Sorting

Hamlet = "to be or not to be that is the question whether tis nobler in the mind to suffer".split()

print("hamlet:", hamlet)
print("sorted(hamlet):", sorted(hamlet))
print("hamlet:", hamlet)

hamlet.sort(): None
print("hamlet:", hamlet)

• Lists are mutable – they can be changed
  – including by functions

Customizing the Sort Order

Goal: sort a list of names by last name


print("names:", names)

This does NOT work:
print("sorted(names):", sorted(names))

When sorting, how should we compare these names?

"Niels Bohr"
"Charles Darwin"
A sort key is a different value that you use to sort a list, instead of the actual values in the list.

```python
def last_name(str):
    return str.split(' ')[1]
```

Two ways to use a sort key:
1. Create a new list containing the sort key, and then sort it
2. Pass a key function to the sorted function

```python
names = ['Isaac Newton', 'Fred Newton', 'Niels Bohr']
keyed_names = []
for name in names:
    keyed_names.append([last_name(name), name])

sorted_names = sorted(keyed_names, reverse = True)
print(sorted_names)
```

1. Use a Sort Key to Create a New List

Create a different list that contains the sort key, sort it, then extract the relevant part:

```python
names = ['Isaac Newton', 'Fred Newton', 'Niels Bohr']
keyed_names = {}
for name in names:
    keyed_names[name] = [last_name(name), name]

print(sorted(keyed_names, reverse = True))
```

2. Use a Sort Key as the Key Argument

Supply the key argument to the sorted function or the sort function.

```python
def last_name_len(name):
    return len(last_name(name))
```

```python
print(sorted(names, key = last_name_len))
```
2. Use a Sort Key as the Key Argument

Supply the `key` argument to the `sorted` function or the `sort` function

```python
def last_name(str):
    return str.split(" ")[1]

names = ["Isaac Newton", "Fred Newton", "Niels Bohr"]
print("sorted(names, key = last_name):")
for name in sorted(names, key = last_name):
    print(name)

print("sorted(names, key = last_name, reverse = True):")
for name in sorted(names, key = last_name, reverse = True):
    print(name)

def last_name_len(name):
    return len(last_name(name))

print(sorted(names, key = last_name_len))
```

### Using itemgetter

```python
from operator import itemgetter
student_score = ("Robert", 8)
itemgetter(0)(student_score) == "Robert"
itemgetter(1)(student_score) == 8

student_scores = [("Robert", 8), ("Alice", 9), ("Tina", 7)]
- Sort the list by name:
  ```python
  sorted(student_scores, key=itemgetter(0))
  ```
- Sort the list by score
  ```python
  sorted(student_scores, key=itemgetter(1))
  ```
```

### Two Ways to Import itemgetter

```python
from operator import itemgetter
student_score = ("Robert", 8)
itemgetter(0)(student_score) == "Robert"
itemgetter(1)(student_score) == 8

Or

import operator
student_score = ("Robert", 8)
operator.itemgetter(0)(student_score) == "Robert"
operator.itemgetter(1)(student_score) == 8
```
Sorting Based on Two Criteria

Two approaches:

Approach #1: Use an itemgetter with two arguments

Approach #2: Sort twice (most important sort last)

\[
\text{student\_scores} = [(\text{'Robert'}, 8), (\text{'Alice'}, 9), (\text{'Tina'}, 10), (\text{'James'}, 8)]
\]

**Goal:** sort based on score; if there is a tie within score, sort by name

Approach #1:

\[
\text{sorted\_by\_name} = \text{sorted}(\text{student\_scores}, \text{key=itemgetter}(0))
\]

Approach #2:

\[
\begin{align*}
\text{sorted\_by\_score} &= \text{sorted}(\text{sorted\_by\_name}, \text{key=itemgetter}(1)) \\
\text{sorted\_by\_hi\_score} &= \text{sorted}(\text{sorted\_by\_name}, \text{key=itemgetter}(1), \text{reverse=True})
\end{align*}
\]

Sort on Most Important Criteria LAST

- Sorted by score (ascending), when there is a tie on score, sort using name

\[
\text{from operator import itemgetter} \\
\text{student\_scores} = [(\text{'Robert'}, 8), (\text{'Alice'}, 9), (\text{'Tina'}, 10), (\text{'James'}, 8)]
\]

\[
\begin{align*}
\text{sorted\_by\_name} &= \text{sorted}(\text{student\_scores}, \text{key=itemgetter}(0)) \\
\text{sorted\_by\_score} &= \text{sorted}(\text{sorted\_by\_name}, \text{key=itemgetter}(1)) \\
\text{sorted\_by\_hi\_score} &= \text{sorted}(\text{sorted\_by\_name}, \text{key=itemgetter}(1), \text{reverse=True})
\end{align*}
\]

More Sorting Based on Two Criteria

If you want to sort different criteria in different directions, you must use multiple calls to sort or sorted

\[
\text{student\_scores} = [(\text{'Robert'}, 8), (\text{'Alice'}, 9), (\text{'Tina'}, 10), (\text{'James'}, 8)]
\]

**Goal:** sort score from highest to lowest; if there is a tie within score, sort by name alphabetically (= lowest to highest)

\[
\begin{align*}
\text{sorted\_by\_name} &= \text{sorted}(\text{student\_scores}, \text{key=itemgetter}(0)) \\
\text{sorted\_by\_hi\_score} &= \text{sorted}(\text{sorted\_by\_name}, \text{key=itemgetter}(1), \text{reverse=True})
\end{align*}
\]

Sorting: strings vs. numbers

- Sorting the powers of 5:

\[
\begin{align*}
\text{sorted([125, 5, 3125, 625, 25])] &= [5, 25, 125, 625, 3125] \\
\text{sorted(['125', '5', '3125', '625', '25'])} &= ['125', '25', '3125', '5', '625']
\end{align*}
\]
Sorting

Bubble Sort

- It repeatedly steps through the list to be sorted,
- compares each pair of adjacent items and swaps them if they are in the wrong order.
- The pass through the list is repeated until no swaps are needed, which indicates that the list is sorted.
- The algorithm, which is a comparison sort, is named for the way smaller elements "bubble" to the top of the list.

```python
def bubbleSort(alist):
    for passnum in range(len(alist)-1,0,-1):
        for i in range(passnum):
            if alist[i]>alist[i+1]:
                temp = alist[i]
                alist[i] = alist[i+1]
                alist[i+1] = temp

alist = [54,26,93,17,77,31,44,55,20]
bubbleSort(alist)
print(alist)
```
Insertion sort

**Idea:**
- maintain a sorted sublist in the lower positions of the list.
- Each new item is then “inserted” back into the previous sublist such that the sorted sublist is one item larger.

```python
def insertionSort(alist):
    for index in range(1, len(alist)):
        currentvalue = alist[index]
        position = index
        while position > 0 and alist[position - 1] > currentvalue:
            alist[position] = alist[position - 1]
            position = position - 1
        alist[position] = currentvalue
alist = [54, 26, 93, 17, 77, 31, 44, 55, 20]
insertionSort(alist)
print(alist)
```

https://www.youtube.com/watch?v=ROalU379l3U