Python Cheatsheet

Basic cheatsheet for Python based on the book writted by Al Sweigart, Automate the Boring Stuff with Python under the Creative Commons license.

PDF Version

Download

Table of Content:

Python Basics

Math Operators

From **Higuest** to **Lowest** precedence:

Operators	Operation	Example
**	Exponent	2 ** 3 = 8
%	Modulus/Remaider	22 % 8 = 16

//	Integer division	22 // 8 = 2
/	Division	22 / 8 = 2.75
*	Multiplication	3 * 3 = 15
-	Subtraction	5 - 2 = 3
+	Addition	2 + 2 = 4

Examples of expressions in the interactive shell:

```
>>> 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
```

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'

String 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 three rules:

1. It can be only one word.
2. It can use only letters, numbers, and the underscore (_) character.
3. It can't begin with a number.

Example:

```
>>> spam = 'Hello'
>>> spam
'Hello'
```

Comments

Inline comment:

```
# This is a comment
```

Multiline Comment:

```
"""
This is a Multiline Comment
You can also use:
''' multiline comment '''
"""
```

The print() Function

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

The input() Function

Example Code:

```
>>> print('What is your name?')    # ask for their name
>>> myName = input()
>>> print('It is good to meet you, ' + myName)
```

Output:

```
What is your name?
Al
It is good to meet you, Al
```

The len() Function

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

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

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

Convert Between Data Types:

Integer to String or Float:

```
>>> str(29)
'29'
>>> print('I am ' + str(29) + ' years old.')
I am 29 years old.
>>> str(-3.14)
'-3.14'
```

Float to Integer:

```
>>> int(7.7)
7
>>> int(7.7) + 1
8
```

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. Let's try some operators now, starting with == and !=.

Examples:

```
>>> 42 == 42
True
>>> 40 == 42
False
>>> 'hello' == 'hello'
True
>>> 'hello' == 'Hello'
False
>>> 'dog' != 'cat'
True
>>> True == True
True
>>> True != False
True
>>> 42 == 42.0
True
>>> 42 == '42'
False
```

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
```

if Statements

```
if name == 'Alice':
    print('Hi, Alice.')
```

else Statements

```
name = 'Bob'
```

```
if name == 'Alice':
    print('Hi, Alice.')
else:
    print('Hello, stranger.')
```

elif Statements

```
name = 'Bob'
age = 5
if name == 'Alice':
    print('Hi, Alice.')
elif age < 12:
    print('You are not Alice, kiddo.')
```

```
name = 'Bob'
age = 30
if name == 'Alice':
    print('Hi, Alice.')
elif age < 12:
    print('You are not Alice, kiddo.')
else:
    print('You are neither Alice nor a little kid.')
```

while Loop Statements

```
spam = 0
while spam < 5:
    print('Hello, world.')
    spam = spam + 1
```

break Statements

If the execution reaches a break statement, it immediately exits the while loop's clause.

```
while True:                         # (1)
    print('Please type your name.')
    name = input()                  # (2)
    if name == 'your name':         # (3)
        break                       # (4)
print('Thank you!')                 # (5)
```

continue Statements

When the program execution reaches a continue statement, the program execution immediately jumps back to the start of the loop.

```
while True:
  print('Who are you?')
  name = input()
  if name != 'Joe':       #(1)
    continue              #(2)
  print('Hello, Joe. What is the password? (It is a fish.)')
  password = input()      #(3)
  if password == 'swordfish':
    break                 #(4)
print('Access granted.')  #(5)
```

for Loops and the range() Function

```
print('My name is')
for i in range(5):
    print('Jimmy Five Times (' + str(i) + ')')
```

Output:

```
My name is
Jimmy Five Times (0)
Jimmy Five Times (1)
Jimmy Five Times (2)
Jimmy Five Times (3)
Jimmy Five Times (4)
```

The *range()* function can also be called with three arguments. The first two arguments will be the start and stop values, and the third will be the step argument. The step is the amount that the variable is increased by after each iteration.

```
for i in range(0, 10, 2):
    print(i)
```

Output:

```
0
```

```
2
4
6
8
```

You can even use a negative number for the step argument to make the for loop count down instead of up.

```
for i in range(5, -1, -1):
    print(i)
```

Output:

```
5
4
3
2
1
0
```

Importing Modules

```
import random
for i in range(5):
    print(random.randint(1, 10))
```

```
import random, sys, os, math
```

```
from random import *.
```

Ending a Program Early with sys.exit()

```
import sys

while True:
    print('Type exit to exit.')
    response = input()
    if response == 'exit':
        sys.exit()
    print('You typed ' + response + '.')
```