CMPUT 175 (B1, B2)
Introduction to Foundations of Computing

UNIVERSITY OF ALBERTA

amii

Osmar R. Zaïane, Ph.D.
McCalla, Killam Professor
Department of Computing Science

Scientific Director

443 Athabasca Hall
Edmonton, Alberta
Canada T6G 2E8

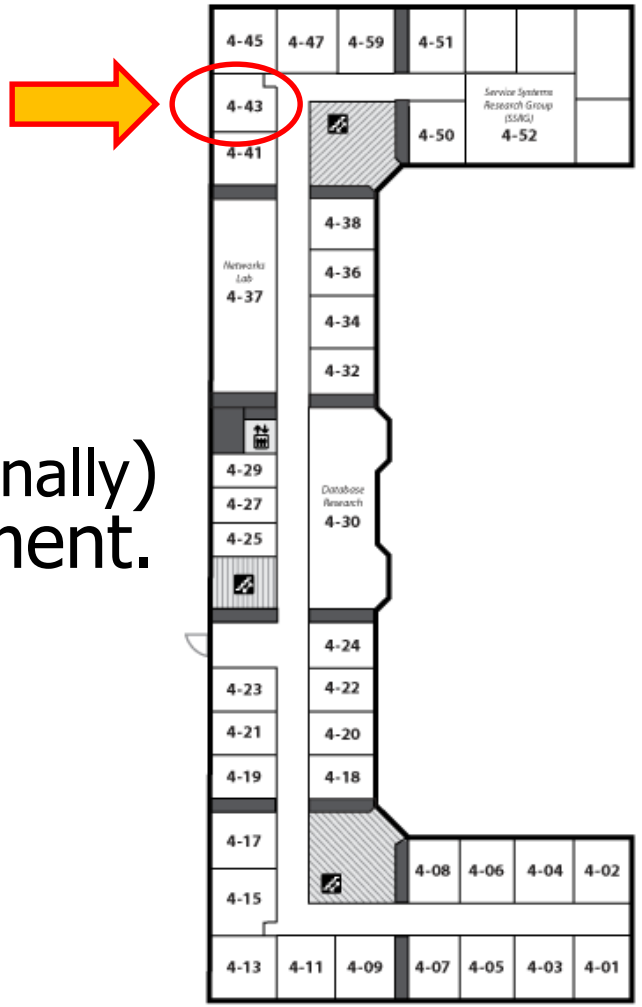
Telephone: Office +1 (780) 492 2860
Fax +1 (780) 492 1071
E-mail: **zaiane@cs.ualberta.ca**
http://www.cs.ualberta.ca/~zaiane/

蔡頤安

@ozaiane

Instructor Information

- Prof. Osmar Zaïane
- Office: Athabasca Hall, room 4-43
- Office hours: 1:00-2:00pm, every Tuesday and Thursday or (exceptionally) by mutually agreed upon appointment.
- TAs also have office hours
- E-mail: zaiane@cs.ualberta.ca (use your UofA account and prefix subject line with "**CMPUT175**")



Course Staff

- Instructors

  - **Osmar Zaïane**
    B1 9:00-9:50
    CAB 243
    115 students
    B2 10:00-10:50
    ED 129
    115 students

  - **Megan Flanders**
    B3 12:00-12:50
    CSC B2
    92 students
    B5 14:00-13:50
    CSC B2
    55 students (31 available)

  - **Greg Kondrak**
    B4 14:00-14:50
    CCIS L1 140
    65 students (31 available)

- Teaching Assistants (B1, B2, B3, B4)
  - **H M Ata-E-Rabbi** ataerabb@ualberta.ca
  - **Abdulsalam Ba Sabaa** basabaa@ualberta.ca
  - **Leepakshi Bindra** leepaksh@ualberta.ca
  - **Jingwei Chen** jingwei5@ualberta.ca
  - **G M Mashrur E Elahi** eelahi@ualberta.ca
  - **Mehdi Faraji** faraji@ualberta.ca
  - **Amir Ahmad Habibi** amirahma@ualberta.ca
  - **Ashley Herman** aeherman@ualberta.ca
  - **Qingfeng Lan** qlan3@ualberta.ca
  - **Tianhua Li** tianhua@ualberta.ca
  - **Moein Owhadi Kareshk** owhadika@ualberta.ca
  - **Saeed Sarabchi** sarabchi@ualberta.ca
  - **Mennatullah Siam** mennatul@ualberta.ca
  - **Victor Silva** vsilva@ualberta.ca
  - **Joshua Sirota** sirota@ualberta.ca
  - **Alexandre Trudeau** trudeau1@ualberta.ca
  - **Jake Tuero** tuero@ualberta.ca
  - **Parnian Yousefi** pyousefi@ualberta.ca
  - **Nazanin Y. Khameneh** nyousefz@ualberta.ca
  - **Mahdi Zafarmand** zafarman@ualberta.ca

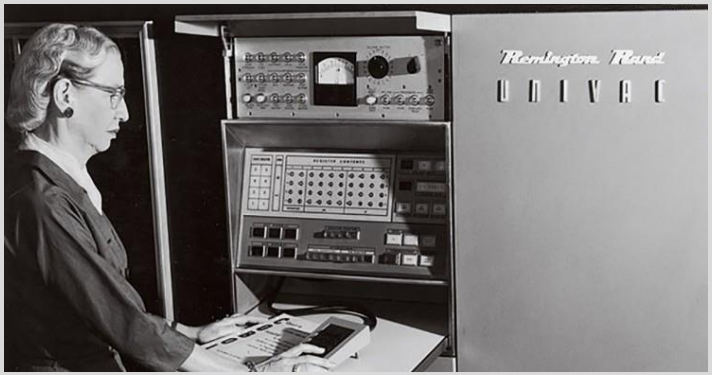
3 hour labs per week in CSC 153 & 129

Computing Science Class 1980s

26 students: 14 girls and 12 boys

Evelyn Berezin in 1976, when she was president and founder (1969) of the Redactron Corporation, with Data Secretary, the first computerized word processor, which she designed and marketed.
CreditCreditBarton Silverman/The New York Times

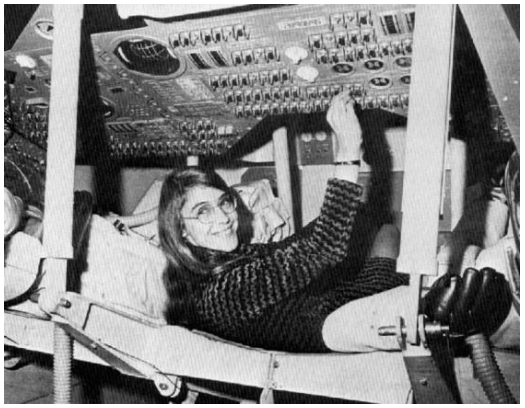
She died in Manhatan December 8th, 2018. She was 93.



