30 Helpful Python Snippets That You Can Learn in 30 Seconds or Less

Fatos Morina

Sep 7, 2019 · 4 min read ★

Photo by Jantine Doornbos on Unsplash

Python represents one of the most popular languages that many people use it in data science and machine learning, web development, scripting, automation, etc.

Part of the reason for this popularity is its simplicity and easiness to learn it.

If you are reading this, then it is highly likely that you already use Python or at least have an interest in it.

In this article, we will briefly see 30 short code snippets that you can understand and learn in 30 seconds or less.

1. All unique

The following method checks whether the given list has duplicate elements. It uses the property of *set()* which removes duplicate elements from the list.

```
def all_unique(lst):
    return len(lst) == len(set(lst))


x = [1,1,2,2,3,2,3,4,5,6]
y = [1,2,3,4,5]
all_unique(x) # False
all_unique(y) # True
```

all_unique.py hosted with ♡ by GitHub view raw

2. Anagrams

This method can be used to check if two strings are anagrams. An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

```
from collections import Counter

def anagram(first, second):
    return Counter(first) == Counter(second)


anagram("abcd3", "3acdb") # True
```

anagram.py hosted with ♡ by GitHub view raw

3. Memory

This snippet can be used to check the memory usage of an object.

```
import sys

variable = 30
```

4. Byte size

This method returns the length of a string in bytes.

```
def byte_size(string):
    return(len(string.encode('utf-8')))


byte_size('😀') # 4
byte_size('Hello World') # 11
```

byte_size.py hosted with ♡ by GitHub view raw

5. Print a string N times

This snippet can be used to print a string *n* times without having to use loops to do it.

```
n = 2
s ="Programming"

print(s * n) # ProgrammingProgramming
```

print.py hosted with ♡ by GitHub view raw

6. Capitalize first letters

This snippet simply uses the method *title()* to capitalize first letters of every word in a string.

```
s = "programming is awesome"

print(s.title()) # Programming Is Awesome
```

capitalize_words.py hosted with ♡ by GitHub view raw

7. Chunk

This method chunks a list into smaller lists of a specified size.

```
def chunk(list, size):
    return [list[i:i+size] for i in range(0,len(list), size)]
```

8. Compact

This method removes falsy values (False, None, 0 and “”) from a list by using *filter()*.

```
def compact(lst):
    return list(filter(None, lst))


compact([0, 1, False, 2, '', 3, 'a', 's', 34]) # [ 1, 2, 3, 'a', 's', 34 ]
```

compact.py hosted with ♡ by GitHub view raw

9. Count by

This snippet can be used to transpose a 2D array.

```
array = [['a', 'b'], ['c', 'd'], ['e', 'f']]
transposed = zip(*array)
print(transposed) # [('a', 'c', 'e'), ('b', 'd', 'f')]
```

transpose.py hosted with ♡ by GitHub view raw

10. Chained comparison

You can do multiple comparisons with all kinds of operators in a single line.

```
a = 3
print( 2 < a < 8) # True
print(1 == a < 2) # False
```

comparisons.py hosted with ♡ by GitHub view raw

11. Comma-separated

This snippet can be used to turn a list of strings into a single string with each element from the list separated by commas.

```
hobbies = ["basketball", "football", "swimming"]

print("My hobbies are:") # My hobbies are:
print(", ".join(hobbies)) # basketball, football, swimming
```

comma_separated.py hosted with ♡ by GitHub view raw

12. Get vowels

This method gets vowels (‘a’, ‘e’, ‘i’, ‘o’, ‘u’) found in a string.

```python
def get_vowels(string):
    return [each for each in string if each in 'aeiou']


get_vowels('foobar') # ['o', 'o', 'a']
get_vowels('gym') # []
```

get_vowels.py hosted with ♡ by GitHub view raw

13. Decapitalize

This method can be used to turn the first letter of the given string into lowercase.

```python
def decapitalize(str):
    return str[:1].lower() + str[1:]


decapitalize('FooBar') # 'fooBar'
decapitalize('FooBar') # 'fooBar'
```

decapitalize.py hosted with ♡ by GitHub view raw

14. Flatten

The following methods flatten a potentially deep list using recursion.

```python
def spread(arg):
    ret = []
    for i in arg:
        if isinstance(i, list):
            ret.extend(i)
        else:
            ret.append(i)
    return ret

def deep_flatten(xs):
    flat_list = []
    [flat_list.extend(deep_flatten(x)) for x in xs] if isinstance(xs, list) else flat_list.appe
    return flat_list

```

flatten.py hosted with ♡ by GitHub view raw

15. Difference

This method finds the difference between two iterables by keeping only the values that are in the first one.

```python
def difference(a, b):
    set_a = set(a)
    set_b = set(b)
    comparison = set_a.difference(set_b)
    return list(comparison)


difference([1,2,3], [1,2,4]) # [3]
```

differences.py hosted with ♡ by GitHub view raw

16. Difference by

The following method returns the difference between two lists after applying a given function to each element of both lists.

```python
def difference_by(a, b, fn):
    b = set(map(fn, b))
    return [item for item in a if fn(item) not in b]


from math import floor
difference_by([2.1, 1.2], [2.3, 3.4], floor) # [1.2]
difference_by([{ 'x': 2 }, { 'x': 1 }], [{ 'x': 1 }], lambda v : v['x']) # [ { x: 2 } ]
```

difference_by.py hosted with ♡ by GitHub view raw

17. Chained function call

You can call multiple functions inside a single line.

```python
def add(a, b):
    return a + b

