CS 105: Introduction to Computer Science

Prof. Thao Nguyen

Spring 2025

Materials adapted from Dave Wonnacott

Python Lists, Part 2: adding items to lists

Multiple ways to add list elements:

```
bills: List[int] = [1, 10, 20, 50]
bills.append(100)  # add item to the end of the list
print(bills)


bills.insert(1,5)  # insert an item at a specified index
print(bills)


bills.extend([2,4])# append items from another list to the current
list
print(bills)
```

Python Lists, Part 3: sorting

Python lists (and strings) can be *sorted*, but with different options

```
bills: List[int] = [1, 10, 100, 50, 20, 5]   # a list, not in order
print(bills)
print(sorted(bills))
print(bills)       # did "bills" change? how to diagram "sorted"?

bills.sort()
print(bills)       # did "bills" change? how to diagram "sort"?

print(sorted('sort these letters')) # "sort" does not work for str
```

How could this help with earliest_letter?

Tuples

Immutable fixed-size varied-type sequence of items

immutable means it works readily in both S.N.M. and D.N.M.

Type information is *per-element* (unlike with lists), thus tuples are fixed-size

```
from typing import List, Tuple

T1: Tuple[str, int] = ("Thao", 6108962852)
T2: Tuple[int, int, float] = (22, 7, 22/7)
T3: Tuple[float] = (12.34, )   # comma keeps Python from treating it like (12.34)

L1: List[float] = (22, 7, 22/7)  # PyCharm would complain if we put a string here
```

We can use +, len, [ ], and "in" for tuples

Things can get complicated

We'll see lots of examples of tuples and lists.

We'll see that these are general-purpose tools that can be used together

(like combining if, while, for, etc.; then those get complicated, we make a *function* w/an abstract goal)

- we can make lists with lists or tuples in them
- we can make tuples with lists in them
- we can make lists or tuples with lists or tuples of lists or tuples in them
- etc...

In CS106, we add *objects*, letting us separate data into *abstract* components

Lists of Tuples

Lists of Tuples provide a way to handle large sets of unstructured data:

```
Faculty: List[Tuple[str, str]] = [
    ("Sara Mathieson", "L302"),
    ("Dave Wonnacott", "L307")
]
```

The type/structure there indicate that:

- we expect the same structure/number of items on each row, and
- we don't mind how many rows there are

How would we add items to this list?