Grace Hopper laid the foundation for the Information Age
Suggested the use of English language words as instructions rather than opaque symbols and numbers.
Developed the first compiler in 1949.
Introduced the notion of code reuse.

Women and Programming

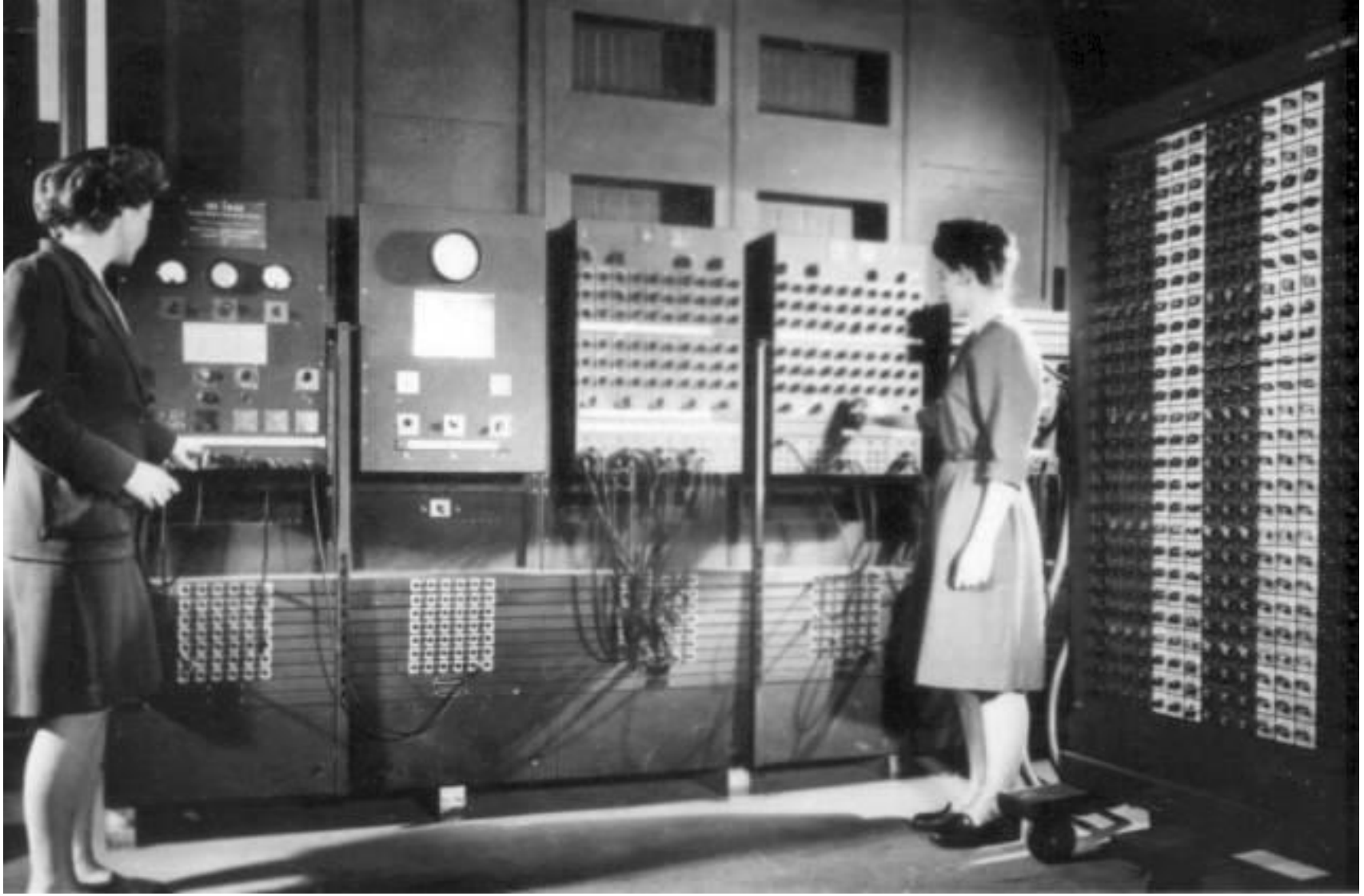
1969: standing beside listings of the software developed by the team she was in charge of.



Coined the term "Software Engineering"

Margaret Hamilton, the Engineer who took the Apollo to the Moon

Thanks to her Neil Armstrong and Buzz Aldrin could take a walk on the Moon.



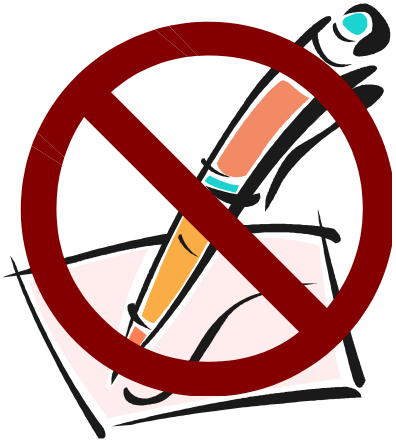
Electronic Numerical Integrator and Computer (ENIAC), the world's first general-purpose electronic computer. Feb 15 (2011) Eniac Day (Philadelphia) University of Pennsylvania.

Some Rules

Cellphones should be turned off in Class/Lab
No cellphones or PDAs in exams

No recording (audio or video) without explicit authorization by instructor.

Laptops or Tablets allowed exclusively for taking notes. **No Facebooking, emailing, etc.**



Doctor's note not required to justify absence.
There is a form to fill in (statutory declaration).
Instructor would have the final say.

Deferral of term work is a privilege and not a right

Prerequisites

- The course draws heavily on CMPUT 174

Computing Science

Computing science is not about computers.
Computing science is about **Problem Solving.**
Compute**r** science is a misnomer. Think astronomer and telescope

Computing science is about developing a step-by-step list of instructions to solve an instance of a given problem.
The ordered list of instructions is called an **Algorithm.**

For a given problem there could be many possible algorithms to solve it. Some problems are not solvable (i.e. don't have a known solution). They are not computable.

