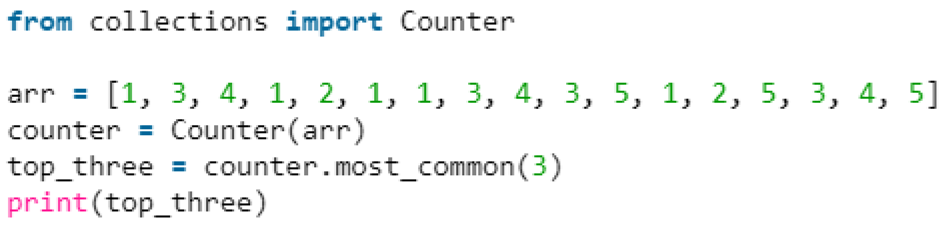
Python HackPack - by Raul

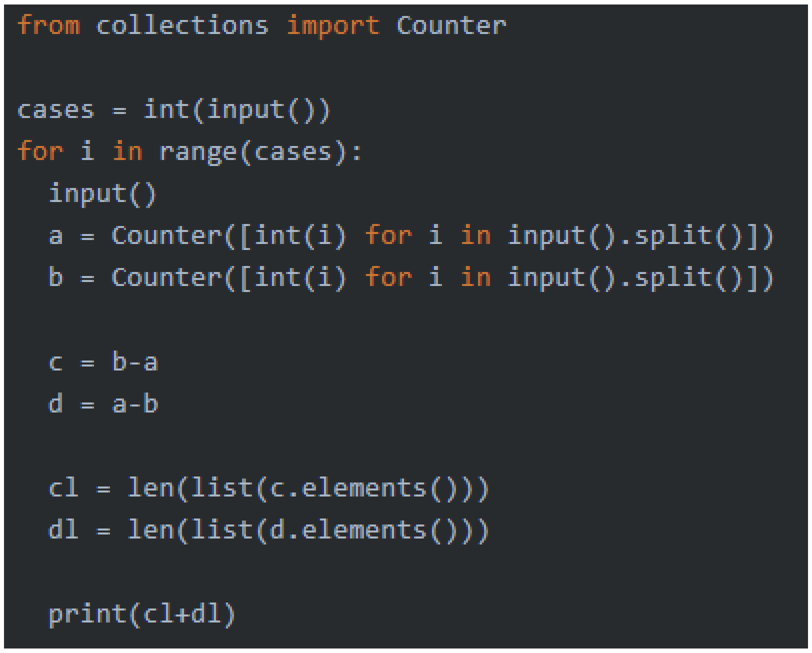
Python Lists

Counter Library



```
from collections import Counter

arr = [1, 3, 4, 1, 2, 1, 1, 3, 4, 3, 5, 1, 2, 5, 3, 4, 5]
counter = Counter(arr)
top_three = counter.most_common(3)
print(top_three)
```

This will print the top 3 most common numbers in a list.



```
from collections import Counter

cases = int(input())
for i in range(cases):
  input()
  a = Counter([int(i) for i in input().split()])
  b = Counter([int(i) for i in input().split()])

  c = b-a
  d = a-b

  cl = len(list(c.elements()))
  dl = len(list(d.elements()))

  print(cl+dl)
```

The Counter library is useful for comparing lists. In this case, I was able to “subtract” two lists.

Ex:

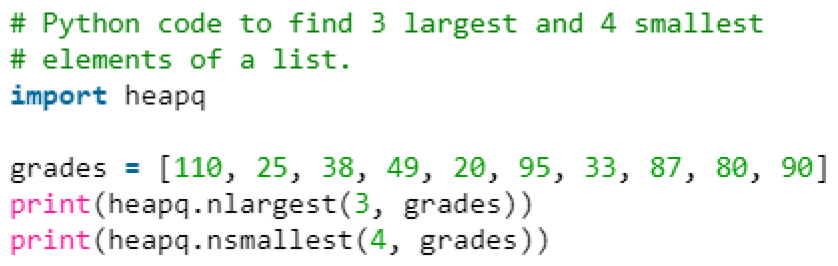
a = [1, 2, 3, 4, 5]

b = [1, 2, 3, 3, 5]

list((a-b).elements()) = [4]

list((b-a).elements()) = [3]

