Introduction to Programming

Spring 2022

Data Collection

- Simple Statistics
- Applying Lists
- **Lists of Records**
- Designing with Lists and Classes
- Case Study: Python Calculator
- Case Study: Better Cannonball Animation
- **Non-Sequential Collections**

Lists of Objects

- All of the list examples we've looked at so far have involved simple data types like numbers and strings.
- We can also use lists to store more complex data types, like our student information from chapter ten.
- Our grade processing program read through a file of student grade information and then printed out information about the student with the highest GPA.
- A common operation on data like this is to sort it, perhaps alphabetically, perhaps by credit-hours, or even by GPA.

Lists of Objects

Our grade processing program read through a file of student grade information and then printed out information about the student with the highest GPA.

A common operation on data like this is to sort it, perhaps alphabetically, perhaps by credit-hours, or even by GPA.

Lists of Objects

- Let's write a program that sorts students according to GPA using our Student class from the last chapter.

```
Get the name of the input file from the user
Read student information into a list
Sort the list by GPA
Get the name of the output file from the user
Write the student information from the list into a file
```

Lists of Objects

- Let's begin with the file processing.
- Write function to read data from a file and add them to a list
- We're using the Student class so we have to import it in our new program
- Write function to write data to an output file.
- Using the above two function we can:
  - Convert our data file into list of students
  - Write the data back into an output file.

Lists of Objects

- All we need to do now is sort the records.
- In case of numbers we can use the sort method.
  - We can use the sort method to sort our list of students.
  - Sort method knows how to sort numbers
  - How do we sort objects (Students)
  - We can sort list of students by:
- Name
- Credit Hours
- Quality Points
- GPA

Lists of Objects

- We need to provide a key to sort function:

```
<list>.sort(key=<key-function>)
```

- To sort by GPA, we need a function that takes a Student as parameter and returns the student's GPA.

```
def use_gpa(aStudent):
    return aStudent.gpa()
```

- We can now sort the data by calling sort with the key function as a keyword parameter.

```
data.sort(key=use_gpa)
```

Lists of Objects

```
data.sort(key=use_gpa)
```

- Notice that we didn't put ()'s after the function name.
- This is because we don't want to call `use_gpa`, but rather, we want to send `use_gpa` to the sort method.

Lists of Objects

- Actually, defining `use_gpa` was unnecessary.
- The `gpa` method in the `Student` class is a function that takes a student as a parameter (formally, self) and returns GPA.
- To use it:

```
data.sort(key=Student.gpa)
```

Non-sequential Collections

- After lists, a dictionary is probably the most widely used collection data type.

Dictionary Basics

- Lists allow us to store and retrieve items from sequential collections.
- When we want to access an item, we look it up by index – its position in the collection.
- What if we wanted to look students up by student id number? In programming, this is called a key-value pair
- We access the value (the student information) associated with a particular key (student id)

Dictionary Basics

- Three are lots of examples!
  - Names and phone numbers
  - Usernames and passwords
  - State names and capitals
- A collection that allows us to look up information associated with arbitrary keys is called a mapping.
- Python dictionaries are mappings. Other languages call them hashes or associative arrays.

Dictionary Basics

- Dictionaries can be created in Python by listing key-value pairs inside of curly braces.
- Keys and values are joined by ":" and are separated with commas.
- `<dictionary>[<key>]` returns the object with the associated key.
  - `passwd = {"guido":"superprogrammer", "turing":"genius", "bill":"monopoly"}`
- We use an indexing notation to do lookups

```
print(passwd["guido"])
'superprogrammer'
```

- Dictionaries are mutable.

Dictionary Basics

- Mappings are inherently unordered.
- Internally, Python stores dictionaries in a way that makes key lookup very efficient.
- When a dictionary is printed out, the order of keys will look essentially random.
- If you want to keep a collection in a certain order, you need a sequence, not a mapping!
- Keys can be any immutable type, values can be any type, including programmer-defined.

Dictionary Basics

- Like lists, Python dictionaries support a number of handy built-in operations.
- A common method for building dictionaries is to start with an empty collection and add the key-value pairs one at a time.

```
passwd = {}
file_in = open ('passwords', 'r')
for line in file_in:
    user, pass = line.split()
    passwd[user] = pass
```

Dictionary Operations

Method	Meaning
<key> in <dict>	Returns true if dictionary contains the specified key, false if it doesn't.
<dict>.keys()	Returns a sequence of keys.
<dict>.values()	Returns a sequence of values.
<dict>.items()	Returns a sequence of tuples (key, value) representing the key-value pairs.
del <dict>[<key>]	Deletes the specified entry.
<dict>.clear()	Deletes all entries.
for <var> in <dict>:	Loop over the keys.
<dict>.get(<key>, <default>)	If dictionary has key returns its value; otherwise returns default.

Dictionary Operations

```
>>> list(passwd.keys())
['guido', 'turing', 'bill']
>>> list(passwd.values())
['superprogrammer', 'genius', 'bluescreen']
>>> list(passwd.items())
[('guido', 'superprogrammer'), ('turing', 'genius'),
('bill', 'bluescreen')]
>>> "bill" in passwd
True
>>> "fred" in passwd
False
```

Dictionary Operations

```
>>> passwd.get('bill','unknown')
'bluescreen'
>>> passwd.get('fred','unknown')
'unknown'
>>> passwd.clear()
>>> passwd
{}
```

Class Work

- Download the file presidents_list.txt from Canvas.
- Write a Python program
  - To read in the file and create a dictionary (key is president's number and value is the name of the president's name
  - Inside a loop (exit the loop if use press enter key):
- Ask user for a number, use dictionary to find the name of the president and display the name