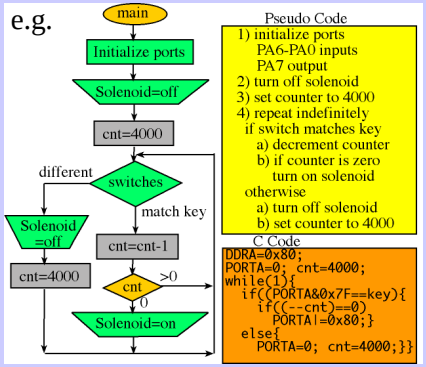
Algorithms are converted into programs for the computers to interpret them (i.e. execute them).

Prerequisites

- The course draws heavily on CMPUT 174

Programming

Programming = Data + Algorithms



Without an algorithm there can be no program

Thinking in pseudo-code
Way to go

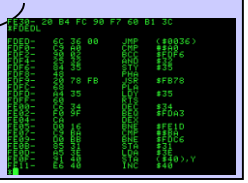
Data Structures + Pseudo-code Algorithms

Thinking in Python
Not recommended

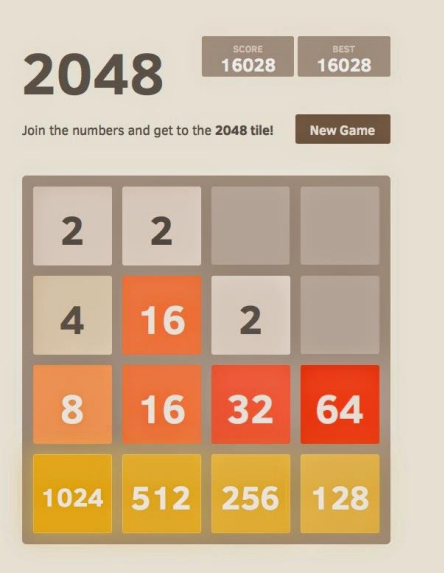
Translated into a programming language:
e.g. Python, Java, C++, Perl, Pascal, Basic, etc.

Thinking in Machine code
Impossible

Compiled or interpreted into machine code



Example for illustration



Data + Algorithms

4 x 4 matrix
Direction choice

- Add one new 2 or 4 in random empty cell
- Get direction; Move cells based on direction
- If two adjacent cells have the same value, Add them up in one cell in the same direction
- Stop when one cell has 2048 or all cells full

Data Structures + Pseudo-code Algorithms

A list of 4 lists of 4 elements
A characters

M=[[2,2,0,0],[4,16,2,0],[8,16,32,64],[1024,512,256,128]]
ch= “a”

- Add a 2 or 4 in any cell with a 0 (random)
- Get direction
- If direction is left
    for each line
        move all cells to the left
            if two cells are identical
                add them up in one cell

…

Prerequisites

- The course draws heavily on CMPUT 174

Computers are dumb

A program is a list of instructions that instruct the computer what to do. The computer does exactly what it was instructed to do; nothing more, nothing less.

Computers take instructions literally as they are given to them. **If the computer doesn't do what you expect it to do that means the algorithm or the program is wrong.**

Computers can repeat things again and again, and are usually fast

Computers don't remember anything unless you instruct them to memorize information

Course Structure

- 3 lecture hours + 3 lab hours per week + online videos for each week
- Topics
  - Catching exceptions, iterators, inheritance
  - Data structures: containers, lists, stacks, queues, trees, hash tables
  - Recursion
  - Sorting and searching
  - Efficiency of Algorithms - Complexity analysis: time and space

Learning Objectives

By the end of this course, students should be able to:

A. Use a range of predefined python data structures including strings, sets, lists, nested lists (e.g. 2D lists) and dictionaries in a program.
B. Write python programs that handle exceptions including throwing and catching exceptions, as well as assertions.
C. Write python programs which use files for reading input and writing output.
D. Generate and format strings using python for output on a screen or file.
E. Analyse algorithms/programs and compare their efficiency in terms of time and space complexity.
F. Manually trace the execution of a python program and explain the expected outcome.
G. Debug a python program by tracing control flow and variable states using a debugger.
H. Extend a python class by exploiting inheritance, overriding its methods and augmenting its interface.
I. Design new Abstract Data Types (ADTs) when the need arises and implement the corresponding data structures in python.
J. Appropriately employ encapsulation and information hiding when writing a program in python
K. Use abstraction to express object behaviour using a class interface and implement it using a class.
L. Use recursion in solving iterative problems and traversing specific data structures.

Values Objectives

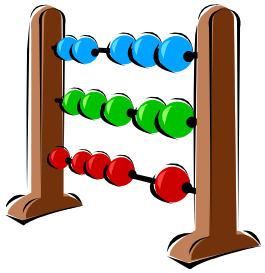
- Value the importance of devising a well-designed algorithm prior to coding a program.
- Appreciate the benefits and importance of isolation testing (unit testing) prior to testing and executing a full program.
- Acknowledge that different solutions to the same problem may diverge significantly in terms of solution quality, time efficiency and space requirements even if the output is the same.

Course Web Site

- Everything is on e-class – (with CCID login)
- https://eclass.srv.ualberta.ca/course/view.php?id=49646
- Slides, Videos, Labs, assignments, resources, etc.

Go to:
https://eclass.srv.ualberta.ca/my/

Final Mark



No Final Exam but 3 Midterms

- Breakdown:

Lab Exercises (10):	10%	35%
Assignments (4):	20%	
Video online assessment	5%	
In-class Assessment	5%	65%
Midterm 1 (Feb 15th):	15%	
Midterm 2 (Mar 13th):	20%	
Midterm 3 (Apr 10th):	25%	

Last day of classes

In-Class Assessment

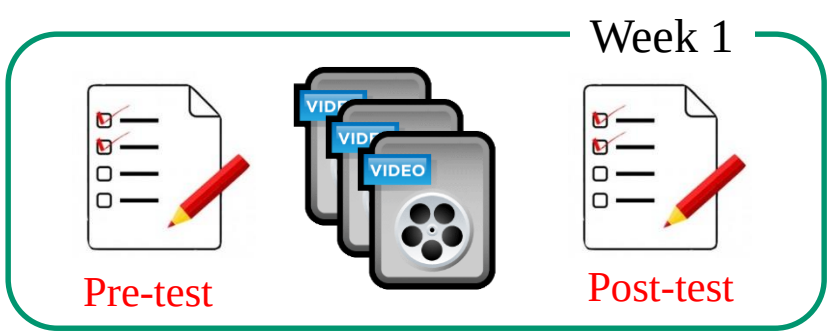
To encourage attendance and regular revision

