Week 1: Introduction to Computation

POP77001 Computer Programming for Social Scientists

Tom Paskhalis

12 September 2022

Module website: bit.ly/POP77001

Overview

- Computers and Computational thinking
- Algorithms
- Programming languages and computer programs
- Debugging
- Command-line Interfaces
- Version controlling with Git/GitHub

Computers

1940	2022

More Computers

Antikythera mechanism (c.100 BC)	Difference Engine (1820s)	Collosus (1940s)	Deep Blue (1997)

Computers

Computers

- Do two things:

Computers

- Do two things:

```
1. Perform calculations
```

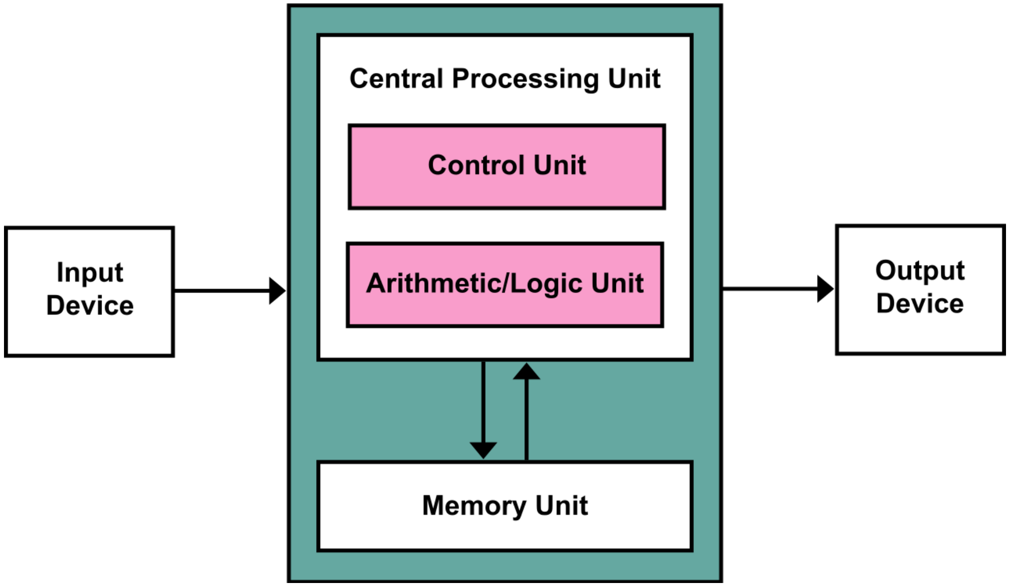
Computers

- Do two things:

```
1. Perform calculations
```

```
2. Store results of calculations
```

von Neumann Architecture



Source: Wikipedia

Computational Thinking

Computational thinking is breaking down a problem and formulating a solution in a way that both human and computer can understand and execute.

- *Conceptualizing, not programming* - multiple levels of abstraction
- *A way, that humans, not computers, think* - creatively and imaginatively
- *Complements and combines mathematical and engineering thinking*

Wing, Jeannette M. 2006. "Computational Thinking." Communications of the ACM, 49 (3): 33–35. doi: 10.1145/1118178.1118215

