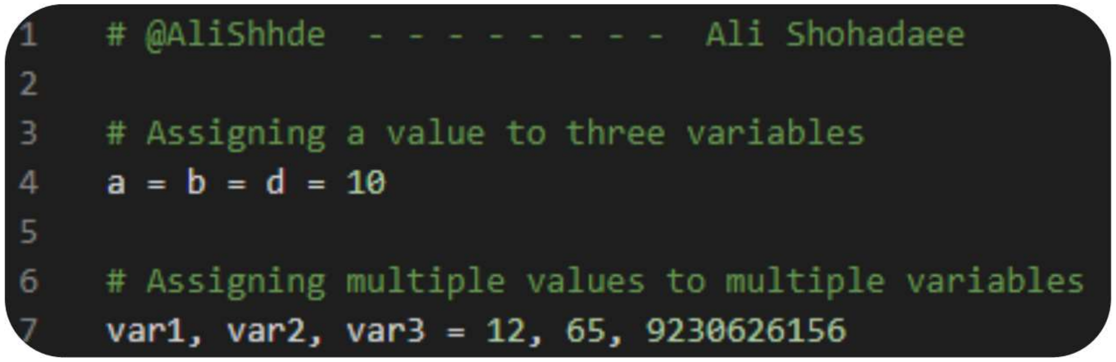


```python
# @AliShhde  - - - - - - - -  Ali Shohadaee

# Camel Case
myVariableName = 'This is a example of Camel Case'

# Pascal Case
MyVariableName = 'This is a example of Pascal Case'

# Snake Case
my_variable_name = 'This is a example of Snake Case'
```

Multiple Assignment to variables

Python allows us to assign a single value to several variables, Or several values to several variables.

Example:

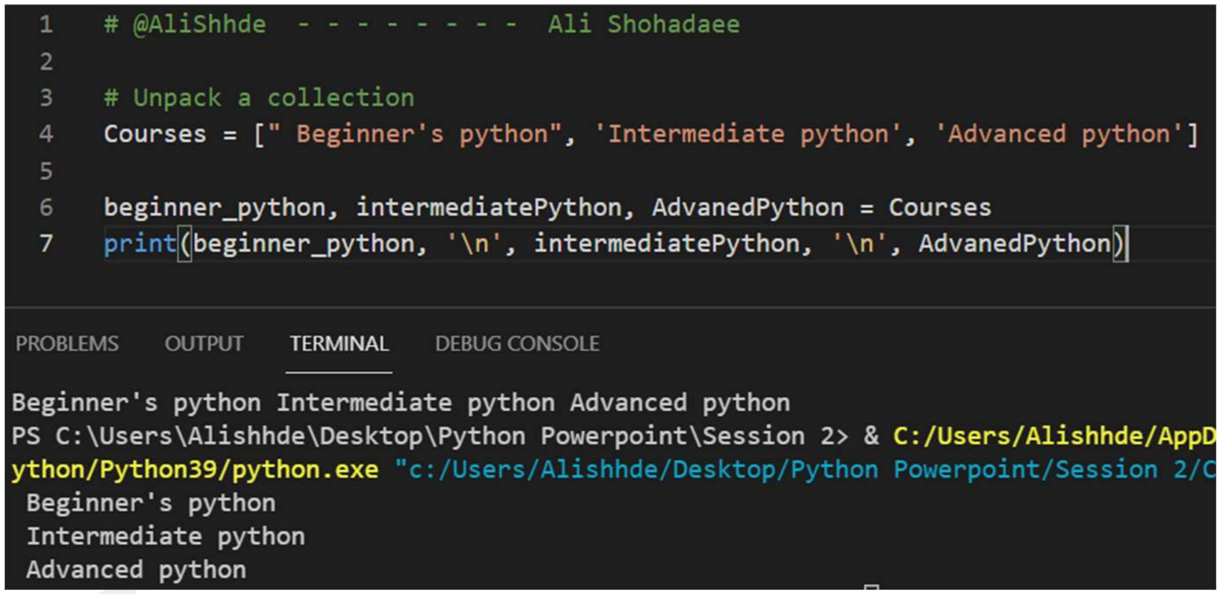


```
# @AliShhde  - - - - - - - -  Ali Shohadaee

# Assigning a value to three variables
a = b = d = 10

# Assigning multiple values to multiple variables
var1, var2, var3 = 12, 65, 9230626156
```

Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you extract the values into variables. This is called unpacking.