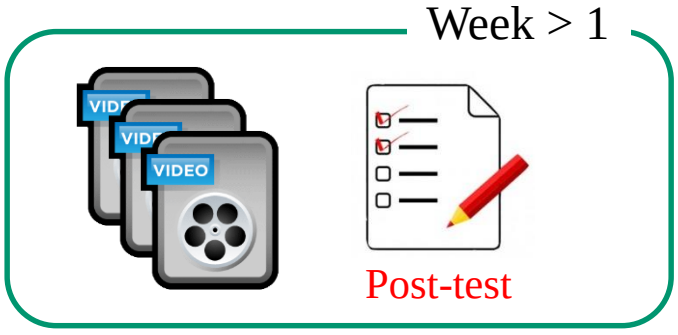
- 5% of the overall grade
- In each lecture, up to 3 names will be randomly selected and asked a simple question about the previous lecture.
- If student absent → -3/10
- Student oral answer marked out of 3/10
- Student might be called up to 3 times in the semester
- If student is never called → full mark.

Video Online Assessment

To encourage preparation before class

- 5% of the overall grade
- Each week.
- One or more short video lectures to see
- Short Post Test (with end-date)
- First week: Pre-test → no mark.
  Post-test marked.

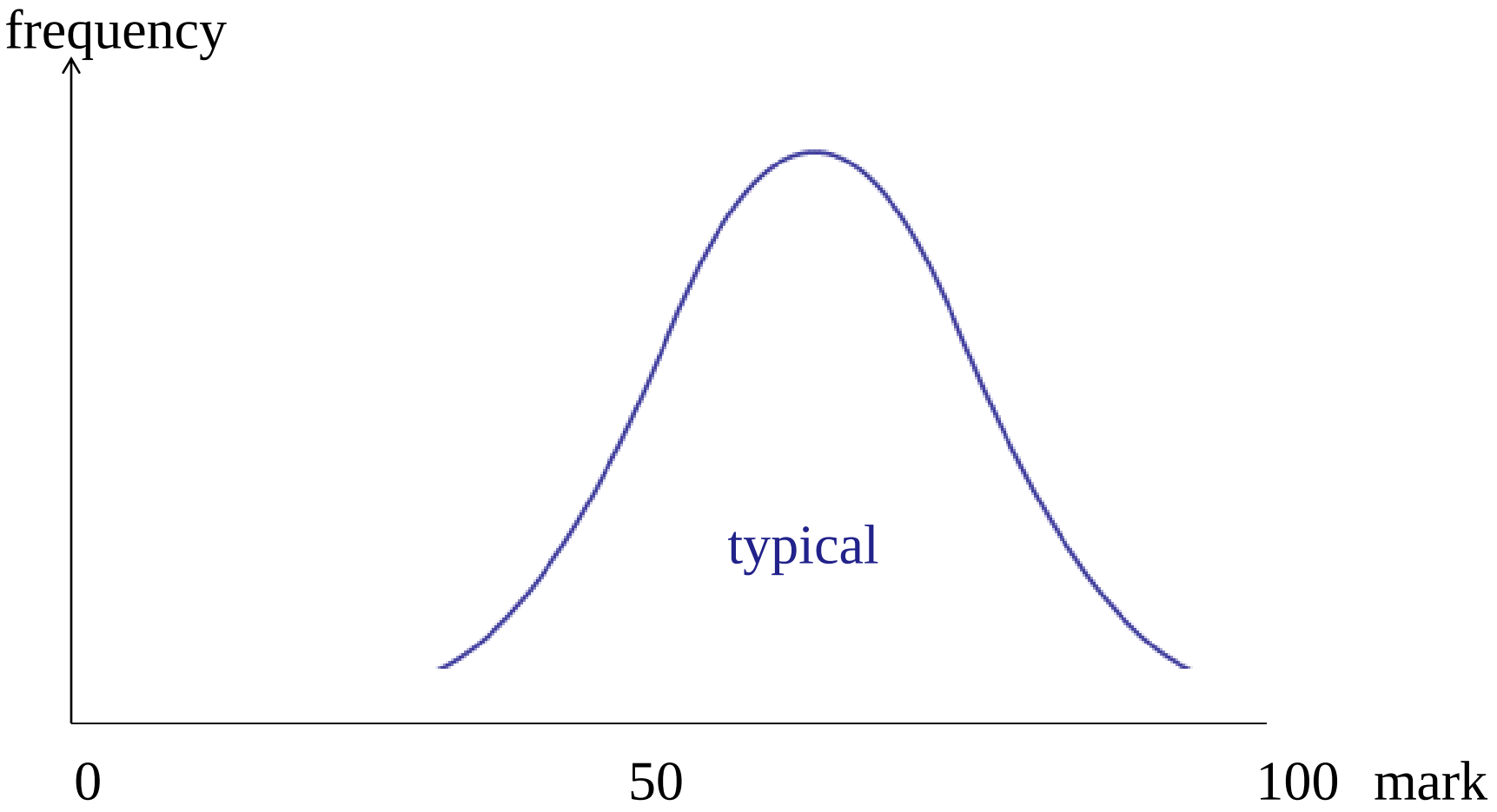




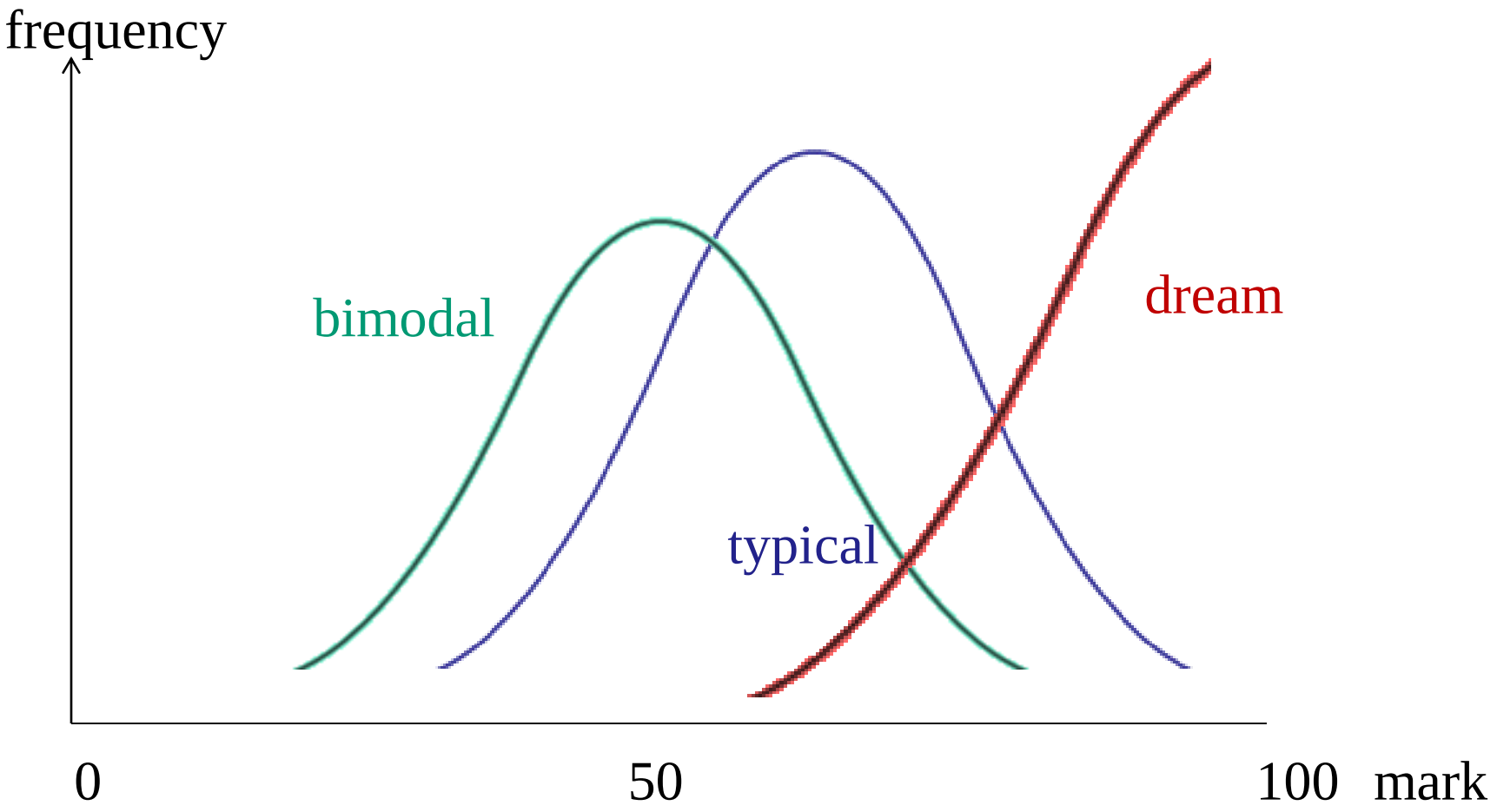
Final Grade Assignment

- No fixed breakpoints (e.g., 50% does not necessarily mean pass)
- Impossible to predict your letter grade (A, B, C, …) until everybody's marks are in (including the last midterm marks)
- I do not necessarily follow a particular curve.
- Marks reflect mastery of course material (instructor's judgment)

