Python

By Efrem G.
Biniam G.

Python

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

● web development (server-side),

● AI, Data science libraries and frameworks,

● Connect to Databases

● Big Data Analytics (example pyspark)

Main Building Blocks or Foundations of Programming

1. Variables / ---> data types
2. Conditional Statements ---> if/else
3. Loops ---> for loop, while loop
4. Functions
5. Object Oriented Programming

Variables

Variables are containers for storing data values.

```
x = 5

y = "John"

print(x)

print(y)
```

- 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

Variables

```
#Legal variable names:

myvar = "John"

my_var = "John"

myVar = "John"

MYVAR = "John"

myvar2 = "John"
```

```
#Illegal variable names:

2myvar = "John"

my-var = "John"

my var = "John"
```

Variables

```
Example #1

x = "awesome"

print("Python is " + x)

Example #3

x = "Python is "

y = "awesome"

z =  x + y

print(z)
```

Variables

```
Example #3

x = 5

y = 10

print(x + y)

Example #4

x = 5

y = "John"

print(x + y)
```

Variables

Example #5

```
x = "awesome"

def myfunc():

  print("Python is " + x)

myfunc()
```

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

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

Data Types

```
x = "Hello World"   -----> str

x = 20   ------> int

x = 20.5 -----> float

x = 1j -------> complex

x = ["apple", "banana", "cherry"]-------> list

x = range(6)--------> range

x = True ---------> bool
```

Operators

Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus	%
Exponentiation	**
Floor division	//

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

Comparison Operators

Operator	Name	Example
==	Equal	X == 5
!=	Not Equal	X != 3
>	greater	X >3
<	Less than	X < 3
>=	Greater or equal	X >= 3

https://www.w3schools.com/python/python_operators.asp

Predict Output

#1

X = 5

Print(x)

2

x = 5

X = x + 5

Print(x)

#3
X = 2
X = x + 2
print(x)
print(x *2)

#4
X = 2
 Y = x + 5
Z = x + y
print(z)

Predict Output

#5

X = “hello”

Y = x

X = x + “world”

Print(y)

6

x = input("Enter your name:")

print("Hello, " + x)

#3
X = 2
X = x + 2
print(x)
print(x *2)

Conditional execution

Boolean expressions

A *boolean expression* is an expression that is either true or false. The following examples use the operator `==`, which compares two operands and produces `True` if they are equal and `False` otherwise:

```
>>> 5 == 5

True

>>> 5 == 6

False
```

Conditional execution

Logical operators

There are three *logical operators*: `and`, `or`, and `not`. The semantics (meaning) of these operators is similar to their meaning in English. For example,

`x > 0 and x < 10`

is true only if `x` is greater than 0 *and* less than 10.

`n%2 == 0 or n%3 == 0`is true if *either* of the conditions is true, that is, if the number is divisible by 2 *or* 3.

Finally, the `not` operator negates a boolean expression, so `not (x > y)` is true if `x > y` is false; that is, if `x` is less than or equal to `y`.

Strictly speaking, the operands of the logical operators should be boolean expressions, but Python is not very strict. Any nonzero number is interpreted as “true.”

Conditional execution

The boolean expression after the `if` statement is called the *condition*. We end the `if` statement with a colon character (:) and the line(s) after the if statement are indented.

```
If door is closed
    If door is locked
        Unlock
    Else:
        Open and go
Else:
    go
```

Conditional execution

```
x = 3

if x < 10:

...     print('Small')

...

 Small
```

```
age = input(“what is your age: “)

If age >= 21:

    Print (“You can go to the party and drink”)

Else:

    print (“I am sorry you can’t get in”)
```

Predict output

```
x = 3

if x < 10:

   print 'less than 10'

else:

   print 'greater than 10'
```

Predict output

```python
a = 200

b = 33

if b > a:

  print("b is greater than a")

elif a == b:

  print("a and b are equal")

else:

  print("a is greater than b")
```

Predict output

```
a = 200

b = 33

c = 500

if a > b and c > a:

  print("Both conditions are True")

else:

    print False
```

Predict output

```python
x = 41

if x > 10:

  print("Above ten,")

  if x > 20:

    print("and also above 20!")

  else:

    print("but not above 20.")
```

Exercise

1. Print "Yes" if `a` is equal to `b`, otherwise print "No". a = 50, b = 10
2. Print "1" if `a` is equal to `b`, print "2" if `a` is greater than `b`, otherwise print "3".
3. This example misses indentations to be correct.Insert the missing indentation to make the code correct:

```
if 5 > 2:

print("Five is greater than two!")
```

4. Print "Yes" if a is equal to b, and c is equal to d. Otherwise ‘No’

   a = 10, b = 20 , c = 10, d = 20

5. Print "Yes" if a is equal to b, or c is equal to d. Otherwise ‘No’

   a = 10, b = 20 , c = 10, d = 20

Practice Questions

1. Which of the following are operators, and which are values?

*

'hello'

-88.8

-

/

+

5

2. Which of the following is a variable, and which is a string?

spam

'spam'

Name three data types.

4. What is an expression made up of? What do all expressions do?

5. This chapter introduced assignment statements, like `spam = 10`. What is the difference between an expression and a statement?

6. What does the variable `bacon` contain after the following code runs?

```
bacon = 20

bacon + 1
```

7. What should the following two expressions evaluate to?

```
'spam' + 'spamspam'

'spam' * 3
```

8. Why is `eggs` a valid variable name while `100` is invalid?

9. What three functions can be used to get the integer, floating-point number, or string version of a value?