Published in Level Up Coding

Abhay Parashar

Oct 11, 2021 · 11 min read ★ · Listen

30 Python Hacks Every Developer Should Know

Tricks That Made You Feel The Power of Python

"Image Created By Author Using Canva"

According to the latest survey done by StackOverflow, python is the most searched and tagged programming language in the world. It has one of the biggest active communities of developers. Python is so famous among beginners because of its simple syntax and easy-to-learn fundamentals. It is a versatile language that can be used to create almost everything in the software industry. One of the biggest advantages of python is its one-liners and packages that can do any task with few lines of code. Having so many built-in functionalities, there are some hacks that you should remember while coding in python. In this blog, I will be sharing 30 Python Hacks Every Developer Should Know.

1. Generating a File Sharing Server

Python offers a very simple way to share files from your computer to another computer or mobile by creating a free online FTP server.

```
python -m http.server 5000
```

You can choose the port range from 0 to 65353. Once the code gets executed you will see your server is running at `127.0.0.1:5000`

Now open chrome or any browser on your mobile phone and simply type

`YOUR_COMPUTER_IP_ADDRESS:PORT_NUMBER`

To get the IP address you can do `ipconfig` on the terminal of your computer. Below you, will see the IPv4 address. For example, if your IP address is `192.168.39.145` and Port Number is `5000` then the file-sharing server will be running at `192.168.39.145:5000`

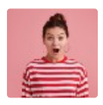
2. Passing Multiple Arguments Without Declaration

In Python, with the help of `*args` you can pass any number of arguments to a function without specifying the number.

```
def add_numbers(*numbers):
        sum = 0
        for number in numbers:
            sum += number
        return sum
print(add_numbers(5,6,233,56,3,5,2,5)) ## 315
```


Abhay Parashar
2.7K Followers

Top Writer | Engineer | Learning and Sharing Knowledge Everyday| Editor of The Pythoneers | Become a medium member bit.ly/3l3PMj4 😉



By specifying `**kwargs` you can pass any number of keyword arguments to a function.

3. Creating List Elements Smartly

A List in Python is similar to an array. It is mutable, can store heterogeneous elements, and is easy to use. Now to add elements to an array you need to run a loop and then add elements one by one. Also If there are any conditionals involved then the size of the code increases. Python provides a more efficient way of doing it combining all the steps in a one-liner called **List comprehension**.

```
'''' Creating a Simple List (The Old Way) ''''
lst = []
for i in range(10):
    lst.append(i)
print(lst)  ## 0,1,2,3,4,5,6,7,8,9


'''' Creating a List (Using List comprehension) ''''
lst = [i for i in range(10)]
print(lst)  ## 0,1,2,3,4,5,6,7,8,9


'''' Creating a List (Conditionals, List Comprehension) ''''
''' Only if '''
lst = [i**2 for i in range(1,10) if i%3==0]
print(lst)  ## [9,36,81]

''' if and else '''
lst = [i**2 if i%5==0 else i/5  for i in range(1,10)]
print(lst)  ## [0.2, 0.4, 0.6, 0.8, 25, 1.2, 1.4, 1.6, 1.8]
```

4. Type Checking 2.0

Checking the type of a variable is a task that you will perform again and again to gain knowledge. `isinstance()` function in python returns a boolean depending on whether the given object is an instance of the specified class. It takes two parameters object and the class itself. It can also be used for normal type checking.


```
class ABC:
    def __init__(self,name,age,salary):
        self.name = name
        self.age = age
        self.salary = salary

Emp1 = ABC("David",56,45000)
print(isinstance(Emp1, ABC))  ## True

data = [2,3,4,5,6]
print(isinstance(data,list)) ## True


print(isinstance(data,ABC)) ## False
```

