MY470 Computer Programming

What Is Computation?

Week 1 Lecture

Overview

- Computational thinking and algorithms
- Computers, programming languages, and computer programs
- Objects, expressions, and variables
- Debugging
- Version control with GitHub

Computational Thinking

> Computational Thinking is the thought processes involved in formulating a problem and expressing its solution in a way that a computer — human or machine — can effectively carry out.

Wing, Jeannette M. (2006). Computational thinking (http://tech-insider.org/academia/research/acrobat/0603.pdf). *Communications of the ACM*, 49(3), 33-35.

Defining Characteristics of Computational Thinking

Wing, Jeannette M. (2006). Computational thinking (http://tech-insider.org/academia/research/acrobat/0603.pdf). *Communications of the ACM*, 49(3), 33-35.

- **Conceptualizing**, not programming — requires thinking at multiple levels of abstraction

- A way that **humans**, not computers, think — requires cleverness and imagination

- Combines **mathematical and engineering** thinking — dictated by the constraints of physical computing devices

- For **everyone**, everywhere — just like reading, writing, and arithmetic

Algorithms

An algorithm is a well-defined computational procedure that takes value(s) as input and produces value(s) as output.

- "Recipe" or "instructions" for solving a well-defined computational problem
- Consists of a sequence of simple steps, control flow, and a stopping rule
- Can be specified in human language or programming language (or even as hardware design)

For example, **a sorting algorithm**

- Takes as input a sequence of numbers
- Returns a permutaion (an ordering) of the input sequence such that successive numbers are larger or equal

Computers

Computers automatically perform calculations, either built-in or user-defined, and store the results.

Programming Languages

A programming language is a formal language used to specify a set of instructions for a computer to execute.

- Primitive constructs
- Syntax
- Static semantics
- Semantics

Markup vs. Programming Languages

	Markup Languages	Programming Languages

	<?xml version="1.0" encoding="UTF-8"?>
	<!DOCTYPE recipe PUBLIC "~//Happy-Monkey//DTD RecipeBook//EN"
	"http://www.happy-monkey.net/recipebook/recipebook.dtd">
	
	<recipe>
	
	<title>Peanut-butter On A Spoon</title>
	
	<ingredientlist>
	<ingredient>Peanut-butter</ingredient>
	</ingredientlist>
	
	<preparation>
	Stick a spoon in a jar of peanut-butter,
	scoop and pull out a big glob of peanut-butter.
	</preparation>
	
	</recipe>
		def add5(x):
		return x+5
		
		def dotwrite(ast):
		nodename = getNodename()
		label=symbol.sym_name.get(int(ast[0]),ast[0])
		print ' %s [label="%s' % (nodename, label),
		if isinstance(ast[1], str):
		if ast[1].strip():
		print '= %s"];' % ast[1]
		else:
		print '"]'
		else:
		print '"];'
		children = []
		for n, child in enumerate(ast[1:]):
		children.append(dotwrite(child))
		print ' %s -> {' % nodename,
		for name in children:
		print '%s' % name,

Examples	TeX, HTML, XML, **Markdown**	C, Java, JavaScript, **Python**, **R**
Use	Structure and present data	Transform and generate data
Execution	Program (e.g. a browser)	Computer hardware
Structure	Inline tags	Primitive constructs, syntax, static semantics, semantics

(Image sources: Wikimedia)

Primitive Constructs in Programming Languages

- Literals

In [2]:
```
470
```

Out[2]:
```
470
```

In [2]:
```
'MY'
```

Out[2]:
```
'MY'
```

- Infix operators

In [3]:
```
470/3
```

Out[3]:
```
156.66666666666666
```

Syntax in Programming Languages

- Rules for forming strings of characters and symbols
- Programming languages have strict syntax

In [4]:
```
470 + 0.5
```

Out[4]:
```
470.5
```

In [5]:
```
470 0.5
```

```
  File "<ipython-input-5-5a5b76bbe317>", line 1
    470 0.5
          ^
SyntaxError: invalid syntax
```

