**Python Basic**

**Math Operators**

| **Operators** | **Operation** | **Example** |
| --- | --- | --- |
| \*\* | Exponent | 2 \*\* 3 = 8 |
| % | Modulus/ Remainder | 22 % 8 = 6 |
| // | Integer division | 22 // 8 = 2 |
| / | Division | 22 / 8 =2.7 |
| \* | Multiplication | 3 \* 3 = 9 |
| - | Subtraction | 5 - 2 = 3 |
| + | Addition | 2 + 2 = 4 |

**Data Types**

| **Data Type** | **Examples** |
| --- | --- |
| Integers | -2, -1, 0, 1, 2, 3, |
| Floating-point numbers | -1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25 |
| Strings | 'a', 'Ironhack' |

**Variables**

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

* It can be only one word.
* It can use only letters, numbers, and the underscore (\_) character.
* It can’t begin with a number.
* Variable name starting with an underscore (\_) are considered as "unuseful`.

**Comments**

Inline comment:

# This is a 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**

*>>> print('Hello world!')*

*Hello world!*

*>>> a = 1*

*>>> print('Hello world!', a)*

*Hello world! 1*

**The input() Function**

*>>> print(‘Bootcamp:')*

*>>> bcamp = input()*

Bootcamp:

Ironhack

**The len() Function**

Evaluates to the integer value of the number of characters in a string:

*>>> len('hello')*

*5*

Note: test of emptiness of strings, lists, dictionary, etc, should **not** use len, but prefer direct boolean evaluation.

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

*>>> if a:*

*>>> print("the list is not empty!")*

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

Integer to String or Float:

*>>> str(29)*

*'29'*

*>>> print('I am {} years old.'.format(str(29)))*

*I am 29 years old.*

*>>> str(-3.14)*

*'-3.14'*

*Float to Integer:*

*>>> int(7.7)*

*7*

**Flow Control**

**Comparison Operators**

| **Operator** | **Meaning** |
| --- | --- |
| == | Equal to |
| != | Not equal to |
| < | Less than |
| > | Greater Than |
| <= | Less than or Equal to |
| >= | Greater than or Equal to |

These operators evaluate to True or False depending on the values you give them.

*>>> 42 == 42*

*True*

*>>> 40 == 42*

*False*

*>>> 'hello' == 'hello'*

*True*

*>>> 'hello' == 'Hello'*

*False*

*>>> 'dog' != 'cat'*

*True*

*>>> 42 == 42.0*

*True*

*>>> 42 == '42'*

*False*

**Boolean evaluation**

Never use == or != operator to evaluate boolean operation. Use the is or is not operators, or use implicit boolean evaluation.

NO (even if they are valid Python):

*>>> True == True*

*True*

*>>> True != False*

*True*

*YES (even if they are valid Python):*

*>>> True is True*

*True*

*>>> True is not False*

*True*

*These statements are equivalent:*

*>>> if a is True:*

*>>> pass*

*>>> if a is not False:*

*>>> pass*

*>>> if a:*

*>>> pass*

*And these as well:*

*>>> if a is False:*

*>>> pass*

*>>> if a is not True:*

*>>> pass*

*>>> if not a:*

*>>> pass*

**Boolean Operators**

There are three Boolean operators: and, or, and not.

The *and* Operator’s *Truth* Table:

| **Expression** | **Evaluates to** |
| --- | --- |
| True and True | True |
| True and False | False |
| False and True | False |
| False and False | False |

The *or* Operator’s *Truth* Table:

| **Expression** | **Evaluates to** |
| --- | --- |
| True or True | True |
| True or False | True |
| False or True | True |
| False or False | False |

The *not* Operator’s *Truth* Table:

| **Expression** | **Evaluates to** |
| --- | --- |
| not True | False |
| not False | True |

**Mixing Boolean and Comparison Operators**

*>>> (4 < 5) and (5 < 6)*

*True*

*>>> (4 < 5) and (9 < 6)*

*False*

*>>> (1 == 2) or (2 == 2)*

*True*

You can also use multiple Boolean operators in an expression, along with the comparison operators:

*>>> 2 + 2 == 4 and not 2 + 2 == 5 and 2 \* 2 == 2 + 2*

*True*