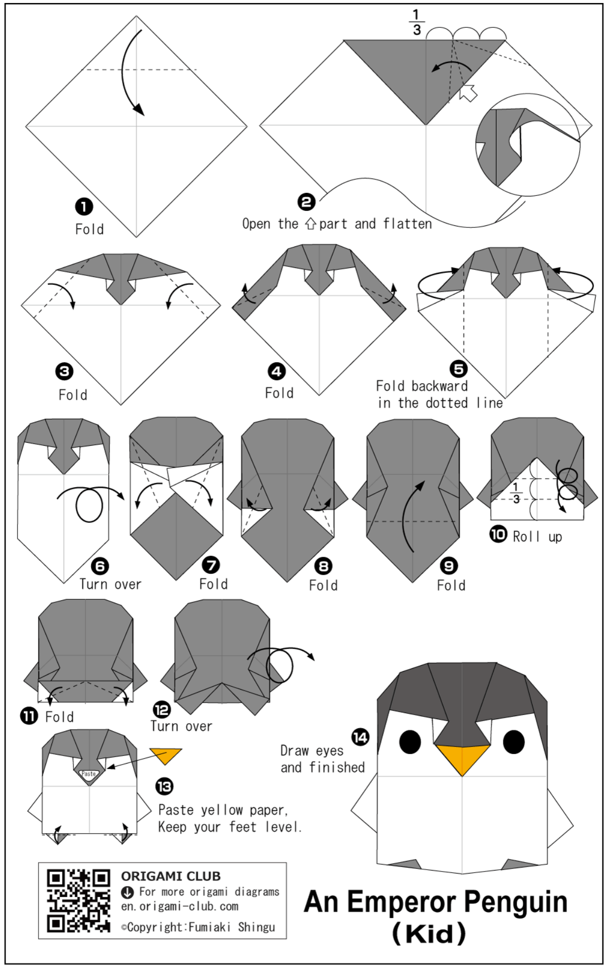
Computational Thinking

- All knowledge can be thought of as:
    1. Declarative (statement of fact, e.g. square root of 25 equals 5)
    2. Imperative (how to, e.g. to find a square root of x, start with a guess g, check whether g*g is close, ...)

Algorithm

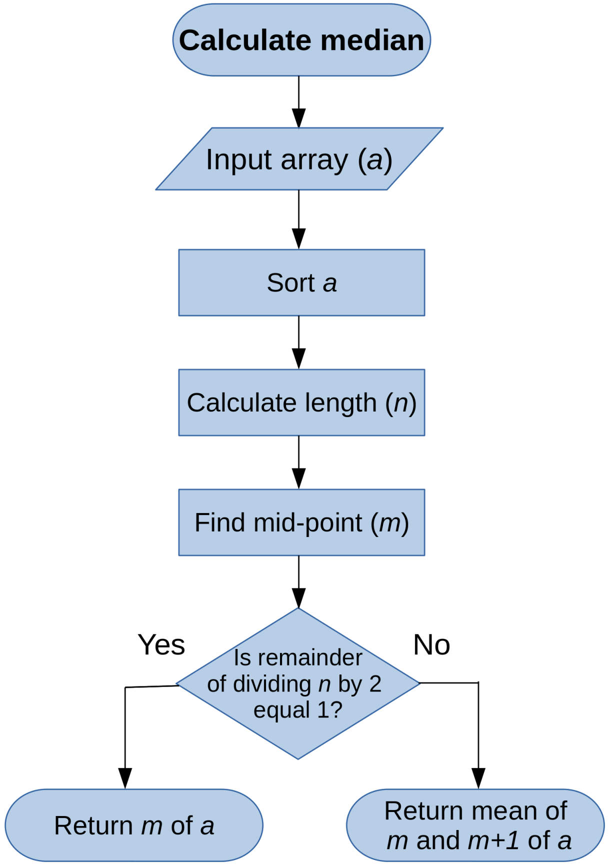
- *Finite list of well-defined instructions that take input and produce output.*
- Consists of a sequence of simple steps that start from input, follow some control flow and have a stopping rule.

Algorithm Example



Source: Origami Club

Algorithm Example



Programming Language

Formal language used to define sequences of instructions (for computers to execute) that includes:

- Primitive constructs
- Syntax
- Static semantics
- Semantics

Types of Programming Languages

- Low-level vs high-level
    - E.g. available procedures for moving bits vs calculating a mean
- General vs application-domain
    - E.g. general-purpose vs statistical analysis
- Interpreted vs compiled
    - Source code executed directly vs translated into machine code

Primitive Constructs in R/Python

- Literals

Primitive Constructs in R/Python

- Literals

In [1]:
```
77001
```

Out[1]:
```
77001
```

Primitive Constructs in R/Python

- Literals

In [1]:
```
77001
```

Out[1]:
```
77001
```

In [2]:
```
'POP'
```

Out[2]:
```
'POP'
```

Primitive Constructs in R/Python

- Literals

In [1]:
```
77001
```
Out[1]:
```
77001
```
In [2]:
```
'POP'
```
Out[2]:
```
'POP'
```