Static Semantics in Programming Languages

- Rules for forming meaningful syntactically valid strings

In [6]:
```
'MY'/470
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-fb25aaf6edea> in <module>()
----> 1 'MY'/470

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

Semantics in Programming Languages

- The meaning associated with a syntactically correct string that has no static semantic errors
- Programming languages have simple semantics — statements have only one meaning

- **But this may not be the meaning the programmer had in mind!**

Types of Programming Languages

- Low-level vs. high-level
- General vs. application-targetted
- Interpreted vs. compiled

Computer Program

- A sequence of definitions and commands
  - Commands (or "statements") instruct the computer to do something
- For interpreted languages:
  - Programs are executed by the language interpreter (or "shell")
  - They can be typed directly in the shell
  - Or they can be stored in a file and run from the shell

Objects, Data Types, and Expressions

- Programs manipulate objects
- Objects have types
  - Scalar — indivisible
  - Non-scalar — with internal structure
- Expressions combine objects and operators

In [ ]:
```
# scalar objects
2
0.125
True
```

In [ ]:
```
# non-scalar objects
'This is a string.'
[1, 2, 3, 'a', 'x']
```

In [ ]:
```
# expressions
2/0.125
'MY' + '470'
```

Variables

- Variables associate objects with a name

In [ ]:
```
a = 3.14
b = 11.2
c = a*(b**2)
```

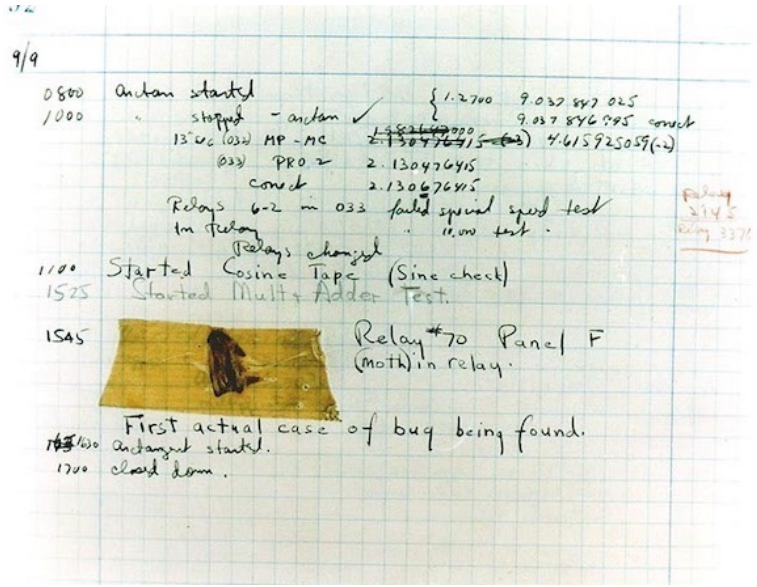
In [ ]:
```
pi = 3.14
diameter = 11.2
area = pi*(diameter**2)
```

- 📖 **Variable names help humans read programs!**

- 📖 **Comments also improve legibility!**

In [ ]:
```
pi = 3.14
diameter = 11.2 # diameter of circle
area = pi*((diameter/2)**2) # estimate area of circle using diameter
```

Computer Bugs



The actual first computer bug. On September 9, 1947, Admiral Grace Hopper found this moth trapped on a relay of the Harvard Mark II computer. (Image source: U.S. Naval Historical Center Online Library)

What Is Computer Programming Really about?

99 little bugs in the code,

99 bugs in the code,

1 bug fixed...run again,

100 little bugs in the code...

How to Debug: Two Options

1. **Google** the error and find an answer on **Stackoverflow**
2. Use **`print()`** systematically

The `print` Function in Python

In [7]:
```
print('The')
print('The', 'winning', 'number', 'is', 7, '.')
print('The winning number is '+ str(7) + '.')
```