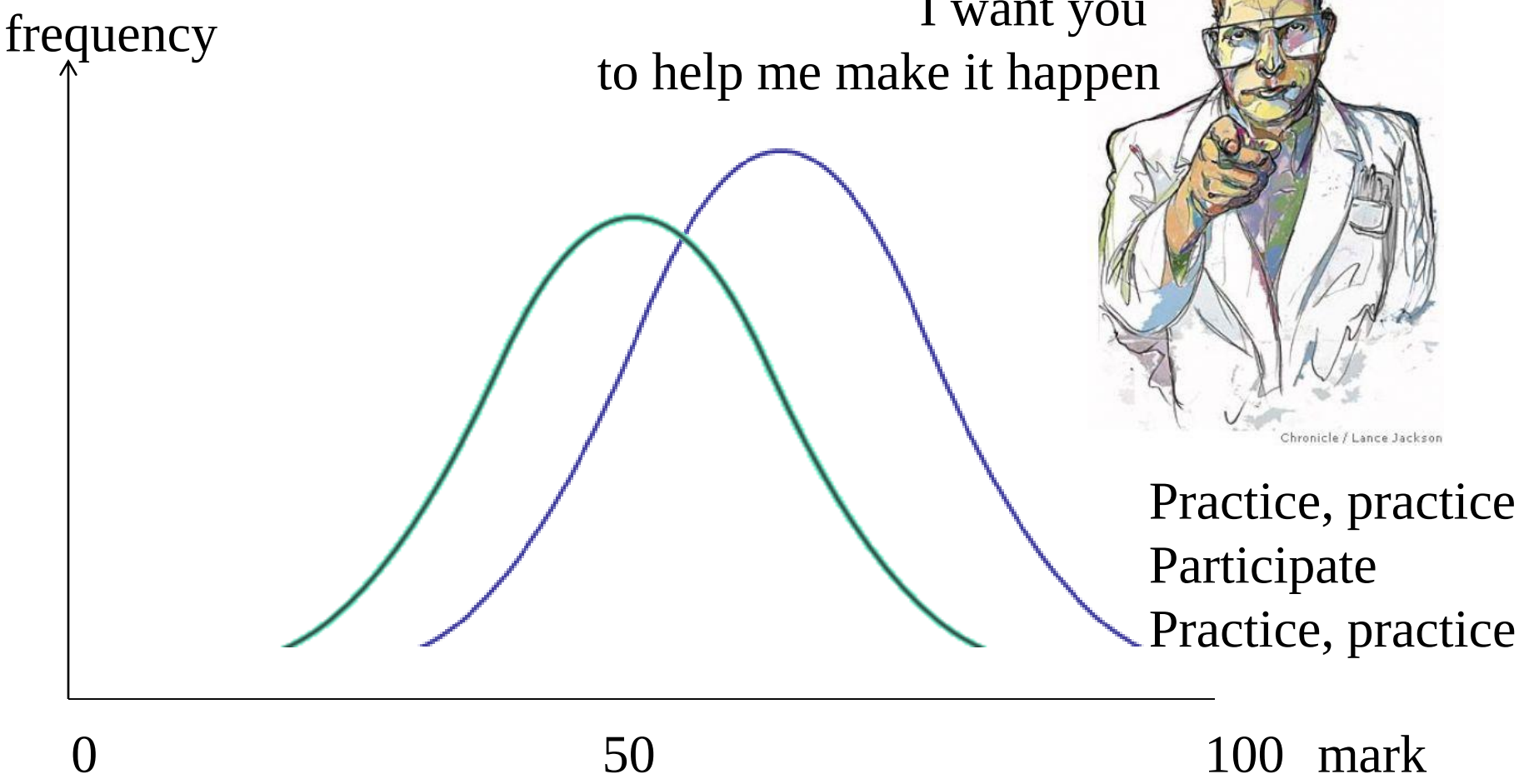
Mark distribution



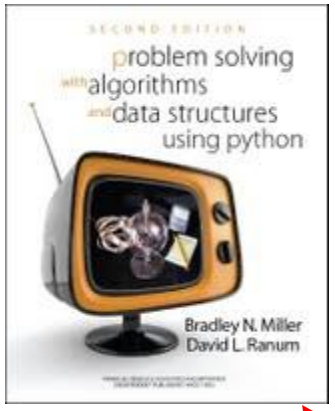
Mark distribution



Mark distribution

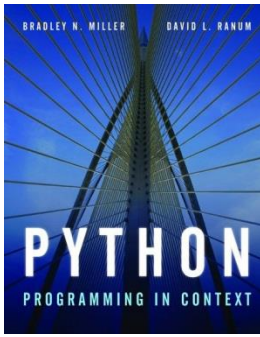


No Imposed Textbook



This semester available on eclass

- Problem Solving with Algorithms and Data Structures using Python
- By Brad Miller and David Ranum
- Any text that covers the list of topics is recommended (there are many on-line tutorials)
- Other 175 sections have used another book (Python programming in context) by the same authors



Slides

- The instruction is based mostly on slides, online videos and lab exercises
- Slides are available in PDF off the website. Each instructor may have different slides.
- Please print them before the lecture and bring them with you to class to take notes
- Slides used in class could be slightly updated with examples (but not necessarily put on-line)

Attendance

P.S.

I distinctly remember that you were often disappointed with the number of people who showed up to class daily since not all enrolled students showed up regularly. Maybe tell them that 3 of your students (myself, [redacted], and [redacted]) were successful because we attended class :P.

Labs

- Labs start next week (January 14th)
- All guidelines and news are on the web
  - Please read carefully to avoid unpleasant surprises
- Please check the News Forum often

Missing Components

- The detailed policy is described on the web
- If you have to miss something (e.g., a lab) please make prior arrangements
- If you missed something unexpectedly then you can apply for an EA
  - Not automatic, need documentation
  - Tight timeline