```

```
a, b = 4, 5
print((subtract if a > b else add)(a, b)) # 9
```

chained.py hosted with ♡ by GitHub view raw

18. Has duplicates

The following method checks whether a list has duplicate values by using the fact that *set()* contains only unique elements.

```
def has_duplicates(lst):
    return len(lst) != len(set(lst))


x = [1,2,3,4,5,5]
y = [1,2,3,4,5]
has_duplicates(x) # True
has_duplicates(y) # False
```

has_duplicates.py hosted with ♡ by GitHub view raw

19. Merge two dictionaries

The following method can be used to merge two dictionaries.

```
def merge_two_dicts(a, b):
    c = a.copy()   # make a copy of a
    c.update(b)    # modify keys and values of a with the ones from b
    return c


a = { 'x': 1, 'y': 2}
b = { 'y': 3, 'z': 4}
print(merge_two_dicts(a, b)) # {'y': 3, 'x': 1, 'z': 4}
```

merge.py hosted with ♡ by GitHub view raw

In Python 3.5 and above, you can also do it like the following:

```
def merge_dictionaries(a, b):
   return {**a, **b}


```

```
print(merge_dictionaries(a, b)) # {'y': 3, 'x': 1, 'z': 4}
```

merge_dictionaries.py hosted with ♡ by GitHub view raw

20. Convert two lists into a dictionary

The following method can be used to convert two lists into a dictionary.

```
def to_dictionary(keys, values):
    return dict(zip(keys, values))


keys = ["a", "b", "c"]
values = [2, 3, 4]
print(to_dictionary(keys, values)) # {'a': 2, 'c': 4, 'b': 3}
```

to_dictionary.py hosted with ♡ by GitHub view raw

21. Use enumerate

This snippet shows that you can use *enumerate* to get both the values and the indexes of lists.

```
list = ["a", "b", "c", "d"]
for index, element in enumerate(list):
    print("Value", element, "Index ", index, )
# ('Value', 'a', 'Index ', 0)
# ('Value', 'b', 'Index ', 1)
#('Value', 'c', 'Index ', 2)
# ('Value', 'd', 'Index ', 3)
```

enumerate.py hosted with ♡ by GitHub view raw

22. Time spent

This snippet can be used to calculate the time it takes to execute a particular code.

```
import time

start_time = time.time()

a = 1
b = 2
```

```
9
10  end_time = time.time()
11  total_time = end_time - start_time
12  print("Time: ", total_time)
13
14  # ('Time: ', 1.1205673217773438e-05)
```

time_taken.py hosted with ♡ by GitHub view raw

23. Try else

You can have an *else* clause as part of a *try/except* block, which is executed if no exception is thrown.

```
1  try:
2      2*3
3  except TypeError:
4      print("An exception was raised")
5  else:
6      print("Thank God, no exceptions were raised.")
7
8  #Thank God, no exceptions were raised.
```

try_else.py hosted with ♡ by GitHub view raw

24. Most frequent

This method returns the most frequent element that appears in a list.

```
1  def most_frequent(list):
2      return max(set(list), key = list.count)
3
4
5  numbers = [1,2,1,2,3,2,1,4,2]
6  most_frequent(numbers)
```

most_frequent.py hosted with ♡ by GitHub view raw

25. Palindrome

This method checks whether a given string is a palindrome.

```
1  def palindrome(a):
```

```
palindrome('mom') # True
```

palindrome.py hosted with ♡ by GitHub view raw

26. Calculator without if-else

The following snippet shows how you can write a simple calculator without the need to use if-else conditions.

```
import operator
action = {
    "+": operator.add,
    "-": operator.sub,
    "/": operator.truediv,
    "*": operator.mul,
    "**": pow
}
print(action['-'](50, 25)) # 25
```

operator.py hosted with ♡ by GitHub view raw

27. Shuffle

This snippet can be used to randomize the order of the elements in a list. Note that shuffle works in place, and returns *None*.

```
from random import shuffle

foo = [1, 2, 3, 4]
shuffle(foo)
print(foo) # [1, 4, 3, 2] , foo = [1, 2, 3, 4]
```

shuffle.py hosted with ♡ by GitHub view raw

28. Spread

This method flattens a list similarly like [].concat(...arr) in JavaScript.

```
def spread(arg):
    ret = []
    for i in arg:
        if isinstance(i, list):
```

```
            ret.append(i)
    return ret


spread([1,2,3,[4,5,6],[7],8,9]) # [1,2,3,4,5,6,7,8,9]
```

spread.py hosted with ♡ by GitHub view raw

29. Swap values

A really quick way for swapping two variables without having to use an additional one.

```
a, b = -1, 14
a, b = b, a

print(a) # 14
print(b) # -1
```

swap.py hosted with ♡ by GitHub view raw

30. Get default value for missing keys

This snippet shows how you can get a default value in case a key you are looking for is not included in the dictionary.

```
d = {'a': 1, 'b': 2}

print(d.get('c', 3)) # 3
```

missing_key.py hosted with ♡ by GitHub view raw

. . .

This was a short list of snippets that you may find useful in your everyday work. It was highly based on this GitHub repository in which you can find many other useful code snippets both in Python and other languages and technologies.

. . .

Thank you for taking the time to read. Please do not forget to give it a clap 👏

appreciate it.

Artificial Intelligence | Technology | Productivity | Education | Programming