Python Basics

Math Operators

From **Highest** to **Lowest** precedence:

Operators	Operation	Example
**	Exponent	`2 ** 3 = 8`
%	Modulus/Remainder	`22 % 8 = 6`
//	Integer division	`22 // 8 = 2`
/	Division	`22 / 8 = 2.75`
*	Multiplication	`3 * 3 = 9`
-	Subtraction	`5 - 2 = 3`
+	Addition	`2 + 2 = 4`

Examples of expressions:

```
>>> 2 + 3 * 6
# 20

>>> (2 + 3) * 6
# 30

>>> 2 ** 8
#256

>>> 23 // 7
# 3

>>> 23 % 7
# 2

>>> (5 - 1) * ((7 + 1) / (3 - 1))
# 16.0
```

Augmented Assignment Operators

Operator	Equivalent
`var += 1`	`var = var + 1`
`var -= 1`	`var = var - 1`
`var *= 1`	`var = var * 1`
`var /= 1`	`var = var / 1`
`var %= 1`	`var = var % 1`

Examples:

```
>>> greeting = 'Hello'
>>> greeting += ' world!'
>>> greeting
# 'Hello world!'

>>> number = 1
>>> number += 1
>>> number
# 2

>>> my_list = ['item']
>>> my_list *= 3
>>> my_list
# ['item', 'item', 'item']
```

Data Types

Data Type	Examples
Integers	`-2, -1, 0, 1, 2, 3, 4, 5`
Floating-point numbers	`-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25`
Strings	`'a', 'aa', 'aaa', 'Hello!', '11 cats'`

Concatenation and Replication

String concatenation:

```
>>> 'Alice' 'Bob'
# 'AliceBob'
```

String Replication:

```
>>> 'Alice' * 5
# 'AliceAliceAliceAliceAlice'
```

Variables

You can name a variable anything as long as it obeys the following rules:

1. It can be only one word.

```
>>> # bad
>>> my variable = 'Hello'

>>> # good
>>> var = 'Hello'
```

2. It can use only letters, numbers, and the underscore (`_`) character.

```
>>> # bad
>>> %$@variable = 'Hello'

>>> # good
>>> my_var = 'Hello'

>>> # good
>>> my_var_2 = 'Hello'
```

3. It can’t begin with a number.

```
>>> # this wont work
>>> 23_var = 'hello'
```

4. Variable name starting with an underscore (`_`) are considered as "unuseful".

```
>>> # _spam should not be used again in the code
>>> _spam = 'Hello'
```

Comments

Inline comment:

```
# This is a comment
```

Multiline comment:

```
# This is a
# multiline comment
```

Code with a comment:

```
a = 1  # initialization
```

Please note the two spaces in front of the comment.

Function docstring:

```
def foo():
    """
    This is a function docstring
    You can also use:
    ''' Function Docstring '''
    """
```

The print() Function

The `print()` function writes the value of the argument(s) it is given. [...] it handles multiple arguments, floating point-quantities, and strings. Strings are printed without quotes, and a space is inserted between items, so you can format things nicely:

```
>>> print('Hello world!')
# Hello world!

>>> a = 1
>>> print('Hello world!', a)
# Hello world! 1
```

The end keyword

The keyword argument `end` can be used to avoid the newline after the output, or end the output with a different string:

```
phrase = ['printed', 'with', 'a', 'dash', 'in', 'between']
>>> for word in phrase:
...     print(word, end='-')
...
# printed-with-a-dash-in-between-
```

The sep keyword

The keyword `sep` specify how to separate the objects, if there is more than one:

```
print('cats', 'dogs', 'mice', sep=',')
# cats,dogs,mice
```

The input() Function

This function takes the input from the user and converts it into a string:

```
>>> print('What is your name?')   # ask for their name
>>> my_name = input()
>>> print('Hi, {}'.format(my_name))
# What is your name?
# Martha
# Hi, Martha
```

`input()` can also set a default message without using `print()`:

```
>>> my_name = input('What is your name? ')  # default message
>>> print('Hi, {}'.format(my_name))
# What is your name? Martha
# Hi, Martha
```

The len() Function

Evaluates to the integer value of the number of characters in a string, list, dictionary, etc.:

```
>>> len('hello')
# 5
```

```
>>> len(['cat', 3, 'dog'])
# 3
```

Test of emptiness example:

```
>>> a = [1, 2, 3]

# bad
>>> if len(a) > 0:  # evaluates to True
...     print("the list is not empty!")
...
# the list is not empty!

# good
>>> if a: # evaluates to True
...     print("the list is not empty!")
...
# the list is not empty!
```

The str(), int(), and float() Functions

These functions allow you to change the type of variable. For example, you can transform from an `integer` or `float` to a `string`:

```
>>> str(29)
# '29'

>>> str(-3.14)
# '-3.14'
```

Or from a `string` to an `integer` or `float`:

```
>>> int('11')
# 11

>>> float('3.14')
# 3.14
```