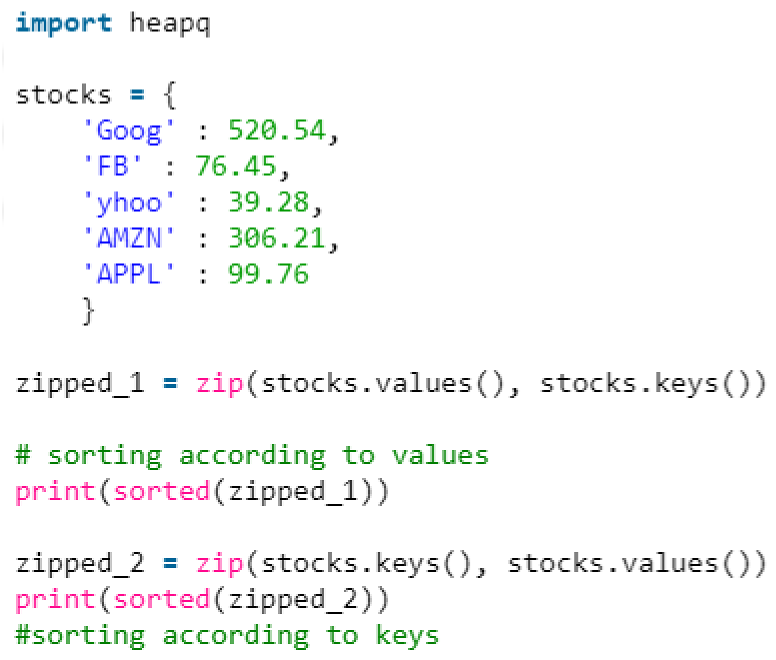
Heapq Library



```
# Python code to find 3 largest and 4 smallest
# elements of a list.
import heapq

grades = [110, 25, 38, 49, 20, 95, 33, 87, 80, 90]
print(heapq.nlargest(3, grades))
print(heapq.nsmallest(4, grades))
```

Heapq is useful for comparing array elements to eachother. These methods get the biggest