5. Trimming Scraped Data

When we scrape some text, heading there is a lot of unwanted text (\t, \n, \t, etc.) also get scraped. Trimming is a way to getting rid of that unwanted data. There is a method in python named `strip()` that will trim all the scraped data.

```
-------------- Trimming A String -----------
data = "\n\n\n \t  David's Foord and Restaurant \t \n\n\n    "
print(data.strip())
--o/p-----
David's Foord and Restaurant


--------- Trimming List of Strings ---------
data = ["\n\n\n Burger \t   ","\n\t Pizza \t  "]
cleaned_data = [i.strip() for i in data]
print(cleaned_data)
---o/p----
["Burger","Pizza"]
```

6. The _ Operator

Single underscore `_` is a valid character in python. It can be used as a variable name. It is a special character that is used to store the result of the previous evaluation according to python docs.

```
---------- As a Variable -----------
_ = 10
```

```
b = 20
sum = _+b
print(sum)
------
30


------------ Restoring The Previous Evaluation Result ------
>>> 200+400
600
>>> _*5
3000
```

7. Shorter Names

One of the biggest features of python is its vast amount of libraries. The problem comes when you have to use them again and again in your program because some of them have bigger and non-English friendly names. Python offers a simpler way to shorten the name of the library with the help of `as` keywords.

```
## Normal Way
import numpy
import speech_recognition

## Shorten Name
import numpy as np
import speech_recognition as sr
```

8. Iterating Multiple Lists Professionally

Most of the time when you scrape data from the web, you store it in a different list. This hack lets you print each element of the list corresponding to each element of another list.

```
Authors = ['J. K. Rowling', 'Albert Einstein']
Quotes = ['"It is our choices, Harry, that show what we truly are, far more than our
abilities."','"Try not to become a man of success. Rather become a man of value."']
data = zip(Authors, Quotes)
for author, quote in list(data):
    print(quote," by ",author,end="\n")

————————————————————OUTPUT————————————————————
"It is our choices, Harry, that show what we truly are, far more than our abilities."  by  J.
K. Rowling
"Try not to become a man of success. Rather become a man of value."  by  Albert Einstein
```

9. Slicing For Advantage

Slicing is a built-in feature of python that allows you to access certain parts of a sequence. It can also be used to modify, or delete items from them. There are tons of examples where you can use slicing to reduce the size of code.

```
''' Checking For Palindrome '''
name = "wow"
print(name==name[::-1])
-----------
True


''' Retriving Even Numbers From a Natural Sequence '''
natural_numbers = [1,2,3,4,5,6,7,8,9,10]
even_numbers = natural_numbers[1::2]
print(even_numbers)
-------------
[2,4,6,8,10]
```

10. Breaking Long Lines With \

One of the biggest reasons a code becomes unreadable is because of the long file address, links, or list elements.

```
url = 'https://medium.com/pythoneers/10-underrated-python-packages-
every-data-scientist-should-know-86b4355cc35e'
```

You can change the point of wrap with the help of a backslash `\`

```
url = 'https://medium.com/pythoneers/'\
       '10-underrated-python-packages-every-'\
       'data-scientist-should-know-86b4355cc35e'
print(url)
```

```
----------------
https://medium.com/pythoneers/10-underrated-python-packages-every-
data-scientist-should-know-86b4355cc35e
```

> *Become a Genuine Medium Member With The Cost of One Pizza. It's Just* ***5$*** *a month. You Can Use* **My Referral Link** *To Become One.* **"Don't Just Read, Support The Writers Too"**

11. Count Occurrence of Elements

It is good to have an idea of the number of times elements occurred in your data structure.

```
from collections import Counter
data= [96,95,96,87,87,88,56,57,57]

occurences = Counter(data)
print(occurences)
---------------
Counter({96: 2, 87: 2, 57: 2, 95: 1, 88: 1, 56: 1})
```

12. Taking Multiple Inputs (The Pythonic Way 😎)

Python provides a simpler way to take multiple inputs using one line of code. You can also define multiple variables in one line.

```
x, y = input().split()

## Interger Inputs
x, y = map(int, input().split())

## Space Seperated Inputs
data = list(map(int, input().split()))
```

