

```
In [ ]: 1 from cs103 import *
        2
        3
```

CPSC 103 - Systematic Program Design

Module 07a Day 1

Rik Blok, with thanks to Jessica Wong and Giulia Toti

Reminders

- this Wed-Fri: Module 6 Tutorial Attendance
- Fri: Module 6 (One Task per Function): Worksheet
- Mon-Wed: Midterm break
- **Wed** (was this Fri): Module 6 (One Task per Function): Code Review
- **Wed** (was this Fri): Module 6 (One Task per Function): Tutorial
- next Thurs: Deadline for final exam [conflict notifications](#)
- next Thu-Fri: Module 7 Tutorial Attendance
 - **Wed tutorial section T1T may attend any [Thu-Fri section](#) next week**

[Project Milestone](#)

DUE: Friday Nov 24

Start working on it now! Watch for your [mentor](#) TA's [office hours](#) to get help.

You do not need to design `main` and `analyze` for the milestone.

Review the [grading rubric](#) carefully so that your submission fulfills our requirements!

Syzygy may crash occasionally or become very slow as the deadline approaches, due to the volume of activity. Be sure to start early.

Don't use built-in functions!

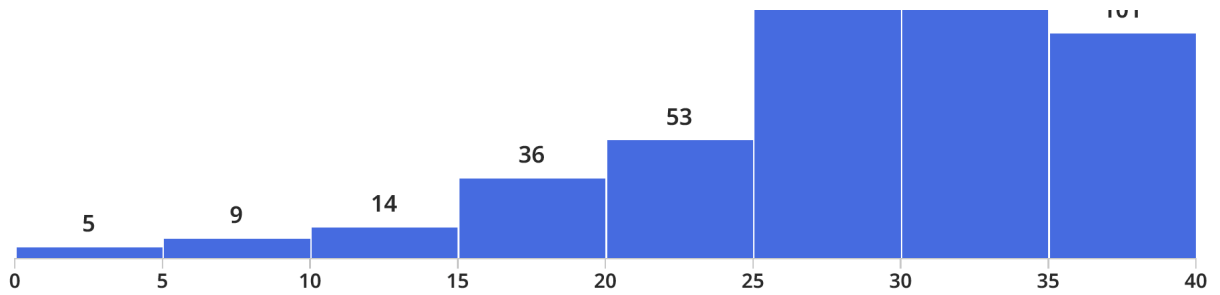
- For the project, you are **NOT** allowed to use built-in functions like `len`, `sum`, `min`, `max`
- Demonstrate how you would use good programming practices to implement the correct code yourself (e.g., helper function)

See your Canvas calendar (<https://canvas.ubc.ca/calendar>) for details.

Midterm Exam

Stay tuned for Midterm Exam grades to be released in the next day or so.

- Average = 71%, standard deviation (spread) = 19%, most scores in range [52%, 90%]
- 64 students (out of 471) scored less than 50%



- Solution will be released on the [Sample Exams](#) page
- You will be able to view your full exam on Gradescope. Follow [this link](#) to link your Canvas account. Review the feedback left by the teaching team

Regrade requests

Found an error in grading? Submit a regrade request using the Gradescope regrade request feature. Submit a separate request for each section part. For example, don't submit a regrade request for the function body (Q5.2) under Q5.1 "Stub and Template".

You must submit **no more than a week** after the exams have been released on Gradescope.

Other parts may also be regraded. **Understanding your mistakes is a valuable learning experience** so we want you to think carefully about why your answers may be wrong. Poorly justified requests may be docked flex points.

Examples:

- 🚫 "I thought the question was asking ... instead." (Unsuccessful and may be docked flex.)
- 🚫 "What I meant to say in my answer was ..." (Unsuccessful and may be docked flex.)
- 🚫 "Rubric item ... should be worth fewer points." (Unsuccessful and may be docked flex.)
- ✅ "I was docked marks for... But if you look at this part... you'll see I actually did meet the requirement..." (May be successful if well justified.)

Module learning goals

The focus moving forward is not so much to introduce new "tools" but rather to learn strategies for tackling larger programs with the tools you have.

By the end of this module, you will be able to:

- Identify the information that is available to you.
- Identify many possible outputs that your program could produce, given the information that you have available.
- Decide which subset of information you need to represent as data in your program to solve particular problems.
- Design appropriate data definitions to represent the chosen information as data in your program.
- Use the "How to Design Analysis Programs" (HtDAP) recipe to design programs that read information from a file, store it as data in your program, and analyze the data in some way.

✅ With the exception of drawing a graph (Module 8), this is essentially all you need to do for your project!

CSV files

- [CSV](#) is a simple file format used to store tabular data, such as a spreadsheet or database.
- It's just a text file that uses (C)ommas to (S)eparate (V)alues.
- Typically, the first "header" line names the columns of data and the following lines contain the data itself.

