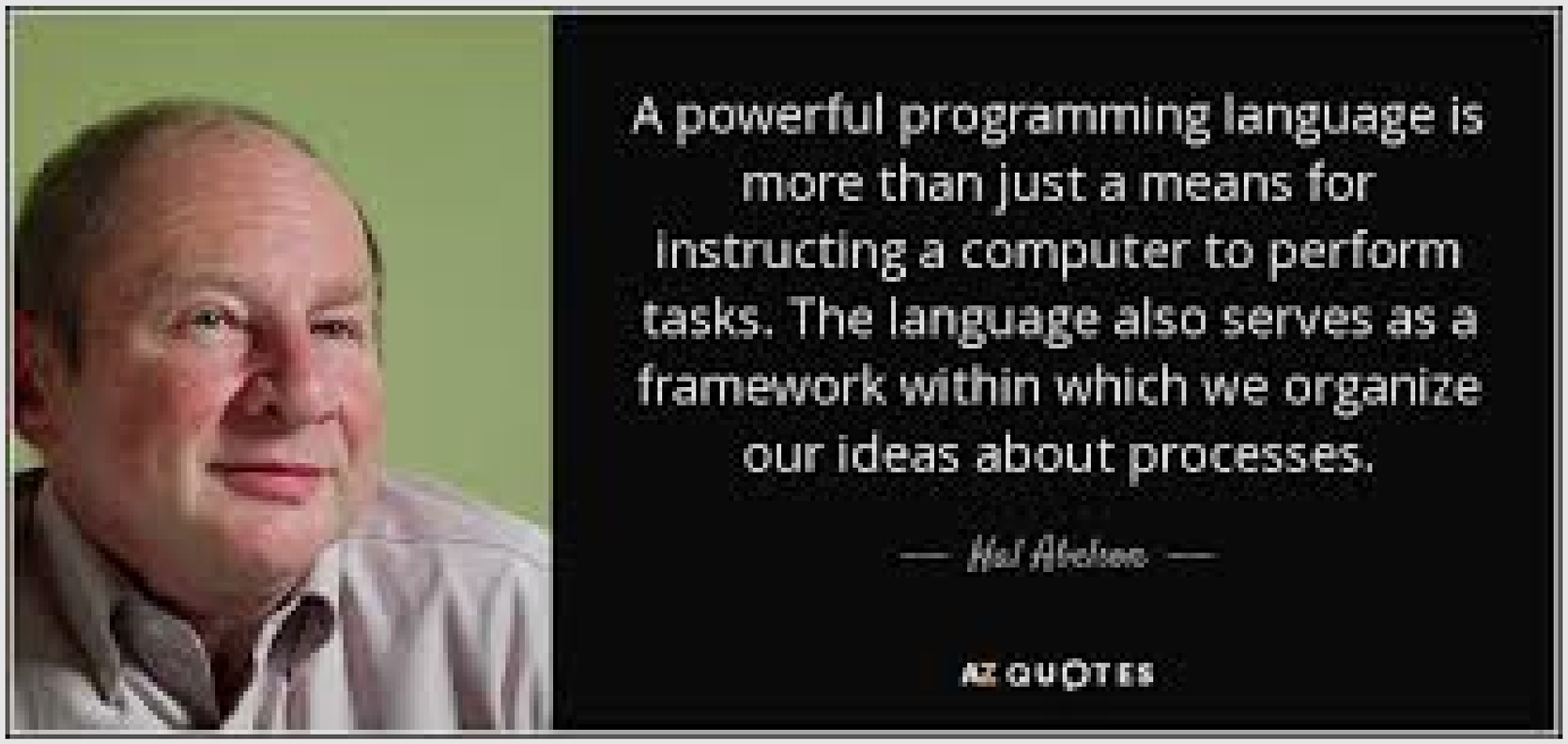
A gentle introduction to Programming Languages

Table of Contents

- 15. Edit time
- 16. Compile time
- 17. Distribution time
- 18. Installation time
- 19. Link time
- 20. Load time
- 21. Run time
- 22. Programming paradigms
- 23. Imperative
- 24. Procedural
- 25. Object Oriented
- 26. Functional Programming
- 27. Functional Programming II

1 Programming languages help us think



2 History

First computers had no languages exept machine code

- Euclid's algorithm for GCD in machine code

3 Machine code

```
55 89 e5 53  83 ec 04 83  e4 f0 e8 31  00 00 00 89  c3 e8 2a 00
00 00 39 c3  74 10 8d b6  00 00 00 00  39 c3 7e 13  29 c3 39 c3
75 f6 89 1c  24 e8 6e 00  00 00 8b 5d  fc c9 c3 29  d8 eb eb 90
```