- Infix operators

Primitive Constructs in R/Python

- Literals

In [1]:
```
77001
```
Out[1]:
```
77001
```
In [2]:
```
'POP'
```
Out[2]:
```
'POP'
```

- Infix operators

In [3]:
```
77001 + 23
```
Out[3]:
```
77024
```

Syntax in R/Python

- Defines which sequences of characters and symbols are well-formed
- E.g. in English sentence "Cat dog saw" is invalid, while "Cat saw dog" is.

Syntax in R/Python

- Defines which sequences of characters and symbols are well-formed
- E.g. in English sentence "Cat dog saw" is invalid, while "Cat saw dog" is.

In [4]:
```
77001 + 23
```

Out[4]:
```
77024
```

Syntax in R/Python

- Defines which sequences of characters and symbols are well-formed
- E.g. in English sentence "Cat dog saw" is invalid, while "Cat saw dog" is.

In [4]:
```
77001 + 23
```

Out[4]:
```
77024
```

In [5]:
```
77001 23 +
```

```
  File "<ipython-input-5-7df1296e3e0d>", line 1
    77001 23 +
           ^
SyntaxError: invalid syntax
```

Static Semantics in R/Python

- Defines which syntactically valid sequences have a meaning
- E.g. in English sentence "Cat seen dog" is invalid, while "Cat saw dog" is.

Static Semantics in R/Python

- Defines which syntactically valid sequences have a meaning
- E.g. in English sentence "Cat seen dog" is invalid, while "Cat saw dog" is.

In [6]:
```
'POP' + 77001
```

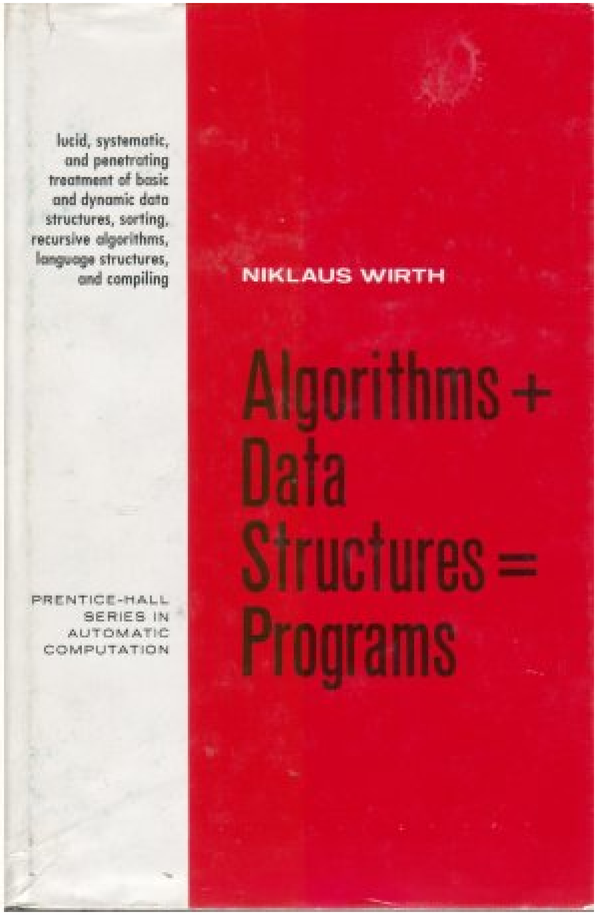
```
---------------------------------------------------------------
------------
TypeError                                 Traceback (most recen
t call last)
<ipython-input-6-7d1b83a55295> in <module>
----> 1 'POP' + 77001

TypeError: can only concatenate str (not "int") to str
```

Semantics in Programming Languages

- Associates a meaning with each syntactically correct sequence of symbols that has no static semantic errors
- Programming languages are designed so that each legal program has exactly one meaning
- **This meaning, however, does not, necessarily, reflect the intentions of the programmer**
- Syntactic errors are much easier to detect