  - Not given for some components
  - No make-up exam ➔ weight transfer

Exams

- Need your ONECard
- Need to be on the class list
- Closed book (unless explicitly stated)
- More legal details on the web

Collaboration

- Collaboration allowed on:
  - Challenge problems
  - Assignments
  - Lab exercises

- Collaboration needs to be acknowledged explicitly in the work you hand in:
  - // I collaborated with P. Brosnan on this problem. Method hook-up() in
  - // class C007 was written jointly with M. Yeoh.
- Each student must be able to explain everything handed in

Submissions are individual. Each student is responsible for his/her submission.

No Collaboration

- On Midterms

Not just for midterms, but any assignment, lab exercise etc.

Plagiarism.

Work submitted by a student that is the work of another student or any other person, is considered plagiarism. Cases of plagiarism are immediately referred to the Dean, who determines what course of action is appropriate.

- Violations of this policy will be forwarded to the Dean's office
- A long, unpleasant process
- Permanent ramifications

Code Of Student Behaviour

- …is available on the web
- Please peruse it carefully
- Contact me if in doubt

Course Schedule

(Tentative, subject to changes)

There are 14 weeks from January 7th to April 10th

Attendance is HIGHLY recommended
There are weekly exercises to be done in the lab.

Midterm 1 **week 6 (Friday February 15th 2019)**
Midterm 2 **week 10 (Wednesday March 13th 2019)**

Midterm 3 **week 14 (Wednesday April 10th 2019)**

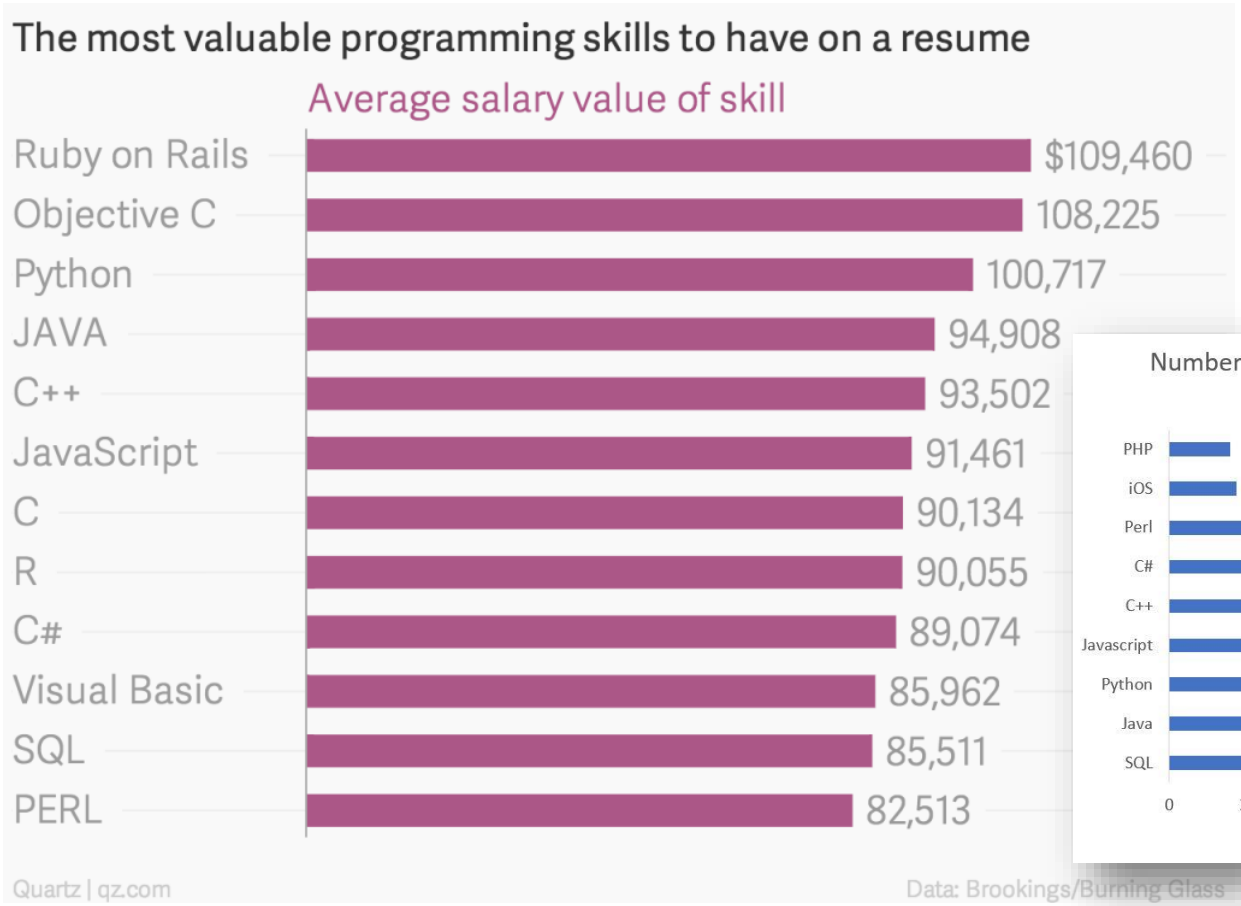
No class from February 18th to 22th (Reading week)

