CSC148 - Week 1*

Arnamoy Bhattacharyya

*Most slides inspired by Bogdan Simion and Danny Heap

Overview

- Course logistics
- What is CSC148 about?
- Brief Python review
- Introduction to Object-Oriented Programming

Instructor Contract

- Email: arnamoycsc148@cs.toronto.edu (**please include CSC148 in the subject**)
- Office Hours: Monday, 11AM - 1PM @ BA3219
- Webpage: http://www.teach.cs.toronto.edu/~csc148h/winter/
- All course materials posted here
- Discussion board - Piazza:
  - Linked from course webpage. Read, ask questions, collaborate (do not post your code!)
- Course Info Sheet (due dates, policies, etc.):
  - Linked from course webpage, MUST read carefully!

Course Info

Assignments x 2: due at 10PM on the due date
Remark requests: submit within 7 days of results being released

Lab/exercises x 8
Starts next week

Tests x 2

Final exam

You must get > 40% to pass this course!

See weights and policies on course sheet!

Active participation

- Strong evidence that people learn better or faster by doing rather than passively listening
- Ask questions, work on exercises, participate

Assignments

- Start early on the assignments!
- Make sure you can submit and submit periodically
- Build gradually, test your code!
- Do not wait until the very last minute to submit your assignment!

Video on how to submit

Help is there for you

- **Help** is available in many forms
- Lectures/labs: Ask questions!
- Office hours: My time dedicated specifically to helping you
- Piazza: collaborative
- Email: Longer turnaround time
- Undergraduate TA Help Center: http://web.cs.toronto.edu/program/ugrad/ug_helpcentre.htm

Plagiarism -- a strict No-No!

- Very serious academic offences
- Clear distinction between collaboration and cheating
- Of course you can help your friend track down a bug
- It is never ok to submit code that is not your own!
- Ask questions on Piazza, but don’t add details about your solution (especially your code!)
- All potential cases will be investigated fully
- Don’t post your code in public places (Github, etc.)
- We will run plagiarism detection software!

What we expect you to know

- From CSC108:
  - **if** statements, **for** loops, **function** definitions and calls, **lists**, **dictionaries**, **searching**, **sorting**, **classes**, **documentation** style.
- We assume you know this!
- Sign up for the ramp-up session!
  - https://doodle.com/poll/2arm5xn44zxn7zda (posted in Portal/Blackboard)
  - Indicate which session you wish to attend

What is CSC148 about?

- How to understand and write a solution for a real-world problem
- Abstract Data Types (**ADTs**) to represent and manipulate information
- **Recursion**: clever functions that call themselves
- **Exceptions**: how to handle unexpected situations
- **Testing**: how to write maintainable, correct code
- **Design**: how to structure a program (some OOP)
- **Efficiency**: how much resources (time/space) does a program use?

Remember

- Write good, well-documented code!
- Test your code!
- Practice makes perfect!
- You must get your hands dirty and try things yourselves!

Python (brief review)

How to design a function

CSC108 teaches a “**recipe**” for writing functions (and methods)

Adapted recipe for 148:

- Write examples of calls and the expected returned values
- Write a ***type contract*** that identifies the return value and the type of each parameter
- Write the function header
- Add a one-line summary of what the function does, above the type contract
- Write the function body
- Test your function, add more examples (tricky corner cases)

The type contract

- One style of type annotation:

  @type parameter: type

  @rtype: type (“return type”)

- Alternative for parameter annotation:

  @param type parameter: description

Allows pycharm to check that your code conforms

The type contract

- One style of type annotation:

  @type parameter: type

  @rtype: type (“return type”)

- Alternative for parameter annotation:

  @param type parameter: description

Allows pycharm to check that your code conforms

```
def sum_numbers(a, b):
    """
    @type a:int
    @type b:int

    @param int a: an integer a
    @param int b: another integer b

    @rtype: None
    """
```

Exercise

Design a function length_is_multiple that takes two argument, a string and an integer. The function returns true if the length of the string is a multiple of the integer, otherwise it returns false.

- Write examples of calls and the expected returned values
- Write a *type contract* that identifies the return value and the type of each parameter
- Write the function header
- Add a one-line summary of what the function does, above the type contract
- Write the function body

```
>>> length_is_multiple("two",3)
True
>>> length_is_multiple("two",2)
False
```

Onto Pycharm

Implementation

```
def length_is_multiple(string, num):
    """
    Return whether the length of the given string is
    multiple of num

    @param str string: a string
    @param int num: a whole number
    @rtype:bool

    >>> length_is_multiple("two",3)
    True
    >>> length_is_multiple("two",2)
    False
    """
    return len(string) % num == 0
```