- Jupyter can open it as a text file. (Spreadsheet programs like Excel can open it, but may convert it to a spreadsheet and Jupyter will not be able to read it anymore.)
- Jupyter shows line numbers on the left (notice that a long line may wrap around on the screen, but it's still only one line).
- The file must have an empty line at the end!
- The CSV file must be in the same folder as your program.

Example

Tabular data

Year	Make	Model
1997	Ford	E350
2000	Mercury	Cougar

File contents

```
Year,Make,Model
1997,Ford,E350
2000,Mercury,Cougar
```

iClicker question: Rows versus columns

We'll sometimes refer to "rows" and "columns" of a table. Let's make sure we know which is which.

Year	Make	Model
1997	Ford	E350
2000	Mercury	Cougar

In the table to the right, let's say row 0 column 0 contains "Year". Then which row/column contains "Mercury"?

- A. Row 1 column 2
- B. Row 3 column 2
- C. Row 2 column 1
- D. Row 2 column 3
- E. None of the above

►  Hint (for your review after class)



How to Design Analysis Programs (HtDAP)

The steps in the HtDAP recipe are:

1. Planning
 - a. Plan input: Identify the information in the file your program will read.
 - b. Plan output: Write a description of what your program will produce.
 - c. Plan examples: Write or draw examples of what your program will produce.
2. Designing the program
 - a. Design data definitions.
 - b. Design read function: Design a function to read the information and store it as data in your program.
 - c. Design analyze function: Design functions to analyze the data.

Step 1a: Plan input

Identify the information in the file your program will read:

- Open the CSV file that you are going to work on and describe its information.
- What does each of the columns represent?
- What kind of values are allowed in each column? What do they mean?
- Are there missing data?
- Are there special values?
- Are there errors in the data?
- How will you treat each of those cases?

Identify all the data in the file, not only the data you are going to use in your analysis.

Step 1a: Plan input - Example: Vehicle data

The columns of the table to the right contain information about different kinds of vehicles:

Year:

- A four-digit number representing a year the vehicle was manufactured.

Make:

- The name of the company that manufactured the vehicle.
- Might be represented by a string.
- Alternatively, since there are few companies and they don't change often, might be *one of* Ford, Mercury, Subaru,...

Model:

- The name of the particular model.
 - There is some missing data (blanks).
 - There are many model names and they may change often so best represented with a string or empty string for blanks.
-

Year	Make	Model
1997	Ford	E350
2000	Mercury	Cougar
1998	Subaru	Legacy
1985	DeLorean	
2017	Subaru	Impreza
:	:	:

Example: Analysing VPD Crime Data

We've uploaded a tiny portion of the crime data shared by the [Vancouver Police Department's Open Data initiative](#). The complete file has well over half a million rows. The portion we uploaded is all crimes labelled as "break and enter" (in two variants: commercial and residential) and "theft of" (in two variants: vehicle and bicycle) in 2018.

You can find these three files in the [crimedata folder](#):

- `crimedata_subset_bne_theft_of_bike_veh_2018.csv` - the data in a CSV file
- `VPD OpenData Crime Incidents Description.pdf` - description of the data, including the columns
- `legal_disclaimer.txt` - the license for this information

We'll **start from the project final submission template** to get good practice both on using HtDAP and preparing for the project! (We've edited it slightly to note places where we'll deviate from the project.)

Step 1a: Plan input - Example

Identify the information in the file your program will read

Describe (all) the information that is available. Be sure to note any surprising or unusual features. (For example, some information sources have missing data, which may be blank or flagged using values like -99, NaN, or something else.)

Put your answer here. Please don't delete the two HTML tags on either end of this paragraph. It's to make your answer blue so the TAs can easily spot it.

▼ ⓘ Sample solution (For later. Don't peek if you want to learn 😊)

Reading the PDF file tells us that the columns of the CSV file represent:

- Type: The type of crime committed, one of
 - "break and enter - commercial",
 - "break and enter - residential",
 - "theft of vehicle", and
 - "theft of bicycle"
- Date and time when the crime was committed including year, month, day, hour and minute (all integers).
According to PDF, time has been removed (zero values) for some crimes.
- Address of the crime, including Hundred Block and Neighborhood; some entries are missing
- Coordinates of the crime in UTM coordinates (not latitude/longitude)

iClicker questions: Data types



Having identified the information, you probably already have in mind some idea of how it could be represented in a program.

Several data types are shown below (A-E).

Indicate which data type would best represent each of the following pieces of information:

1. The type of crime committed

► Next

- A. Simple Atomic
- B. Interval
- C. Optional
- D. Enumeration
- E. Compound

► ⓘ Hints (for your review after class)

Step 1b: Plan output

Write a description of what your program will produce:

- Consider the information available to you. Brainstorm some analyses that might be interesting, writing down your ideas.
- Choose the idea you want your program to produce (e.g., a forecast, statistic, visualization, ...) and briefly describe it.