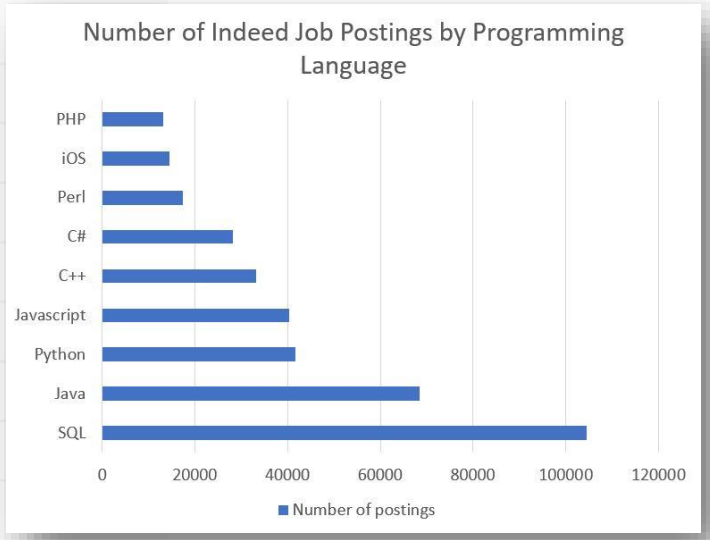
General Schedule per week

Week 1	Week 5	Week 10	Week 14
M: Syllabus + Intro	M: Handling Exceptions	M: Search	M: Revision
W: Python 101	W: Handling Exceptions	W: **Midterm 2**	W: **Midterm 3**
F: Python 101	F: Handling Exceptions	F: Sorting	
Week 2	Week 6	Week 11	
M: Enciphering	M: Queue	M: Sorting	
W: Enciphering/Hangman	W: Queue	W: Sorting	
F: Hangman	F: **Midterm 1**	F: Sorting	
Week 3	Week 7: **Reading Week**	Week 12	
M: Algorithm Analysis	Week 8	M: Trees	
W: Algorithm Analysis	M: Reverse Polish Notation	W: Trees	
F: Algorithm Analysis	W: Reverse Polish Notation	F: Trees	
Week 4	F: Linked Lists	Week 13	
M: ADT	Week 9	M: Binary search trees	
W: Stack	M: Linked Lists	W: Hash tables	
F: Stack	W: Doubly-Linked Lists	F: Hash tables	
	F: Recursion		

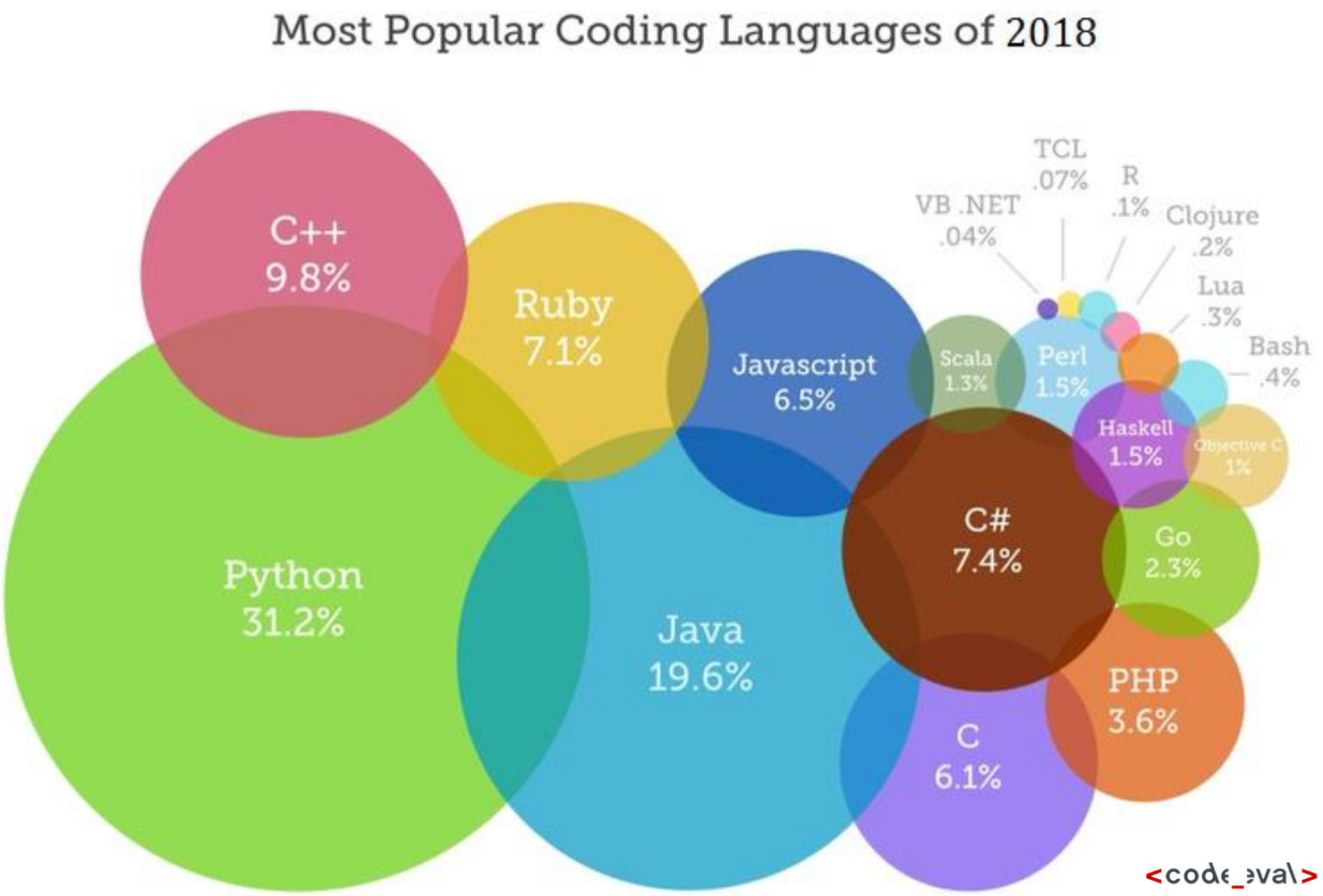
Why Python

About 25 years old; became lately very popular; easy language and easy to learn.