Algorithms + Data Structures = Programs



Computer Program

- A collection of instructions that can be executed by computer to perform a specific task
- For interpreted languages (e.g. Python, R, Julia) instructions (source code)
  - Can be executed directly in the interpreter
  - Can be stored and run from the terminal

Programming Errors

- Often, programs would run with errors or behave in an unexpected way
- Programs might crash
- They might run too long or indefinitely
- **Run to completion and produce an incorrect output**

Computer Bugs

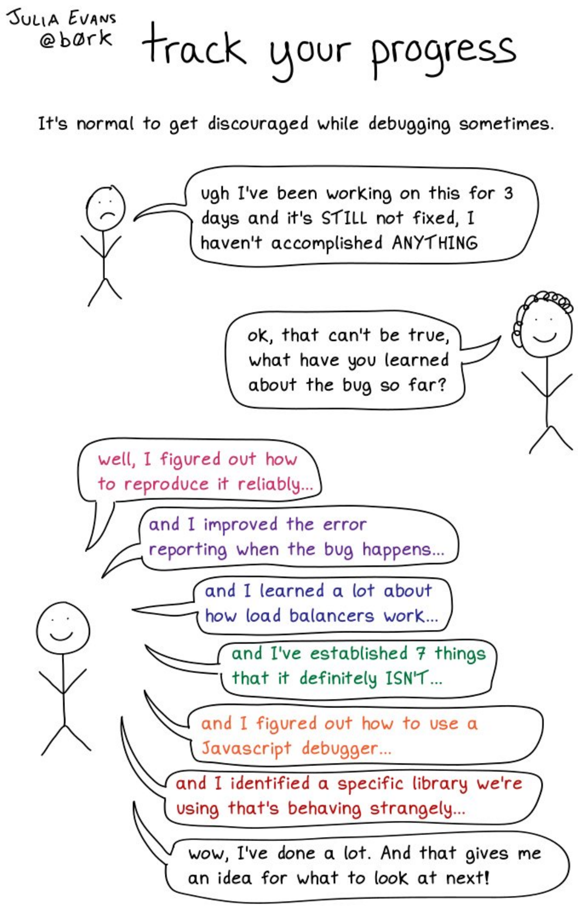
Grace Murray Hopper popularised the term *bug* after in 1947 her team traced an error in the Mark II to a moth trapped in a relay.

Source: US Naval History and Heritage Command

How to Debug

- Search error message online (e.g. StackOverflow or, indeed, #LMDDGTFY)
- Insert `print()` statement to check the state between procedures
- Use built-in debugger (stepping through procedure as it executes)
- More to follow!

Debugging



Source: Julia Evans

Command-line Interface (aka terminal/console/shell/command line/command prompt)

- Most users today rely on graphical interfaces
- Command line interpreters (CLIs) provide useful shortcuts
- Computer programs can be run or scheduled in terminal/CLI
- **CLI/terminal is usually the only available interface if you work in the cloud (AWS, Microsoft Azure, etc.)**

Extra: Five reasons why researchers should learn to love the command line

CLI Examples

Microsoft PowerShell (Windows)	Z shell, zsh (macOS)	bash (Linux/UNIX)