```
The
The winning number is 7 .
The winning number is 7.
```

Debugging Systematically

1. Compare input in successful and failing runs
2. Formulate a hypothesis
3. Design an experiment to test the hypothesis; use `print()`
4. Keep record of your experiment
5. Repeat

After Debugging for Hours...

- Stop

- Try commenting your code or explaining it to someone else

- Sleep on it

(Image source: Reddit)

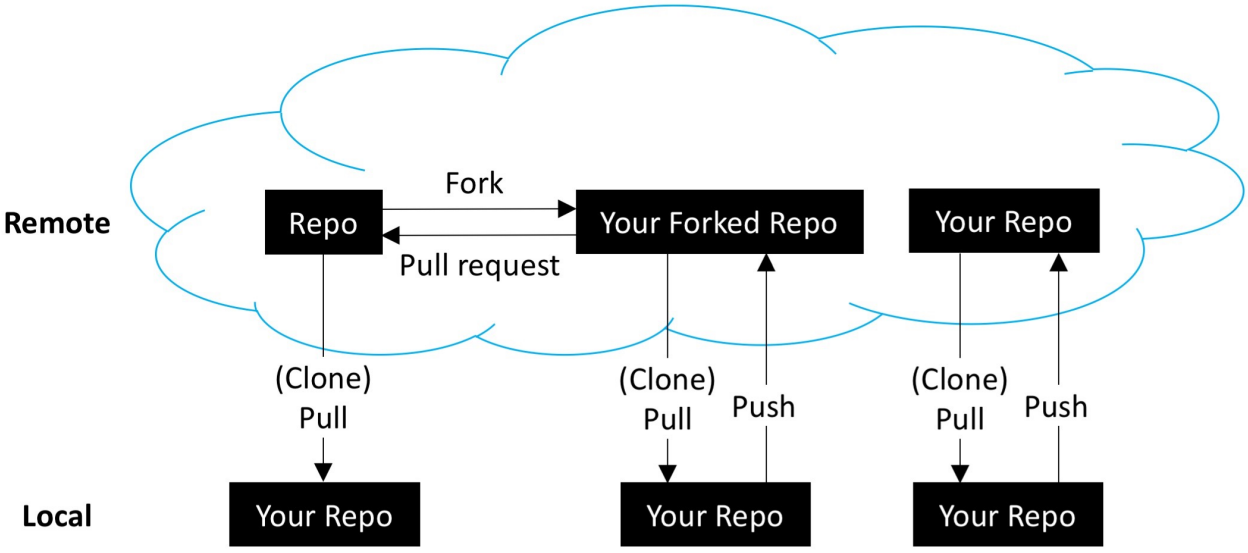
Version Control with GitHub

- Code hosting platform for version control and collaboration
- Based on Git
  - Version control system for tracking changes in computer files and coordinating work on those files among multiple people
  - Created in 2005 by Linus Torvalds
- Largest host of source code in the world
- Bought by Microsoft in 2018

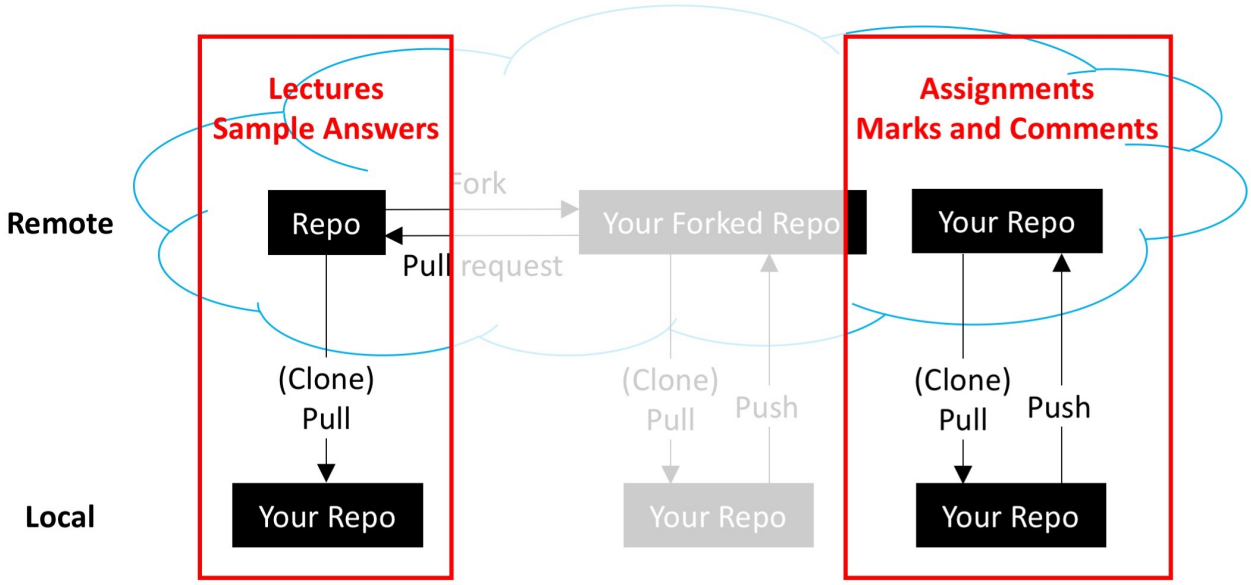
GitHub Lingo

- **Repository** – a space for a project/assignment
- **Clone** – a copy of the repository that lives on your computer
- **Branch** – a paralel version of the repository
- **Commit** – save changes with a short description
- **Pull request** – ask changes to be merged
- **Merge** – incorporate changes (then delete branch)

GitHub Workflow



GitHub Workflow in MY470



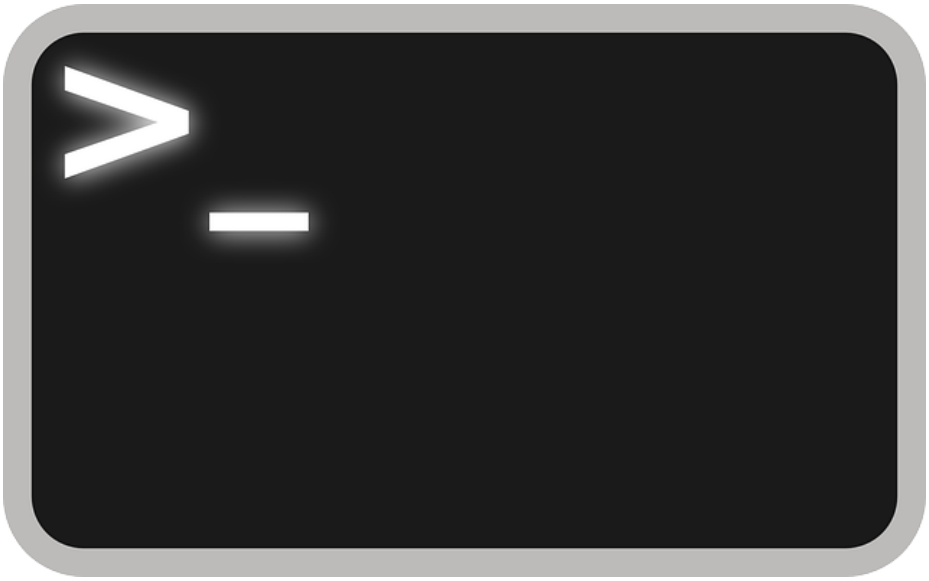
Getting Started with GitHub

- Create personal account on https://github.com/ (https://github.com/)
- Go to https://education.github.com/ (https://education.github.com/) and get the Student Developer Pack for some cool freebies

- Three ways to interact (covered in lab)
  1. Browser
  2. Command line
  3. VS Code (or alternative IDE/app)

Terminal = Console = Shell = Command Line = Command Prompt

(for our purposes here)



- Efficient way to access files, run programs, and execute code
- Allows to schedule and batch-process tasks
- Provides scripts for reproducible workflows across different operating systems

Useful Bash Commands

- Print current working directory