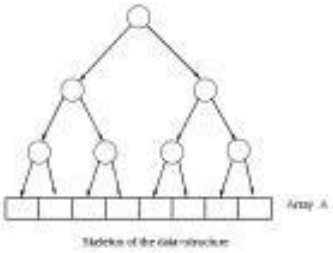


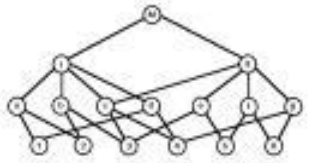


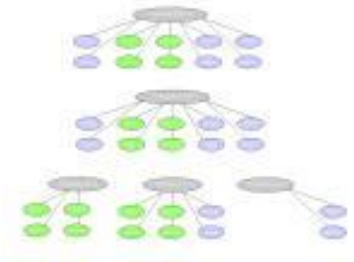
Why Python

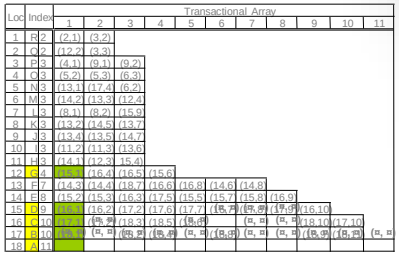

Most Popular Coding Languages of 2018
C++
9.8%
Ruby
7.1%
Javascript
6.5%
TCL
.07%
R
.1%
VB .NET
.04%
Clojure
.2%
Lua
.3%
Bash
.4%
Scala
1.3%
Perl
1.5%
Haskell
1.5%
Objective C
1%
C#
7.4%
Go
2.3%
Python
31.2%
Java
19.6%
PHP
3.6%
C
6.1%
<code_eval>

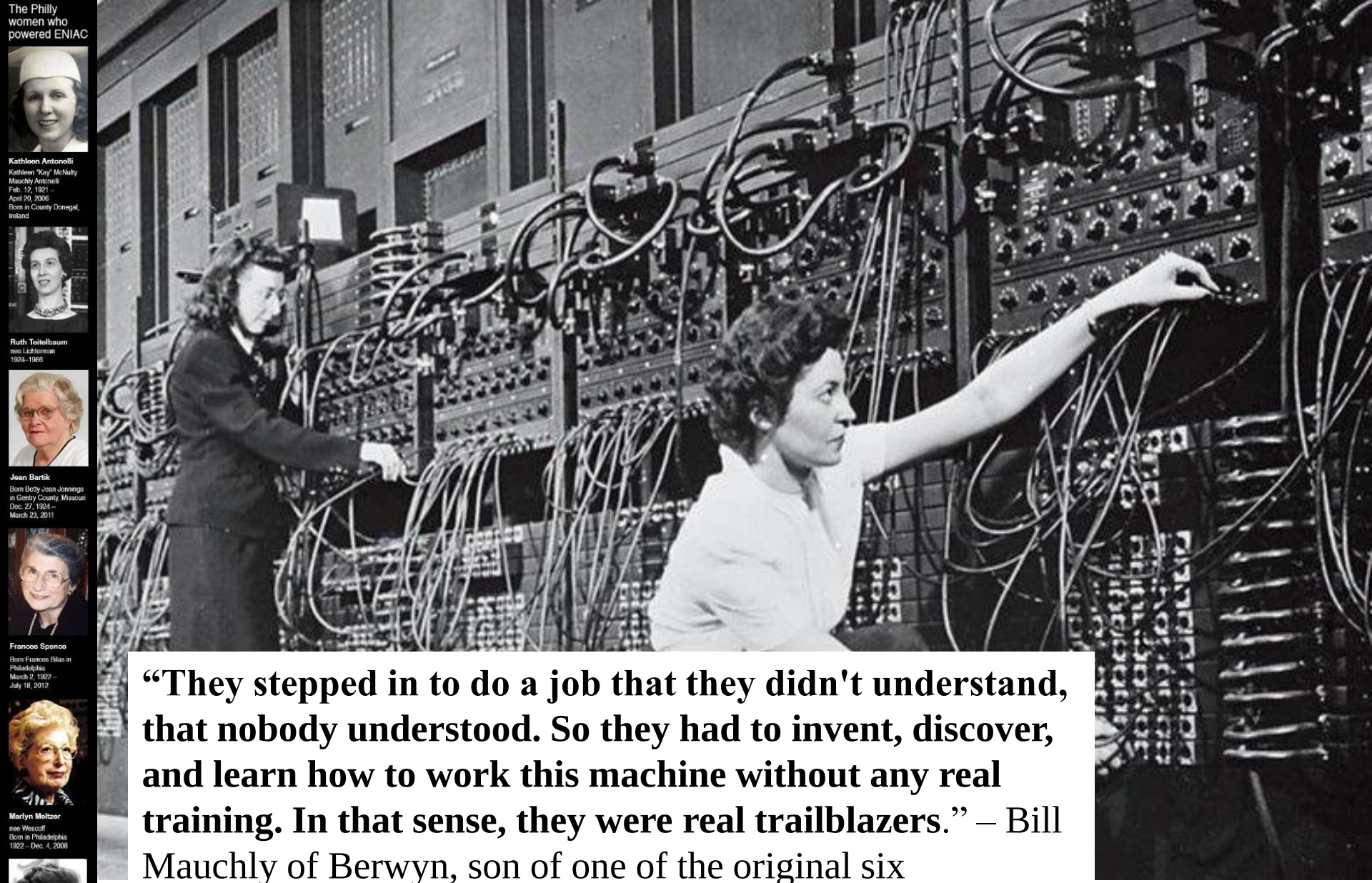
Questions











"They stepped in to do a job that they didn't understand, that nobody understood. So they had to invent, discover, and learn how to work this machine without any real training. In that sense, they were real trailblazers." – Bill Mauchly of Berwyn, son of one of the original six programmers