Sorting – sorted()

- The syntax of sorted() method is:

  `sorted(iterable[, key][, reverse])`

Parameters
- `sorted()` takes two three parameters:
- **iterable** - sequence (`string`, `tuple`, `list`) or collection (`set`, `dictionary`, `frozen set`) or any iterator
- **reverse (Optional)** - If true, the sorted list is reversed (or sorted in Descending order)
- **key (Optional)** - function that serves as a key for the sort comparison
Example:

```python
pySet = {'e', 'a', 'u', 'o', 'i'}
print(sorted(pySet, reverse=True))

# dictionary
pyDict = {'e': 1, 'a': 2, 'u': 3, 'o': 4, 'i': 5}
print(sorted(pyDict, reverse=True))

# frozen set
pyFSet = frozenset(('e', 'a', 'u', 'o', 'i'))
print(sorted(pyFSet, reverse=True))
```

Output:

```
['o', 'u', 'i', 'e', 'a']
['o', 'u', 'i', 'e', 'a']
['o', 'u', 'i', 'e', 'a']
```

Example:

```python
def takeSecond(elem):
    return elem[1]

random = [(2, 2), (3, 4), (4, 1), (1, 3)]
sortedList = sorted(random, key=takeSecond)

print('Sorted list:', sortedList)
```

Output:

```
Sorted list: [(4, 1), (2, 2), (1, 3), (3, 4)]
```
from operator import itemgetter

lis = [{ "name": "Nandini", "age": 20 },
{ "name": "Manjeet", "age": 20 },
{ "name": "Nikhil", "age": 19 }]

print ("The list printed sorting by age: ")
print (sorted(lis, key=itemgetter('age')))

print ("The list printed sorting by age and name: ")
print (sorted(lis, key=itemgetter('age', 'name')))

print ("The list printed sorting by age in descending order: ")
print (sorted(lis, key=itemgetter('age'), reverse = True))

Output:
The list printed sorting by age:
[{'name': 'Nikhil', 'age': 19}, {'name': 'Nandini', 'age': 20}, {'name': 'Manjeet', 'age': 20}]

The list printed sorting by age and name:
[{'name': 'Nikhil', 'age': 19}, {'name': 'Manjeet', 'age': 20}, {'name': 'Nandini', 'age': 20}]

The list printed sorting by age in descending order:
[{'name': 'Nandini', 'age': 20}, {'name': 'Manjeet', 'age': 20}, {'name': 'Nikhil', 'age': 19}]
Python Comprehensions

- Python comprehensions are syntactic constructs that enable sequences to be built from other sequences in a clear and concise manner. Python comprehensions are of three types namely:
  - list comprehensions,
  - set comprehensions and
  - dict comprehensions.
Comprehensions

Example:

```
[x*x for x in lst if x % 2 == 0]
```

- `x*x`: The result
- `for x in lst`: The original list being iterated
- `if x % 2 == 0`: Conditional statement for `x`
List Comprehensions

• List comprehensions provide a concise way to create a new list of elements that satisfies a given condition from an iterable. An iterable is any python construct that can be looped over.

Example: for loop

```python
squares = []
for x in range(10):
    squares.append(x**2)
print(squares)
```

Output:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

list comprehension

```python
squares = [x**2 for x in range(10)]
print(squares)
```

Output:

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```
Example:

```python
even_squares = [i**2 for i in range(10) if i % 2 == 0]
print(even_squares)
```

Output:

```
[0, 4, 16, 36, 64]
```

Example:

```python
S = [x**2 for x in range(10)]
V = [2**i for i in range(13)]
M = [x for x in S if x % 2 == 0]

print(S)
print(V)
print(M)
```

Output:

```
S: [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
V: [1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096]
M: [0, 4, 16, 36, 64]
```
Nested *for* loops in List Comprehensions

- List comprehensions can also be used with multiple or nested *for* loops.

