Computing for Medicine: Phase 3, Seminar 6 Project

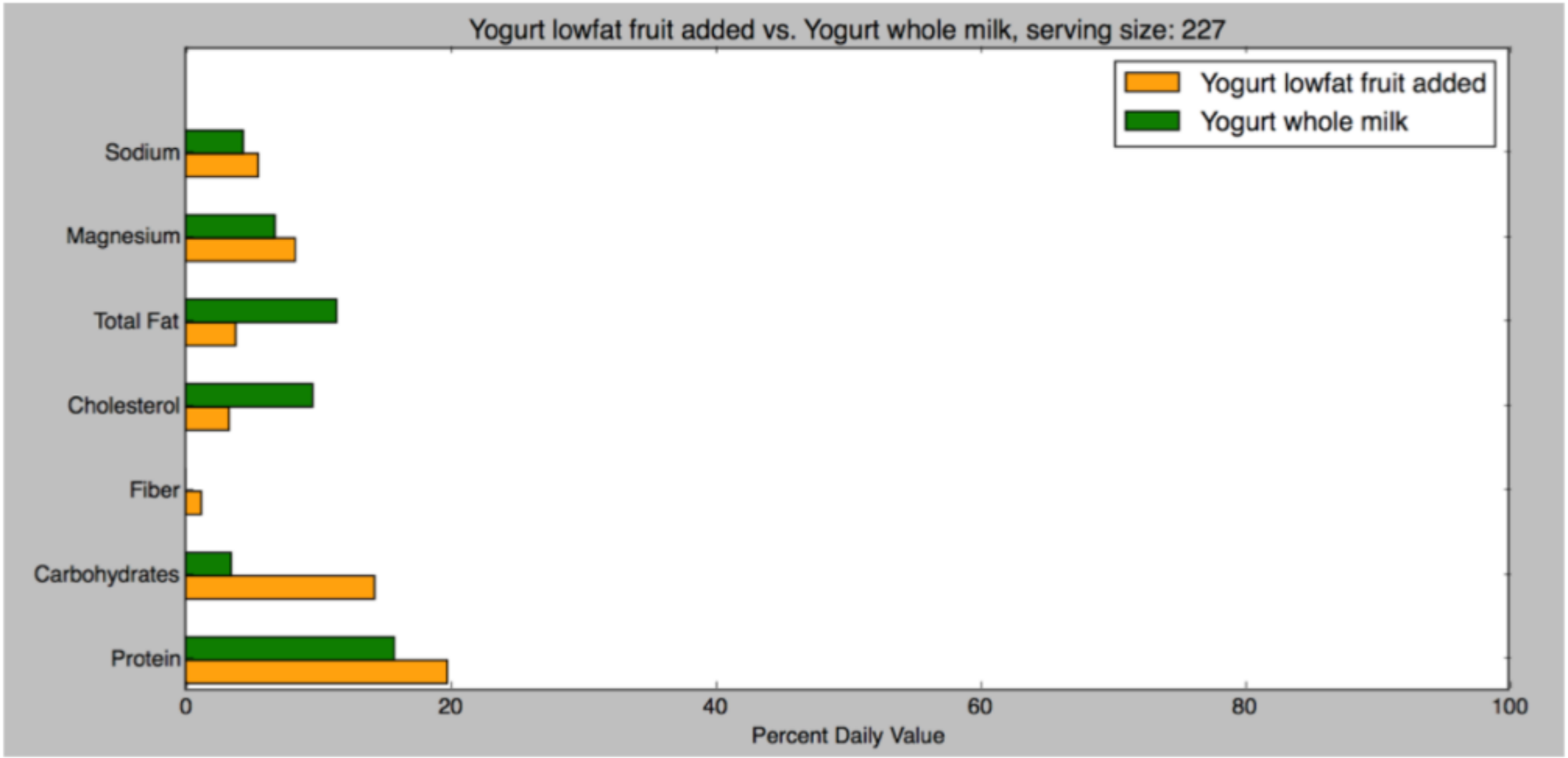
Jennifer Campbell
Associate Professor, Teaching Stream
c4m@cs.toronto.edu

Seminar 6 Project

- The project handout is posted:
  - http://c4m.cdf.toronto.edu/cohort2/phase3/

- Two approaches for doing your work:
  - Use the Computer Science Teaching Labs computing network.
  - Use your personal computer.