```
pwd
```

- Change current working directory

```
cd Path/to/directory
```

- Go back to the parent directory of the current one

```
cd ..
```

- Go back to your home directory

```
cd ~
```

- Create a new directory

```
mkdir dirname
```

- Print a list of files and subdirectories

```
ls
```

- Launch a Python interpreter (type `exit()` to stop and go back to bash)

```
python
```

Change Your Default Text Editor for Git

You can use your favorite editor by customizing the Git default editor.

For example, you can use **Nano** (https://www.nano-editor.org/). It is much easier to use than Vim: `Ctrl+o` to save and `Ctrl+x` to close.

To set Nano as the default editor for your commit messages, run the following:

```
git config --global core.editor "nano"
```

Nano comes pre-installed with Linux and OS. For Windows, download and install **Nano-win** (https://github.com/mcandre/nano-win).

Important Git Commands

- Copy online repository

```
git clone https://github.com/lse-my470/lectures.git
```

- Update local repository

```
git pull
```

- See the status of local respository

```
git status
```

- See the change history of local respository

```
git log
```

- Stage all changes

```
git add --all
```

- Commit staged changes

```
git commit -m "your commit message here"
```

- Upload your changes to online repository

```
git push
```

Resources

- Get started: GitHub tutorials (https://guides.github.com/)
- Get it done: Git cheatsheet (https://education.github.com/git-cheat-sheet-education.pdf)

What Is Computation?

We use programming languages to write programs that instruct computers to perform algorithms, which calculate results or process data.

- **Lab**: Installing Anaconda, working with VS Code and Jupyter notebooks, uploading assignments on GitHub
- **Next week**: Data types in Python