13. One Liner Functions

Python offers a one-liner function that is called the lambda function. It is also referred to as "Anonymous Function". The reason is that it doesn't require `def` keywords for definition. It can take any number of arguments but only one expression at a time. A good practice is to use it as an expression rather than binding it to a variable.

```
''' Identifying Even Numbers Using Lambda Function In a List '''

data = [2,3,7,4,8,10]
is_even = list(map(lambda x:x%2==0 , data))
print(is_even)

————————OUTPUT————————
[True, False, False, True, True, True]
```

14. Applying a Function To Each Element of the list

Use the `map` function to apply the same transformation to every element in a List.

```
ID = ["R72345","R72345&"]
results = list(map(str.isalnum, ID))
print(results)
-----------------
[True, False]
```

15. Calculate Execution Time of a Program

It is a major task in Machine learning. It is important to know the time taken by your code or function to run so that you can improve it later with some tricks.

```
''' The Simplest Way '''
import time
start_time = time.time()
...
func()
```

```
...
end_time = time.time()
print("Execution Time: ",(end_time-start_time))
```

Check Out This StackOver Flow Discussion For More Examples On How To Calculate Execution Time.

16. Use Get Method Over Square Brackets

Most Python developers are in a habit of using square brackets when accessing an element from a data structure like Dictionaries. There is no issue with square brackets but when it comes to a value that doesn't exit it shows an ugly error. Now, to save yourself from that error you can use `get` method.

```
data_dict = {a:1, b:2,c:3}

print(data_dict['d']) ## KeyError: 'd'

print(data_dict.get('d')) ## None
```

17. Handling Exceptions At Runtime

An Exception is a condition that occurs during the execution of the program due to some reason and interrupts the execution. To handle exceptions in python we use `try` and `except` block. `try` block contains the code that needs to be executed and `except` block contains the code that will execute if some error occurred in the try block.

```
''' Exception '''
a = 5
b = 0
print(a/b) ## ZeroDivisionError: division by zero

''' Exception Handling '''
try:
    a = 5
    b = 0
    print(a/b)
except:
    print("Can't Divide Number With 0")
----------------------
Can't Divide Number With 0
```

18. Converting a Normal Method onto a Static One

A Static Method is a type of method in python classes that is bound to a particular state of the class. They can not access or update the state of the class. You can convert a normal method or instance method to a static method with the help of `staticmethod(function)`

```
class ABC:
    def abc(num1,num2):
        print(num1+numn2)


# changing torture_students into static method
ABC.abc = staticmethod(ABC.abc)

ABC.abc(4,5) ## 9
```

To know the reason why you should convert a normal or instance method into a static one check out this discussion.

19. Printing Readable Results

The normal `print` statement works fine in most situations but when the output is in the form of tables, JSON, or contains multiple results then you have to add some functionality to it or use some external package.

20. Division Version 2.0

The `divmod()` is a kind of inbuilt function in python that takes two numbers as input and returns the remainder and quotient both in a tuple.

```
a = 2560
b = 27

result = divmod(a,b)
print(result)
--------------------
(94, 22)
```

21. Solving Expressions In One Line

Python provides a very useful function `eval()` for solving expressions. It takes three parameters as input — the mathematical expression to be evaluated, reference to a variable, direct value reference.

```
''' Basic Evaluation '''
print(eval('2+5-3*34//2%6'))                    ## 4


''' Reference To a Variable '''
num = 6
print(eval('2+5-3*34//2%x', {'x': num}))        ## 4


''' Direct Value Reference '''
print(eval('2+5-3*34//2%x', {}, {'x': 6}))      ## 4
```

22. For Else Method

You can use else keyword inside a `for` loop. It will specify a block of code that will run after the successful execution of your `for` loop. This block can be used to specify any end condition or message for the loop.

```
for x in range(9):
  print(x)
else:
  print("Finally finished!")
```


23. Casting a Mutable To Immutable

Type casting is a feature of Python that lets you convert one data structure into another. With the help of it, you can also change a mutable into an immutable data structure.

```
''' Mutable List '''
lst = [1,2,3,4,5]
lst[0] = 6
print(lst)
------------
[6,2,3,4,5]

''' Converting It to Immutable '''
lst = [1,2,3,4,5]
lst2 = tuple(lst)
lst[0] = 6
```

```
--------------
TypeError: 'tuple' object does not support item assignment
```