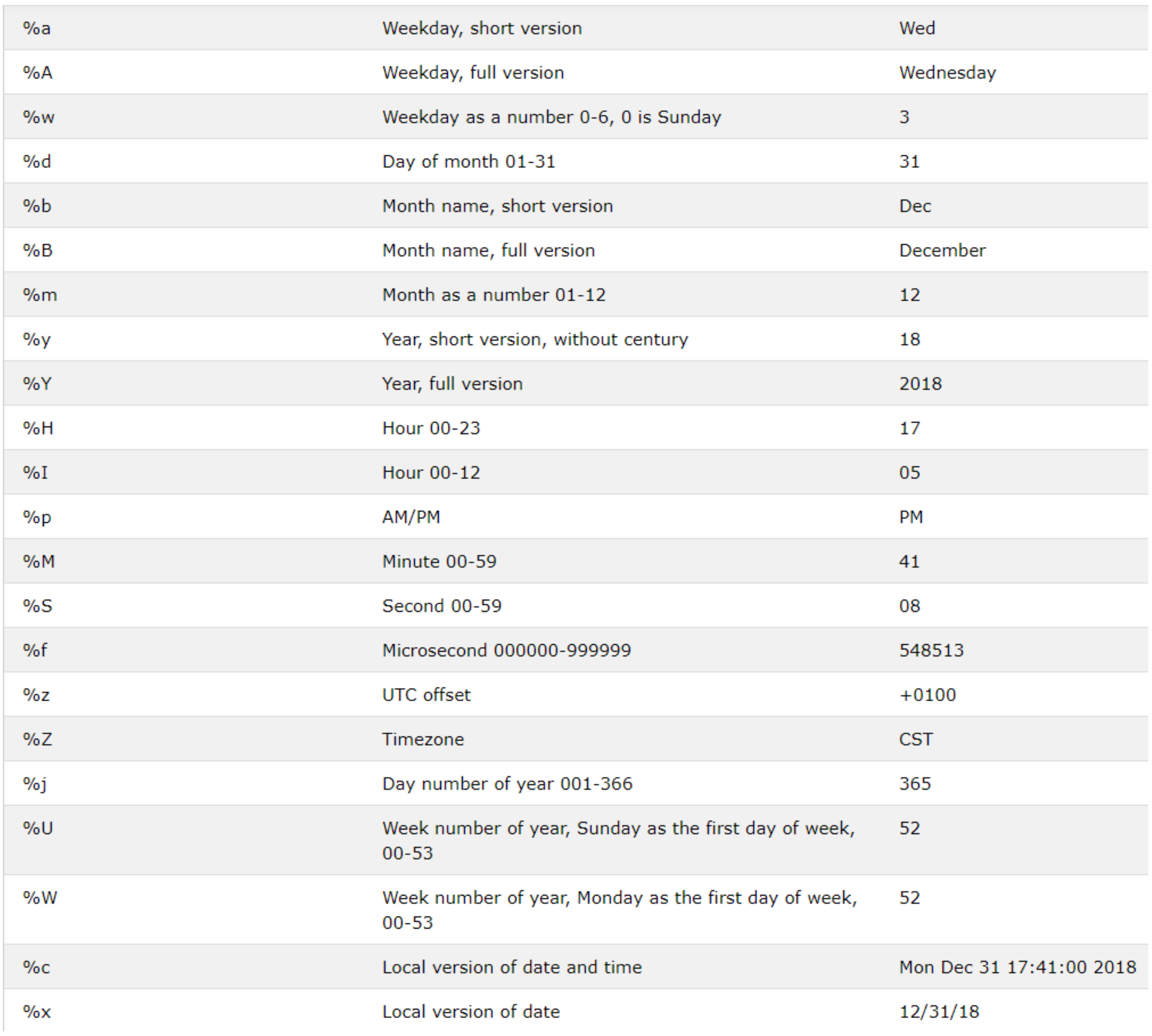
3 and smallest 4 items (respectively).



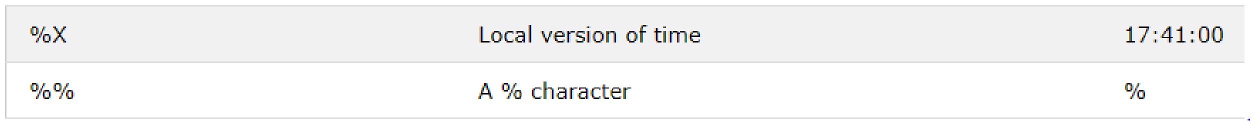
```
import heapq

stocks = {
    'Goog' : 520.54,
    'FB' : 76.45,
    'yhoo' : 39.28,
    'AMZN' : 306.21,
    'APPL' : 99.76
    }

zipped_1 = zip(stocks.values(), stocks.keys())

# sorting according to values
print(sorted(zipped_1))

zipped_2 = zip(stocks.keys(), stocks.values())
print(sorted(zipped_2))
#sorting according to keys
```

zip essentially “zips” together the keys and values into an array, and makes an array out of these arrays.

Datetime Library



%a	Weekday, short version	Wed
%A	Weekday, full version	Wednesday
%w	Weekday as a number 0-6, 0 is Sunday	3
%d	Day of month 01-31	31
%b	Month name, short version	Dec
%B	Month name, full version	December
%m	Month as a number 01-12	12
%y	Year, short version, without century	18
%Y	Year, full version	2018
%H	Hour 00-23	17
%I	Hour 00-12	05
%p	AM/PM	PM
%M	Minute 00-59	41
%S	Second 00-59	08
%f	Microsecond 000000-999999	548513
%z	UTC offset	+0100
%Z	Timezone	CST
%j	Day number of year 001-366	365
%U	Week number of year, Sunday as the first day of week, 00-53	52
%W	Week number of year, Monday as the first day of week, 00-53	52
%c	Local version of date and time	Mon Dec 31 17:41:00 2018
%x	Local version of date	12/31/18



%X	Local version of time	17:41:00
%%	A % character	%

Mapping input to output

```python
# Python code to apply a function on a list
income = [10, 30, 75]

def double_money(dollars):
    return dollars * 2

new_income = list(map(double_money, income))
print(new_income)
```

This will make an array with the input mapped to the output.