- Python3 packages to install:
  - `numpy`
  - `matplotlib (pyplot)`


Yogurt lowfat fruit added vs. Yogurt whole milk, serving size: 227
Yogurt lowfat fruit added
Yogurt whole milk
Sodium
Magnesium
Total Fat
Cholesterol
Fiber
Carbohydrates
Protein
0
20
40
60
80
100
Percent Daily Value

JSON

JSON (JavaScript Object Notation)

- Standard data-interchange format.
- Commonly used in web programming for communication between a web browser and server.
- Example of JSON to represent a person:

```
{
   "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
   "age": 25,
   "children": []
}
```

(Example source: Wikipedia)

Another JSON Example

```
{
        "id": 1,
        "name": "Foo",
        "price": 123,
        "tags": [ "Bar", "Eek" ],
        "stock": {
                "warehouse": 300,
                "retail": 20
        }
}
```

(Example source: Wikipedia)

Terminology from handout

- *"In this project, we will work with food labels stored in a JSON format since most APIs (e.g. Open Food Facts, MyNetDiary, Spoonacular's food API) provide detailed information in this format."*

- API: Application programming interface
  - A set of programming routines (e.g., functions) used for producing software applications.

EXPLORING MATPLOTLIB

`matplotlib.pyplot`

- http://matplotlib.org/api/pyplot_api.html

- For this project, you will need to explore the `pyplot` documentation to find appropriate functions to use for the data visualization tasks.
- Demo: using `pyplot` to display a pie chart.

PYTHON:
IMPORT AND MAIN

Example 1: witho

```
def fahr_to_cels(temp):
    return (temp - 32) *
t = input("Enter a temp:
result = fahr_to_cels(float(t))
print("Celsius:", result)
```

convert.py

Sample run of `temperature.py`:

```
Enter a temp: 102.6
Celsius: 39.22222222222222
Patient's temp: 101.4
Fever report:  True
```

```
import convert
def has_fever(c_temp):
    f_temp = convert.fahr_to_cels(c_temp)
    return f_temp > 37.5
t = input("Patient's temp: ")
result = has_fever(float(t))
print("Fever report: ", result)
```

temperature.py

Example 2: with _

Sample run of `temperature.py`:

```
Patient's temp: 101.4
Fever report:  True
```

```
def fahr_to_cels(temp):
    return (temp - 32) * 5 / 9
if __name__ == '__main__':
    t = input("Enter a temp: ")
    result = fahr_to_cels(float(t))
    print("Celsius:", result)
```

convert.py

```
import convert
def has_fever(c_temp):
    f_temp = convert.fahr_to_cels(c_temp)
    return f_temp > 37.5
t = input("Patient's temp: ")
result = has_fever(float(t))
print("Fever report: ", result)
```

temperature.py

Summary

- Importing a module, executes the code in that module.
- If the module being imported contains a main block (`if __name__ == '__main__'`), the code within the main block will NOT be executed when that module is imported.
- However, when that module is run directly, both the code inside and outside of the main block is executed.

How this applies to your project

- You will write code in three files:
  - `seminar6_part1.py`
  - `seminar6_part2.py`
  - `seminar6_part3.py`
- To reuse functions written in part 1, you will import `seminar6_part1` in `seminar6_part2.py`.
- To prevent the user interaction code from `seminar6_part1` from being executed when that module is imported by `seminar6_part2`, place that code within a main block.