The Treasures of Python’s built in Libraries

towardsdatascience.com/the-treasures-of-pythons-built-in-libraries-605deb73daf6

George Seif

November 2, 2018

Discover the treasures hidden within Python!

Python is a beautiful language. Simple to use yet powerfully expressive. But are you using everything that it has to offer? Every well experienced developer knows that knowing the hidden treasures of their programming language of choice helps them get around many common bugs and everyday coding hassles. Here are some of those treasures that come straight from Python’s built in libraries!

(1) Convenient os functions

Makes interaction with your file-system *so* much easier.

```
import os

### Execute a shell command
os.system("echo 'My name is bob the builder'")

### Return the current working directory
os.getcwd()

### List all of the files and sub-directories in a particular folder
os.listdir("Documents")

### Create a single folder
os.mkdir("Data Science Projects")
```

```
### Create folders recursively
### The below line creates a folder "Documents" with a subfolder
### inside it called "Data Science Projects" with a subfolder inside ### that one
called "Project 1"
os.makedirs("Documents/Data Science Projects/Project 1")

### Delete a file
os.remove("data.txt")

### Delete a folder
os.rmdir("Data Science Projects")

### Delete directories recursively.
os.removedirs("Documents/Data Science Projects/Project 1")

### Rename a file or folder
os.rename("My Documents", "Your Documents")
```

(2) Easy work with file paths

No more messing around with building file paths

```
import os

### Handling slashes / when creating file paths
file = "process.py"
folder = "Documents/project1"
full_path = os.path.join(folder, file)

### Get the directory and file name from a full path
file = os.path.basename(full_path)
folder = os.path.dirname(full_path)

### Check if a file or folder exists
os.path.exists(full_path)

### Get the extension of a file
name, extension = os.path.splitext(file)
```

(3) Advanced unpacking from lists

You can already do this:

```
>>> a, b = range(2)
>>> a
0
>>> b
1
```

But did you know you can get even *more* clever with python:

```
>>> a, b, *c = range(8)
>>> a
0
>>> b
1
>>> c
[2, 3, 4, 5, 6, 7]
```

You can actually do this anywhere within the list:

```
>>> a, *b, c = range(8)
>>> a
0
>>> b
[1, 2, 3, 4, 5, 6]
>>> c
7
```

(4) Working with time

No more hard coding dates and times. And measure the run time of each of your major functions

```
import time
import datetime
import calendar

### Get the current date and time
>>> datetime.datetime.now()
2018-11-01 20:17:12.964253

### Get just the current time
>>> datetime.datetime.now().time()
2018-11-01 20:19:16.819745

### Freeze the program for a set period of time
>>> time.sleep(secs=5)

### Measure runtime of a python command
>>> start = time.time()
>>> 100 / 5
>>> end = time.time()
>>> print(end - start)
0.000005346
```

(5) Functions on lists

Making things easier when working with lists

```
### Sorting a list form low to high
list_1 = [7, 26, 34, 2, 12, 98, 56]
list_2 = sorted(list_1)

### Getting the max, min, and sum of a list
list_sum = sum(list_1) #
max_val = max(list_1)
min_val = min(list_1)

### Convert a string to a list where each element is a character
>>> list("bob")
["b", "o", "b"]

### You can quickly reverse a list
>>> list_1.reverse()
[56, 98, 12, 2, 34, 26, 7]
```

```
### Are any or all of the items in our list True?
>>> bool_list = [True, False, True, True]
>>> any(bool_list)
True
>>> all(bool_list)
False

### You can loop through a list's values AND indices simultaneously
for index, value in enumerate(list_1):
    "do stuff"
```

(6) Playing smart with strings

We look these up a million times on stackoverflow, so let's just list them here!

```
### Check if a string contains a substring
sentence = "Hi I'm Bob!"
if "Bob" in sentence:
    print("YES")

### Let's take care of those capital letters too, just in case the ### string for
"bOb" looks a little bit different ...
if "bOb".lower() in sentence.lower():
    print("YES")

### These two will convert a string to lower and upper case letters
sentence.lower()
sentence.upper()

### Check the properties of your string
sentence.isalpha() # Alphabetic characters only (no symbols or nums)
sentence.isnumeric() # Numbers only
sentence.islower() # Is it all lower case?
sentence.isupper() # Is it all upper case?

### Clean up your string
sentence.capitalize() # First char is capitalized
sentence.lstrip() # Remove whitespace on left side
sentence.rstrip() # Remove whitespace on right side
sentence.strip() # Remove whitespace on both sides

### The .join() method can concatenate strings with a separator
### list --> string
>>> " ".join(["Bob","has","a","balloon"])
"Bob has a balloon"

### The .split() method does the opposite of .join()
### string --> list
>>> "Bob has a balloon".split(" ")
["Bob","has","a","balloon"]

### The .replace() method can replace a substring with another
>>> "Bob has a balloon".replace("has", "is")
"Bob is a balloon"
```

(7) A few lesser known Math functions

These'll come in handy when you want to do a few out of the box things with your numbers

```
import math

### Remainder
>>> math.fmod(12, 5)
2

### Getting both the faction and integer parts
>>> math.modf(96.12)
(0.1200000000, 96.0)

### Computes the Euclidean norm sqrt(x*x + y*y)
>>> hypot(2, 3)
3.6056
```

A Secret Python book

Want to learn more about the hidden treasures of Python? This book was recommended to me and has been an absolute gem for discovering Python tips and tricks ever since!

Like to learn?

Follow me on twitter where I post all about the latest and greatest AI, Technology, and Science!