24. Generating Sequence As Per Requiment

Generator in python is a type of function that returns an object that can be iterated over. In simple words generator lets to generate a sequence when required. They are very memory efficient. Generators allow you to create iterators and perform lazy evaluations.

```
def fibo(limit):
    a,b = 0,1
    while a<limit:
       yield a
       a, b = b, a+b
series = fibo(10)

print(next(series)) ## 0
print(next(series)) ## 1
print(next(series)) ## 1
```

25. Logging Instead of Print For Debugging

Logging is the process of capturing the flow of code when it executes. It is very helpful in debugging the code easily. One of the major advantages of logging over print is that even after the application is closed the logs are saved in a file to review later, which comes with log messages and other records like line number and module name. Print only save and show the data until the application is alive and running. Python provides a module name `logging` to generate and write logs.

26. Adding New Functionalities Smartly

Decorator is a feature of Python that lets you add new functionality to an existing code without explicitly modifying it. A great application of decorator is adding average functionality to a function that calculates addition and percentage without modifying the function.

27. Use Context Managers For Resource Handling

Context Manager is a great tool in python that lets you allocate and release resources when you want. The most used and recognized example of context manager is `with` statement. `with` is mostly used to open and close a file. One of the biggest advantages of using with is that it makes sure the files get closed after use.

```
with open ('content.txt','w') as f:
    f.write("Hello Python")
```

28. PyForest (Lazy Developers Only)

This Hack is literally one of my favorite and I use it in every project. Most of the time it happened you spend a lot of time importing the basic libraries Like Numpy, Pandas, Etc. To save time and remove the headache `pyforest` is a library for you. It imports all the necessary libraries automatically that are required for a machine learning project.

```
from pyforest import *
```

29. Itetools

This module implements a number of iterator building blocks inspired by constructs from APL, Haskell, and SML. It provides many amazing functions that make this library a gem of python. Check the Documentation for different functions available in this library.

```
''' Convert a list of list into a single list  '''
import itertools
data = [[1, 2], [3, 4], [5, 6],[7,8,9, 10] ]
lst = list(itertools.chain.from_iterable(data))
print(lst)   ## [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

30. Collections In Python

Collection in python are containers that are used to store the collection of data. Python provides a package `collection` that contains different types of useful containers that can be used for different purposes.

For Example:

Counter — Takes an iterable and returns a dictionary where Keys = elements and Value = their count in the iterable.

namedtuple — Returns a tuple with names for each position in the tuple.

Ordered Dict — Type of dictionary where the order is maintained at any cost.

Default Dict — Contains default values for each key if not assigned.

```
from collections import Counter
data = [1,2,3,4,5,5,2,3]
count = Counter(data)
print(count)    ## Counter({2: 2, 3: 2, 5: 2, 1: 1, 4: 1})
```

“Liked What You Read, Become a Medium Member To Read Unlimited Content Like This”

Join Medium with my referral link - Abhay Parashar

As a Medium member, a portion of your membership fee goes to writers you read, and you get full access to every story...

abhayparashar31.medium.com

Some HandPicked Articles For You To Read Next

Are You a Newbie In Python, or an Old Developer Looking To Refresh his skills Check Out This Blog To Get a Good Understanding and Refreshment of Basic Concepts of Python.

The Ultimate Python Tutorial For Beginners

Covering All The Basic Topics, From Variables To Classes & Objects

medium.com

Not A Newbie Then Check Out the Below Article To Learn Some Advanced Concepts of Python.

Concepts that help to increase your Python knowledge

levelup.gitconnected.com

Not Interested In Learning Looking For Some Projects To Finish This Weekend Check Out The Below Article.

6 Cool Python Projects That You Can Finish This Weekend

Projects To Enhance Your Ability to Think and Code

medium.com