Module 11: Programming as a productivity tool

Topics:

- Design choices in Python programs
- A case study in Python

Why are there so many different ways to store data?

We have seen lists, classes, and dictionaries

- Each storage tool answers a different question
- One may be favoured over the others in certain situations
  - Speed of operations
  - Ease of programming
  - Memory requirements

When to use lists?

- Lists are a good choice when order matters –
  - Sorted order (numerical, alphabetical, etc.), or
  - Length of time in the collection (first added at beginning or end of list)

When to use dictionaries?

- Dictionaries are very powerful when the primary operations is searching via a key value
  - Easier to maintain than lists
  - Incredibly fast to search (essentially `O(1)`)
- Don't do a whole lot more
  - There is no order (and if you end up sorting the dictionary entries a lot – consider a list instead)
  - Reverse look-up is brute force

Example: Architectural History Website

- Suppose we have information about a collections of buildings, including the year that the construction began
- Task: find all buildings built in a specified year and afterwards
- Note: year is not a unique identifier for a building – multiple buildings could have been built in a single year

How should we organize the data?

- List?
  - Sorted? Unsorted?
  - How to retrieve information?
- Dictionary?
  - What would be the key?
  - What would be the associated value?
- Compare the options

Another Example: DNA Sequences

Suppose you run a genetics lab, and want to study patterns in the Y chromosome. You have a collection of Y chromosome sequences. As part of your study, you want to retrieve the symbols stored at specific locations in the sequences (e.g. at position 12,025,774) as efficiently as possible.

How should we organize the data?

- List?
  - Sorted? Unsorted?
  - How to retrieve information?
- Dictionary?
  - What would be the key?
  - What would be the associated value?
- Compare the options

Example: Voter Registry

- Voter registries have to keep track of voter registrations, where voters live and vote. They may have to be updated (e.g. changing addresses and names, or adding new voters). They should also be useful in preventing voter fraud.

How should we organize the data?

- List?
  - Sorted? Unsorted?
  - How to retrieve information?
- Dictionary?
  - What would be the key?
  - What would be the associated value?
- Compare the options

When to use classes?

- Use a new class when you have several pieces of related information and want to treat them as a single item
- If your class has only two fields, and one is a unique identifier, then consider a dictionary instead of a list of objects

Example: How should we represent houses in a neighbourhood?

```
def class House:
   'fields: house_num, residents, pets'
```

Should we use

- **`(listof House)`** or
- **`(dictof int House)`** ?

Answer: It depends …

- Look up neighbours by **`house_num`**
- Find all even-numbers houses
- Total number of pets in neighbourhood
- etc.

Design Choices

- It isn't always an easy answer
- How we use the data may change
- Consider:
  - Algorithm
  - Ease of programming
  - Efficiency of algorithm
  - Memory requirements

→ Need to be flexible and adjust as needed

Putting it all together

How can programming a computer improve your productivity and your life?

- Programming can automate tasks that are mindless but important
- A computer can do complicated calculations with more accuracy
- Your programs can solve problems much more quickly than you could by hand

Case Study: Thanking a list of charitable donors

A charity accepts on-line donations. At the end of the year, the charity would like to:

- Send one thank-you note to everyone who donated at least once
- Send one receipt to each donor, for the total amount given

Where to start?

- What does the data look like?
  - Charity maintains a data file
  - Each donation is on a separate line, containing (in order)
    - Email of donor (e.g. `generous@person.com`)
    - Date of donation (in the format day/month/year, e.g. `13/9/11`)
    - Amount given in dollars (e.g. `50` or `67.21`)
  - There may be any number of spaces between the email, data and amount (at least one)

Sample input file

```
pinesap@moergrobben.cz    1/1/11    50
youthfulness@lamusic.it   12/2/11   5.25
angel@tm-druck.at         18/2/11   50
viii@bldsci.com           18/2/11   100
youthfulness@lamusic.it   2/5/11    10
```


What should be produced?

- Another program will produce the actual thank you notes and receipts
- Your program needs to print information about each donor on a line, containing (in order)
  - Total amount given (to 2 decimal places)
  - Email address
- Write exactly one space between the email and the amount, and place a newline after the amount
- The donors do not need to be listed in any particular order.

Sample output file

```
50.00 pinesap@moergrobben.cz
15.25 youthfulness@lamusic.it
50.00 angel@tm-druck.at
100.00 viii@bldsci.com
```


More formally

Write a function **`produce_donations`** that consumes two file names: **`donors_in`** and **`donors_out`**. The function reads the donations from **`donors_in`**, and writes the distinct donors (and amount given) to **`donors_out`**, in the formats previously illustrated.

The end of CS116

Learning to program computers is an extremely challenging task, and is harder for some people than others

- Computers do not tolerate errors
- There are also lots of places for errors
- Small changes can significantly affect run-time

Knowing how to program can be profoundly powerful!

Will you use Scheme or Python ever again?

- Python is an extremely powerful tool for processing files efficiently – it might prove very useful in other contexts
- Scheme programs can be quite handy for solving mathematical problems quickly
- You have developed useful resources. The knowledge is now yours to use!

After CS 116, …

In subsequent courses (234, 230, 330, etc.), you can learn more about how to use computers effectively in other ways:

- Building databases
- Developing more complex mathematical ways to structure your data
- Managing large information systems projects
- Developing mid-scale software despite not being a computer scientist

Goals of Module 11

- Understand that multiple factors influence the best way to structure data for a specific task
  - Efficiency
  - Memory requirements
  - Simplicity
- Understand how you can dramatically improve productivity using your programming skills