Some Useful CLI Commands

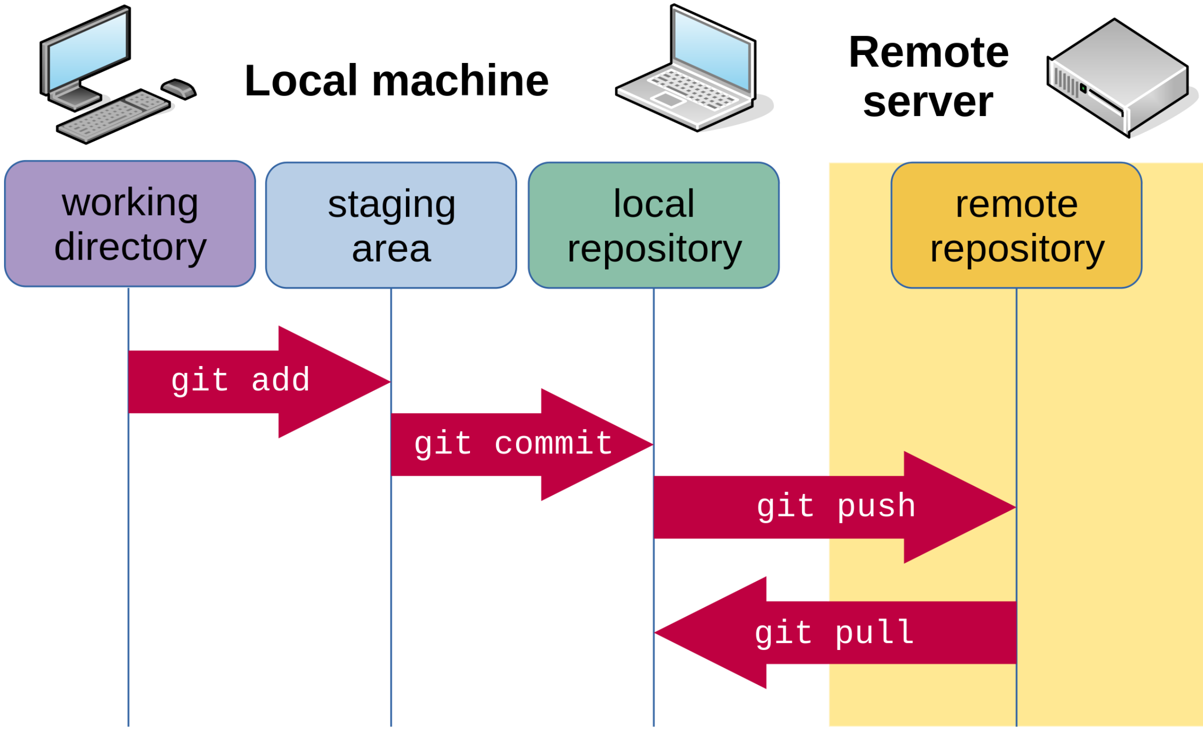
Command (Windows)	Command (macOS/Linux)	Description
`exit`	`exit`	close the window
`cd`	`cd`	change directory
`cd`	`pwd`	show current directory
`dir`	`ls`	list directories/files
`copy`	`cp`	copy file
`move`	`mv`	move/rename file
`mkdir`	`mkdir`	create a new directory
`del`	`rm`	delete a file

Extra: Introduction to CLI

Version Control and Git

- Version control systems (VCSs) allow automatic tracking of changes in files and collaboration
- Git is one of several major version control systems (VCSs, see also Mercurial, Subversion)
- GitHub is an online hosting platform for projects that use Git for version control

Git/GitHub Workflow



Some Useful Git Commands

Command	Description
`git init <project name>`	Create a new *local repository*
`git clone <project url>`	Download a project from *remote repository*
`git status`	Check project status
`git diff <file>`	Show changes between *working directory* and *staging area*
`git add <file>`	Add a file to the *staging area*
`git commit -m “<commit message>”`	Create a new *commit* from changes added to the *staging area*
`git pull <remote> <branch>`	Fetch changes from *remote* and merge into *merge*
`git push <remote> <branch>`	Push local branch to *remote repository*

Extra: Git Cheatsheet

Things to Do

- Make sure you have installed R, Python and Jupyter.
- Make sure you can run R code through Jupyter (requires extra steps).
- Do the readings.

Things to Try (CLI)

- Identify an appropriate CLI for your OS.
- Try navigating across folders and files using CLI.
- Try creating a test folder and test file inside it.

Things to Try (git)

- Register on GitHub and GitHub Education (for free goodies!)
- Create a test repository in CLI and initialise as a Git repository
- Or create a repository on GitHub and clone to your local machine
- Create test.txt file, add it and commit
- Push the file to GitHub

Next

- Tutorial: Jupyter Notebooks, CLIs, Git/GitHub
- Next week: R Basics