Give any special information you are going to need to assume or problems you can see in your data file. (CSV files may have problems and you are not allowed to change them in any way!)



iClicker question: Program output

Would the following description be a good choice for what our program could produce from the VPD Crime Data?

- "Plot a map showing all commercial break-and-enters."
- A. Yes
- B. No, because the data is incomplete and does not include all break-and-enter incidents.
- C. No, because the program cannot distinguish between commercial and residential break-and-enters.
- D. No, because the needed information isn't available in the data.
- E. No, because it doesn't do a substantial computation.

► Hint (for your review after class)

Step 1b: Plan output - Example

Brainstorm ideas for what your program will produce

Select the idea you will build on for subsequent steps

You must brainstorm at least three ideas for graphs or charts that your program could produce and choose the one that you'd like to work on. You can choose between a line chart, histogram, bar chart, scatterplot, or pie chart.

✓ For this lecture, think of at least:

- one quantity/number you'd be interested to see computed from the information and
- one graph showing some kind of relationship.

Put your answer here. Please don't delete the two HTML tags on either end of this paragraph. It's to make your answer blue so the TAs can easily spot it.

► Sample solution (For later. Don't peek if you want to learn 😊)

Step 1c: Plan examples

Write or draw examples of what your program will produce:

- Add an example of how you are going to run your program (function name, its inputs or parameters, and expected output).
- Describe any values it should return.
- Attach a drawing of the graph you are going to plot.
- Your drawing is just a sketch to give the reader an idea of what the graph might look like. It's not based on the data. (If we already knew the answer, why would we need the program?)

To insert an image into a Jupyter notebook:

1. Edit a markdown (not code) cell.

2. Place the cursor at the position in the cell where you want the image to appear.
3. Then either,
 - a. Drag and drop an image file (e.g., PNG, JPG) from your computer, OR
 - b. Select menu `Edit > Insert Image` then `[Browse] > (select file) > [OK]` .

You will see text like `! [image.png] (attachment: image.png)` inserted at the cursor position. When you `[▶run]` the cell, the image will appear.


Step 1c: Plan examples - Example

Write or draw examples of what your program will produce

You must include an image that shows what your chart or plot will look like. You can insert an image using the Insert Image command near the bottom of the Edit menu.

- ✓ For this lecture, let's say we've decided in Step 1b that our program will produce the following:
- Given a type of crime, find the time of day (hour) with the highest frequency**

Insert your example text or image in this cell. Feel free to remove this prompt from this cell.

- ▶  Sample solution (For later. Don't peek if you want to learn 😊)

Step 2a: Design data definitions

Design data definitions:

- You should design data that your program is going to use.
- You should focus only on the fields/columns that your program is going to use and choose the correct type for each one.
- If you choose a non-primitive (interval, enumeration, optional...) you have to design it.
- Create a compound data to represent one line of data on the CSV file (`Consumed`).
- Create a list from this compound data to represent all lines of data in the CSV file (`List[Consumed]`).
- You may need more data that can be added later (for example, `List[int]` or `List[str]`).

Step 2a: Design data definitions - Example

Document which information you will represent in your data definitions

Before you design data definitions in the code cell below, you must explicitly document here which information in the file you chose to represent and why that information is crucial to the chart or graph that you'll produce when you complete step 2c.

Note: we'll skip the "chart or graph" part!

- ✓ For this example, we'll skip the "chart or graph" because our program is producing a numerical quantity, instead (a particular hour of the day).

Put your answer here. Please don't delete the two HTML tags on either end of this paragraph. It's to make your answer blue so the TAs can easily spot it.

Design data definitions

```
In [ ]: 1 from cs103 import *
        2 from typing import NamedTuple, List
        3 import csv
        4
        5 #####
        6 # Data Definitions
        7
        8 Consumed = ...
        9
       10
       11
       12 # List[Consumed]
       13 # interp. a list of Consumed
       14
       15 LOC0 = []
       16
       17 @typecheck
       18 def fn_for_loc(loc: List[Consumed]) -> ...:
       19     ... # choose which template body to use for List[Consumed]
       20
       21
```

►  Sample solution (For later. Don't peek if you want to learn 😊)

Now it's your turn

Recall, the steps in the HtDAP recipe are:

1. Planning
 - a. Plan input: Identify the information in the file your program will read.
 - b. Plan output: Write a description of what your program will produce.
 - c. Plan examples: Write or draw examples of what your program will produce.
2. Designing the program
 - a. Design data definitions.
 - b. Design read function: Design a function to read the information and store it as data in your program.
 - c. Design analyze function: Design functions to analyze the data.

Can you apply Step 1(a-c) to your project? Let's get started!

Got a project partner? Team up with them now!