Built-In List Joining

```python
my_list = ['geeks', 'for', 'geeks']
print(''.join(my_list))
```

Nested List => Single List

```python
import itertools
a = [[1, 2], [3, 4], [5, 6]]
print(list(itertools.chain.from_iterable(a)))
```

Permutations/Combinations

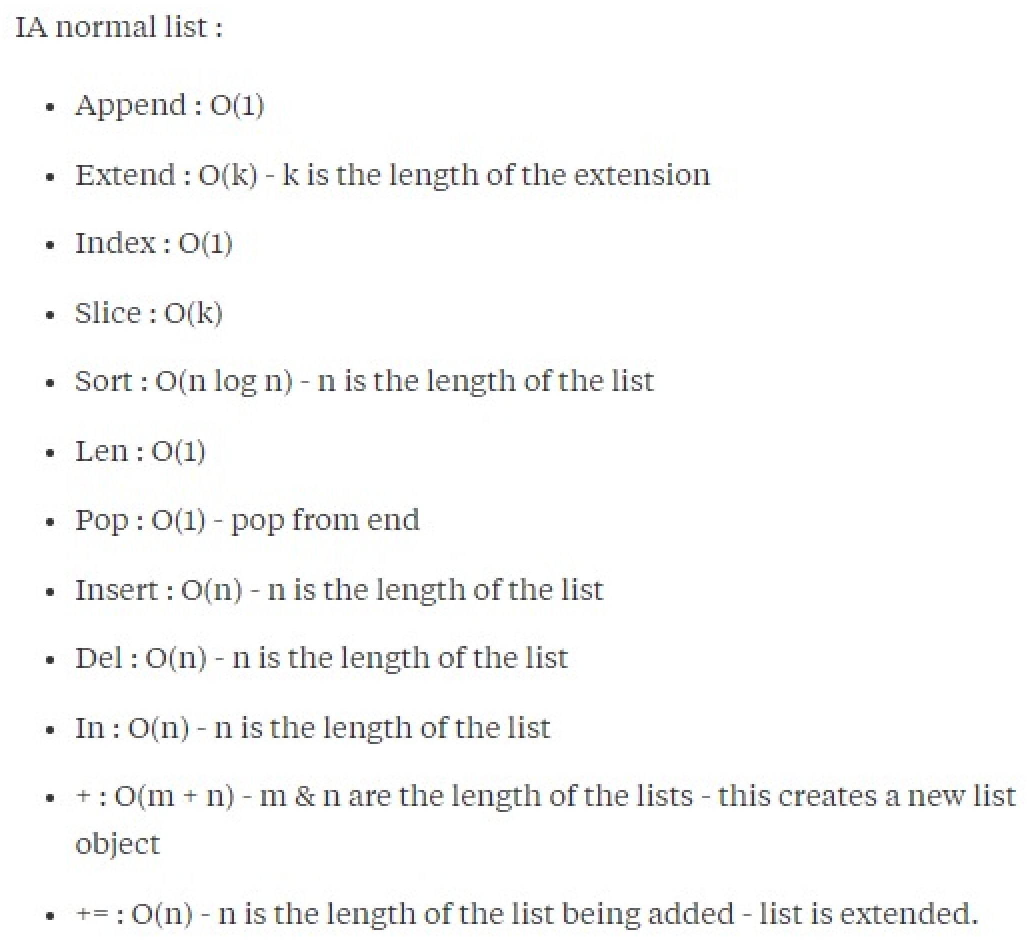
```python
from itertools import permutations
perm = permutations([1, 2, 3], 2)
for i in list(perm):
    print i

# Answer->(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)
```