**Example: nested for loops**

```python
combs = []
for x in [1, 2, 3]:
    for y in [3, 1, 4]:
        if x != y:
            combs.append((x, y))
print(combs)
```

**list comprehension**

```python
combs1 = [(x, y) for x in [1, 2, 3] for y in [3, 1, 4] if x != y]
print(combs1)
```

**Output:**

- `[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]`

- `[(1, 3), (1, 4), (2, 3), (2, 1), (2, 4), (3, 1), (3, 4)]`
Set Comprehensions

• In set comprehensions, we use the braces rather than square brackets.

Example:

```python
x = {i**2 for i in range(10)}
print(type(x))
print(x)
```

Output:

```
<class 'set'>
{0, 1, 64, 4, 36, 9, 16, 49, 81, 25}
```
Dict Comprehensions

Example:

```python
x = {i:i**2 for i in range(10)}
print(type(x))
print(x)
```

Output:
```
<class 'dict'>
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```
Example:

```python
noprimes = [j for i in range(2, 8) for j in range(i*2, 50, i)]
primes = [x for x in range(2, 50) if x not in noprimes]

print(noprimes)
print(primes)
```

Output:

Noprimes: [4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42, 44, 46, 48, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 8, 12, 16, 20, 24, 28, 32, 36, 40, 44, 48, 10, 15, 20, 25, 30, 35, 40, 45, 12, 18, 24, 30, 36, 42, 48, 14, 21, 28, 35, 42, 49]
Primes: [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
Example:

```python
words = 'The quick brown fox jumps over the lazy dog'.split()
print(words)
stuff = [[[w.upper(), w.lower(), len(w)] for w in words]
for i in stuff:
    print(i)
```

Output:

```
['The', 'quick', 'brown', 'fox', 'jumps', 'over', 'the', 'lazy', 'dog']
['THE', 'the', 3]
['QUICK', 'quick', 5]
['BROWN', 'brown', 5]
['FOX', 'fox', 3]
['JUMPS', 'jumps', 5]
['OVER', 'over', 4]
['THE', 'the', 3]
['LAZY', 'lazy', 4]
['DOG', 'dog', 3]
```
Example:

```python
def zip(lst1, lst2):
    
    Made an assumption both lst1 and lst2 will have the same length.
    Used the range function to get the position the item so that we can use the position
    as the index key for both list.
    
    return [(lst1[i], lst2[i]) for i in range(len(lst1))]

print(zip([1, 2, 3], ["a", "b", "c"]))
```

Output:
```
[(1, 'a'), (2, 'b'), (3, 'c')]
```

Example:

```python
non_flat = [[1,2,3], [4,5,6], [7,8]]
list=list=[y for x in non_flat for y in x]
print(list)
```

Output:
```
[1, 2, 3, 4, 5, 6, 7, 8]
```
Example:

```python
def map(func, lst):
    
    This was pretty simple following the basic formula.
    Since we can pass functions around as an argument, the map function
    receives the function to be applied. The function is then applied to
    each item in the list.
    
    return [func(i) for i in lst]

def square(x):
    return x * x

assert map(square, range(5)) == [0, 1, 4, 9, 16]
```
2D Data Plotting in Python:

- **matplotlib** is a Python 2D plotting library

- You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc.

- Installing matplotlib: [http://matplotlib.org/users/installing.html](http://matplotlib.org/users/installing.html)


- Or use Anaconda that provides numerous built-in Python packages including matplotlib: [https://www.continuum.io/downloads](https://www.continuum.io/downloads)
Vertical Bar Chart Plotting

• Example:

```python
import matplotlib.pyplot as plot

students = ['Emre', 'Esma', 'Ahmet', 'Demet', 'Kerem']
grades = [90, 30, 45, 100, 87]
x_pos = [x for x in range(len(students))]

plot.bar(x_pos, grades, align='center', color='b', alpha=0.8)
plot.xticks(x_pos, students)
plot.ylabel('Score')
plot.title('Exam Grades')
plot.show()
```

• Output:
Horizontal Bar Chart Plotting

• Example:

```python
import matplotlib.pyplot as plt

students = ['Emre', 'Esma', 'Ahmet', 'Demet', 'Kerem']
grades = [90, 30, 45, 100, 87]
y_pos = [x for x in range(len(students))]

plt.barh(y_pos, grades, align='center', color='g', alpha=0.5)
plt.yticks(y_pos, students)
plt.xlabel('Score')
plt.title('Exam Grades')
plt.show()
```

• Output:
• **NumPy** ([http://www.numpy.org](http://www.numpy.org)) is the fundamental package for scientific computing with Python. It supports among other things:
  • a powerful N-dimensional array object,
  • sophisticated (broadcasting) functions,
  • useful linear algebra, Fourier transform, and random number capabilities,
  • efficient multi-dimensional container of generic data,
  • arbitrary data-types.


• Or use Anaconda that provides numerous built-in Python packages including NumPy: [https://www.continuum.io/downloads](https://www.continuum.io/downloads)
A simple plot with a custom dashed line

• Example:

```python
import matplotlib.pyplot as plt
import numpy as np

x = np.linspace(0, 10)

line, = plt.plot(x, np.sin(x), '--', linewidth=2, color="r")
dashes = [10, 5, 100, 5]  # 10 points on, 5 off, 100 on, 5 off
line.set_dashes(dashes)

plt.show()
```

• Output:

New function: `numpy.linspace(start, stop)`

Returns evenly spaced numbers over a specified interval `[start, stop]`. 
A simple plot of fill function

• Example:

```python
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(0, 1)
y = np.sin(4 * np.pi * x) * np.exp(-5 * x)

plt.fill(x, y, 'y')
plt.grid(True)
plt.show()
```

New functions:
- `numpy.sin()` - Trigonometric sine, element-wise
- `numpy.exp()` - Calculate the exponential of all elements in the input array
- `numpy.pi()` - π mathematical constant

• Output:
Histogram Plotting

A histogram is a graphical representation of the distribution of numerical data.

Example:

```python
import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt

mu = 100  # mean of distribution
sigma = 15  # standard deviation of distribution
x = mu + sigma * np.random.randn(10000)

num_bins = 50
# the histogram of the data
n, bins, patches = plt.hist(x, num_bins, normed=1, facecolor='green')
# add a 'best fit' line
y = mlab.normpdf(bins, mu, sigma)
plt.plot(bins, y, 'b-')
plt.xlabel('Smarts')
plt.ylabel('Probability')
plt.title(r'Histogram of IQ: $\mu$=100$, $\sigma$=15$')

# Tweak spacing to prevent clipping of ylabel
plt.subplots_adjust(left=0.15)
plt.show()
```

New function:

```python
numpy.random.randn(dimension)
```

Returns a sample (or samples) from the “standard normal” distribution.
Histogram Plotting Continued (Subplots)

• Example:

```python
import numpy as np
import matplotlib.pyplot as plt

mu = 200
sigma = 25
x = mu + sigma*np.random.randn(10000)
print(x)
fig, (ax0, ax1) = plt.subplots(nrows=2, figsize=(8, 4))

ax0.hist(x, 20, normed=1, histtype='stepfilled', facecolor='g', alpha=0.75)
ax0.set_title('Histogram type: stepfilled')

# Create a histogram by providing the bin edges (unequally spaced).
bins = [100, 150, 180, 195, 205, 220, 250, 300]
ax1.hist(x, bins, normed=1, histtype='bar', rwidth=0.7)
ax1.set_title('Histogram type: bar, unequal bins')

plt.tight_layout()
plt.show()
```

• Output:
2D Plotting and Scientific Computing in Python

• For more matplotlib examples: http://matplotlib.org/examples/index.html

• Plotting Commands Summary: http://matplotlib.org/api/pyplot_summary.html

• NumPy Manual: https://docs.scipy.org/doc/numpy/index.html