Example:



```python
# @AliShhde - - - - - - - -  Ali Shohadaee

# Unpack a collection
Courses = [" Beginner's python", 'Intermediate python', 'Advanced python']

beginner_python, intermediatePython, AdvanedPython = Courses
print(beginner_python, '\n', intermediatePython, '\n', AdvanedPython)
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
Beginner's python Intermediate python Advanced python
PS C:\Users\Alishhde\Desktop\Python Powerpoint\Session 2> & C:/Users/Alishhde/AppD
ython/Python39/python.exe "c:/Users/Alishhde/Desktop/Python Powerpoint/Session 2/C
 Beginner's python
 Intermediate python
 Advanced python
```

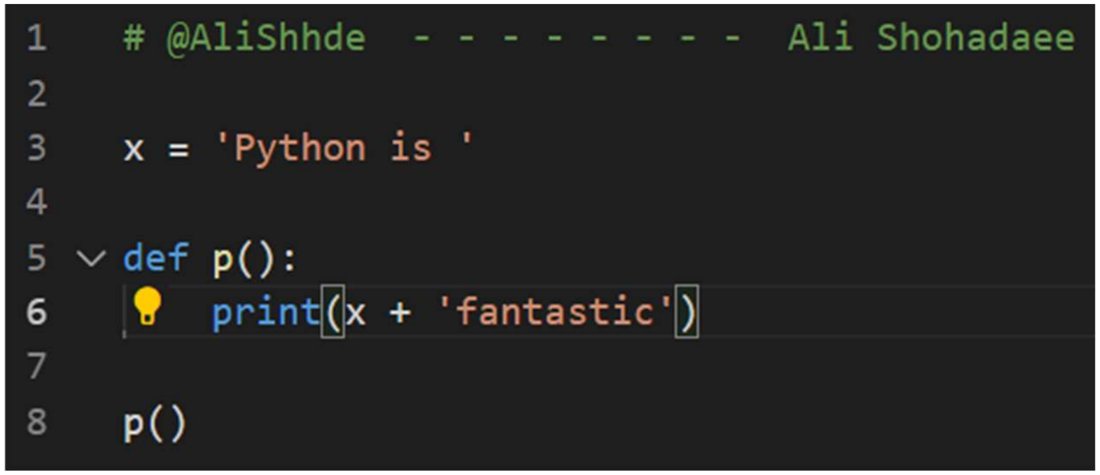
Global Variables

Variables that are created outside of a function are known as global variables. Global variables can be used by everyone, both inside of function and outside.

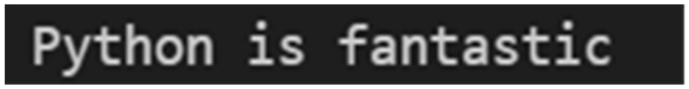
If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value.

Let’s see an example of each...

Using globally

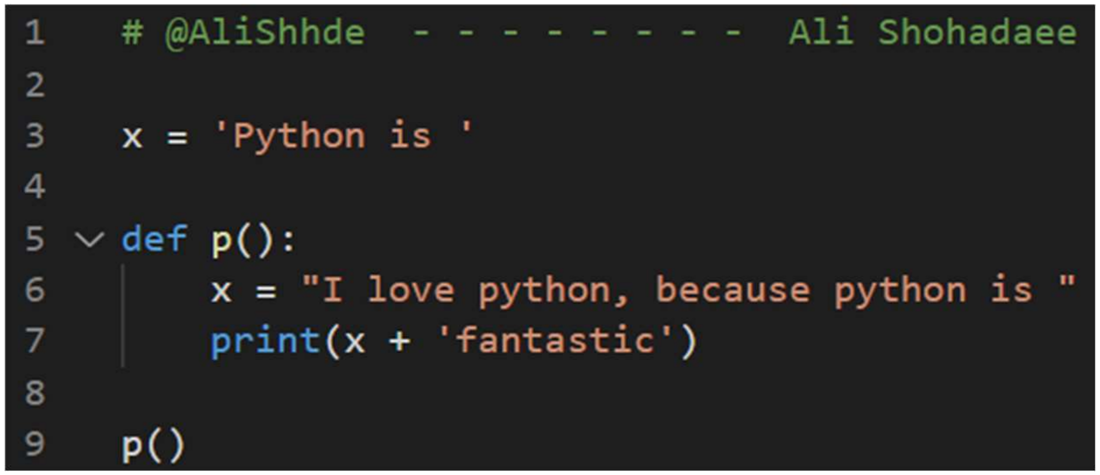


```python
# @AliShhde - - - - - - - - Ali Shohadaee

x = 'Python is '

def p():
    print(x + 'fantastic')

p()
```

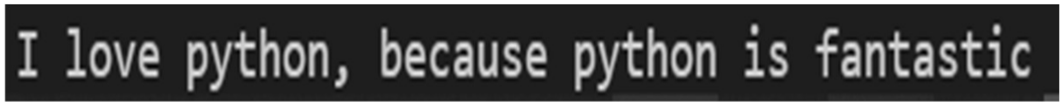


```
Python is fantastic
```

Using locally



```python
# @AliShhde - - - - - - - - Ali Shohadaee

x = 'Python is '

def p():
    x = "I love python, because python is "
    print(x + 'fantastic')

p()
```



```
I love python, because python is fantastic
```

We Also Have a Global Keyword

But I don’t know it, I just know that there is kind of keyword, Let’s together find this out!