Pairing Same Indexed Elements

```python
matrix = [[1, 2, 3], [4, 5, 6]]
print(zip(*matrix))
```

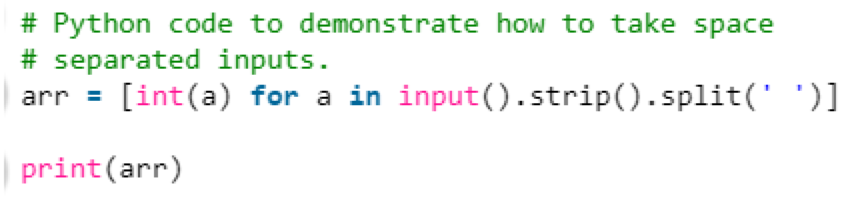
Ex: [(1,4), (2,5), (3,6)]

Standard List functions


IA normal list :

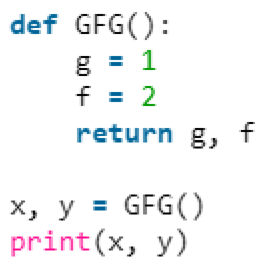
- Append : O(1)
- Extend : O(k) - k is the length of the extension
- Index : O(1)
- Slice : O(k)
- Sort : O(n log n) - n is the length of the list
- Len : O(1)
- Pop : O(1) - pop from end
- Insert : O(n) - n is the length of the list
- Del : O(n) - n is the length of the list
- In : O(n) - n is the length of the list
- + : O(m + n) - m & n are the length of the lists - this creates a new list object
- += : O(n) - n is the length of the list being added - list is extended.

Convenient Input Methods


```python
# Python code to demonstrate how to take space
# separated inputs.
arr = [int(a) for a in input().strip().split(' ')]

print(arr)
```

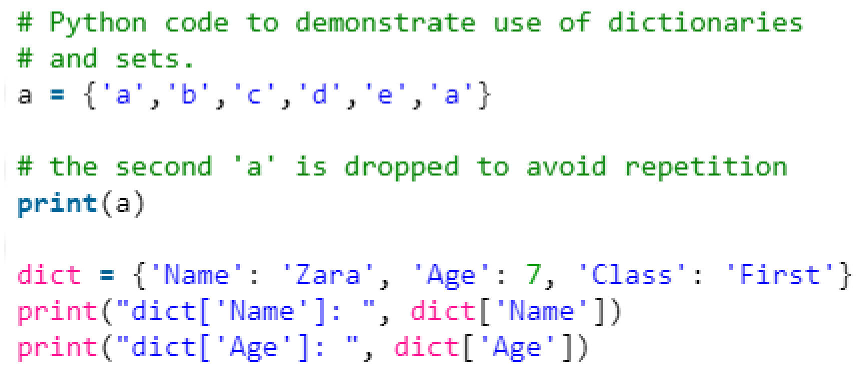
You can cast items as you iterate through a list.

Multiple Outputs



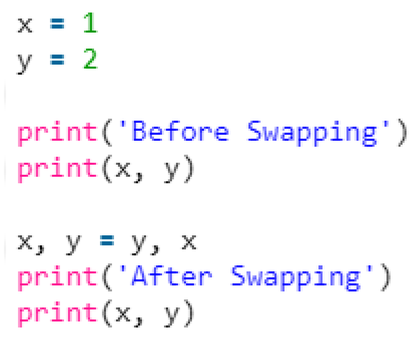
```python
def GFG():
    g = 1
    f = 2
    return g, f

x, y = GFG()
print(x, y)
```

Sets and Dictionaries



```python
# Python code to demonstrate use of dictionaries
# and sets.
a = {'a','b','c','d','e','a'}

# the second 'a' is dropped to avoid repetition
print(a)

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print("dict['Name']: ", dict['Name'])
print("dict['Age']: ", dict['Age'])
```

Convenient Swapping



```python
x = 1
y = 2

print('Before Swapping')
print(x, y)

x, y = y, x
print('After Swapping')
print(x, y)
```