Python Standard Library

Vivek K. S., Deepak G.

Information Systems Decision Sciences (ISDS)
MUMA College of Business
University of South Florida
Tampa, Florida

2017

Python Standard Library

- The Python Standard Library offers ready to import modules for most common tasks in Programming with Python.
- These modules help the programmer in plenty of ways as they offer many useful pre-built solutions to most common requirements and are kept away from the core language to avoid bloating.
- Some Useful resources.
  - Documentation of Modules - `https://docs.python.org/3/library`
  - Tutorial on their usage - `https://docs.python.org/3.3/tutorial/stdlib.html`
  - Module of the Week by Doug Hellmann - `https://pymotw.com/2/contents.html`
  - The Python Standard Library By Example Book - `https://doughellmann.com/blog/the-python-standard-library-by-example`

Counter()

Counter() is used to count the number of occurrences of a unique element in a sequence. Example,

```
from collections import Counter
list = ['Orange','Apple','Mango','Orange']
list_counter = Counter(list)
list_counter
Counter({'Apple': 1, 'Mango': 1, 'Orange': 2})
```

The most_common() function returns all elements in descending order, or just the top count elements if given a count:

```
list_counter.most_common()
[('Orange', 2), ('Mango', 1), ('Apple', 1)]
```

Deque

Deque is a special data structure that is a combination of both Stack and Queue type. A deque is a double-ended queue.

- Its useful as a structure where we can add and delete elements from both ends of the sequence.
- The function popleft() removes the leftmost item and pop() removes the rightmost element.
- A deque could be used to create a palindrome checker as follows.

Palindrome Checker Using Deque

```
def palindrome_checker(word):
    from collections import deque
    dq = deque(word)
    while len(dq) > 1:
        if dq.popleft() != dq.pop():
            return False
    return True

palindrome_checker('racecar')
True
palindrome_checker('hello')
False
```

Iterate with Itertools

Itertools contains special-purpose iteration functions that come in handy in a lot of scenarios.
chain() runs through each argument as though they are all part of one iterable:

```
import itertools
for item in itertools.chain([1, 2],
['one', 'two']):
        print(item)
```

cycle() is an infinite iterator, cycling through its arguments:

```
import itertools
for item in itertools.cycle([1, 2]):
    print(item)
```

Iterate with Itertools

accumulate() calculates accumulated values. By default, it calculates the sum:

```
import itertools
for item in itertools.accumulate([1, 2, 3, 4]):
    print(item)
```

If a function is passed as the second argument to accumulate(), it will supersede the default addition operation.

```
def multiply(a, b):
    return a * b

import itertools
for item in itertools.accumulate([1, 2, 3, 4],
multiply):
    print(item)
```

The function should take two arguments and return a single result.

Print Pretty Statements with pprint()

pprint pretty prints data for us.

```
data = [ (i, { 'a':'A','b':'B','c':'C',
'd':'D','e':'E'})
          for i in range(3)
          ]
from pprint import pprint

# Check the difference between
print(data)
pprint(data)
```

Summary

- We learned how Python pushed all the custom, advanced functionalities into its Standard library so that the language would not become too bloated.
- We learned how to use different tools such as Counter, Deque, itertools and pprint to make our code deliver more.
- We learned how to import these modules into our code as and when we need, even selectively, rather than importing the entire package.