Casting

Python is an object-oriented language, and as such it uses classes to define data types, including its primitive types.

- int() - Construct an integer number from an integer, float and string.
- float() - Construct a float number from an integer, float and string.
- str() - Construct a string from a wide variety of data types, including strings, integer numbers and float numbers.

Well see it soon...

Booleans value

We have two booleans value: `True` or `False`

Many times in programming we need to know about that a expression is `True` or `False`.

When we evaluate any expression in python, we expected one of these two answer. When we compare two values, the expression is evaluated and python returns the boolean answer.

Search for bool() function.

Most Values are True

Almost any value is evaluated to `True` if it has some sort of content.

- Any String is True, except empty strings
- Any number is True, except 0
- Any list, tuple, set, and dictionary are True, except empty ones

Operators

Operators are used to perform operations on variables and values.

Python divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Arithmetic Operators

Operator	Name	Example
+	Addition	X + Y
-	Subtraction	X - Y
*	Multiplication	X * Y
/	Division	X / Y
%	Reminder	X % Y
**	Exponentiation	X ** Y
//	Floor Division	X // Y

Python Assignment Operators

Operator	Example	Same as
=	X = 5	X = 5
+=	X += 3	X = X + 3
-=	X -= 3	X = X - 3
*=	X *= 3	X = X * 3
/=	X /= 3	X = X / 3
%=	X %= 3	X = X % 3
//=	X //= 3	X = X // 3

Operator	Example	Same as
**=	X **= 3	X = X ** 3
&=	X &= 3	X = X & 3
|=	X |= 3	X = X | 3
^=	X ^= 3	X = X ^ 3
>>=	X >>= 3	X = X >> 3
<<=	X <<= 3	X = X << 3

Python Comparison Operators

Comparison operators are used to compare two values.

Operator	Name	Example
==	Equal	X == Y
!=	Not equal	X != Y
>	Greater than	X > Y
<	Less than	X < Y
>=	Greater than or equal to	X >= Y
<=	Less than or equal to	X <= Y

Python Logical Operators

Logical operators are used to combine conditional statements.

Operator	Description	Example
and	Returns True if both statements are true	X < 5 and X < 10
or	Returns True if one of the statements is true	X < 5 or X < 10
not	Reverse the result, returns False if the result is true	not(X < 5 and X < 10)

Python Identity Operators

Identity operators are used to compare the objects, not if they are equal, but if they are actually the same object, with the same memory location.

Operator	Description	Example
is	Returns True if both variables are the same object	X is Y
is not	Returns True if both variables are not the same object	X is not Y

Python Membership Operators

Membership operators are used to test if a sequence with the specified value is present in the object or not.

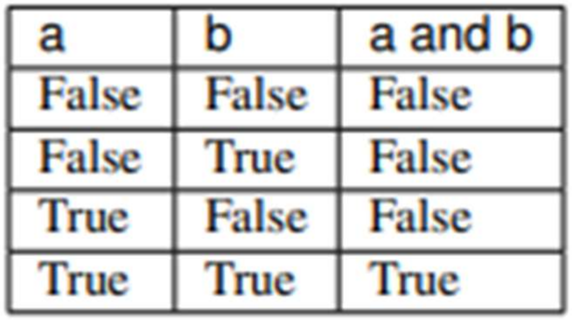
Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	X in Y
not in	Returns True if a sequence with the specified value is not present in the object	X not in Y

Python Bitwise Operators

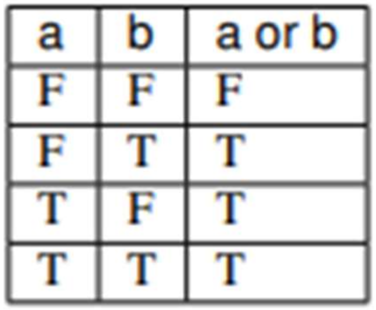
Bitwise operators are used to compare (binary) numbers.

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the left most bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

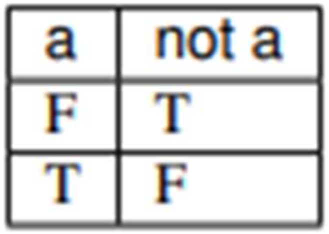
Truth Tables



a	b	a and b
False	False	False
False	True	False
True	False	False
True	True	True



a	b	a or b
F	F	F
F	T	T
T	F	T
T	T	T



a	not a
F	T
T	F