4 Assembler code

```
        pushl   %ebp
        movl    %esp, %ebp
        pushl   %ebx
        subl    $4, %esp
        andl    $-16, %esp
        call    getint
        movl    %eax, %ebx
        call    getint
        cmpl    %eax, %ebx
        je      C
A:      cmpl    %eax, %ebx
        jle     D
        subl    %eax, %ebx
B:      cmpl    %eax, %ebx
        jne     A
C:      movl    %ebx, (%esp)
        call    putint
        movl    -4(%ebp), %ebx
        leave
        ret
D:      subl    %ebx, %eax
        jmp     B
```

5 Limitations

- Each language depended on the machine
- So detail oriented that it was hard to keep track of

6 Desire for machine independence

- Invention of Fortran
- Literally "Formula Translation"
- First compiled language

7 Definition of a compiler

- A compiler is a piece of software that translates a higher level language into machine code.

8 Limitations of compiled languages

- Humans could always write programs that beat the compiler

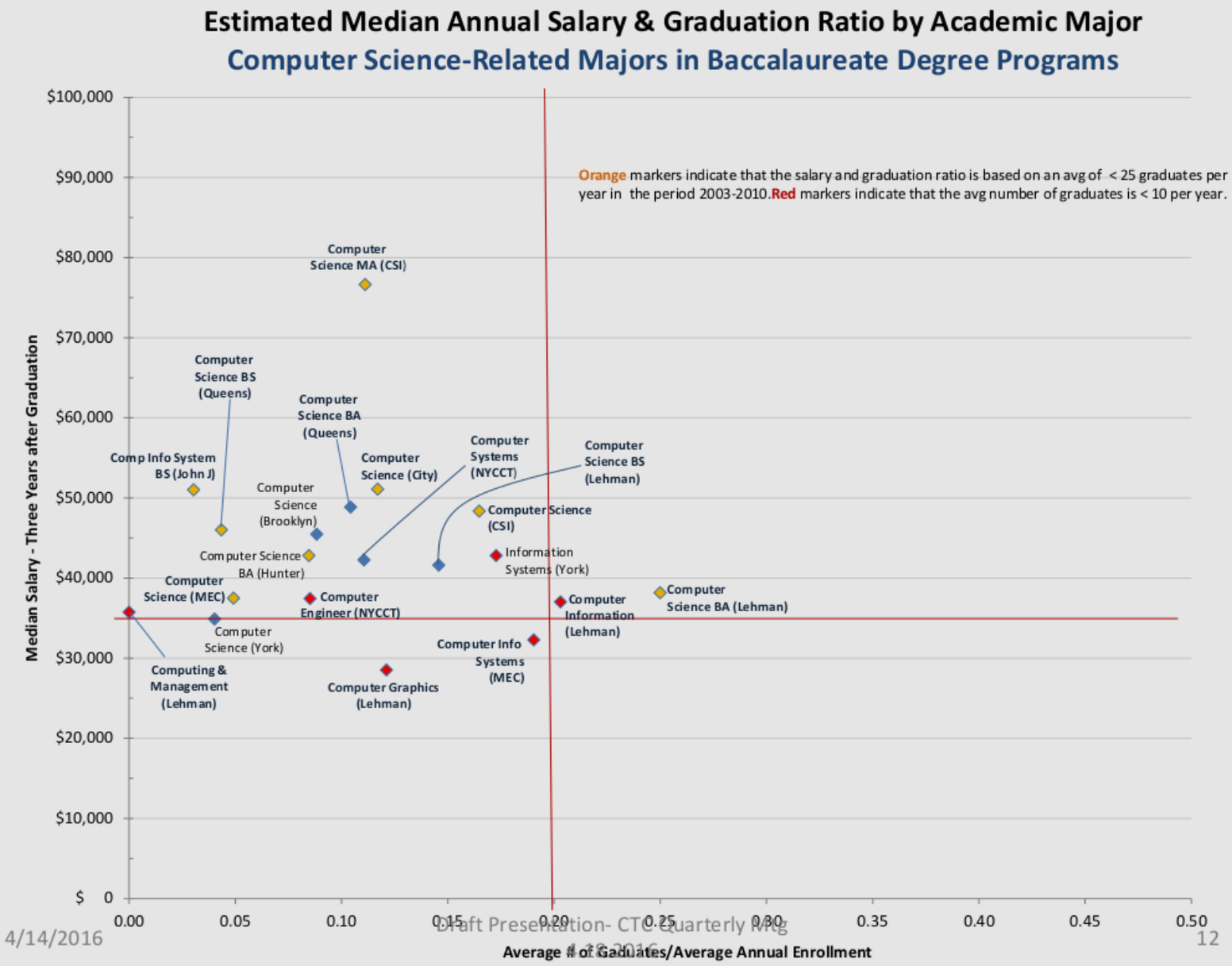
9 Now compilers always beat humans

- Technology improved and compiled code is either better or no worse than human optimized code.

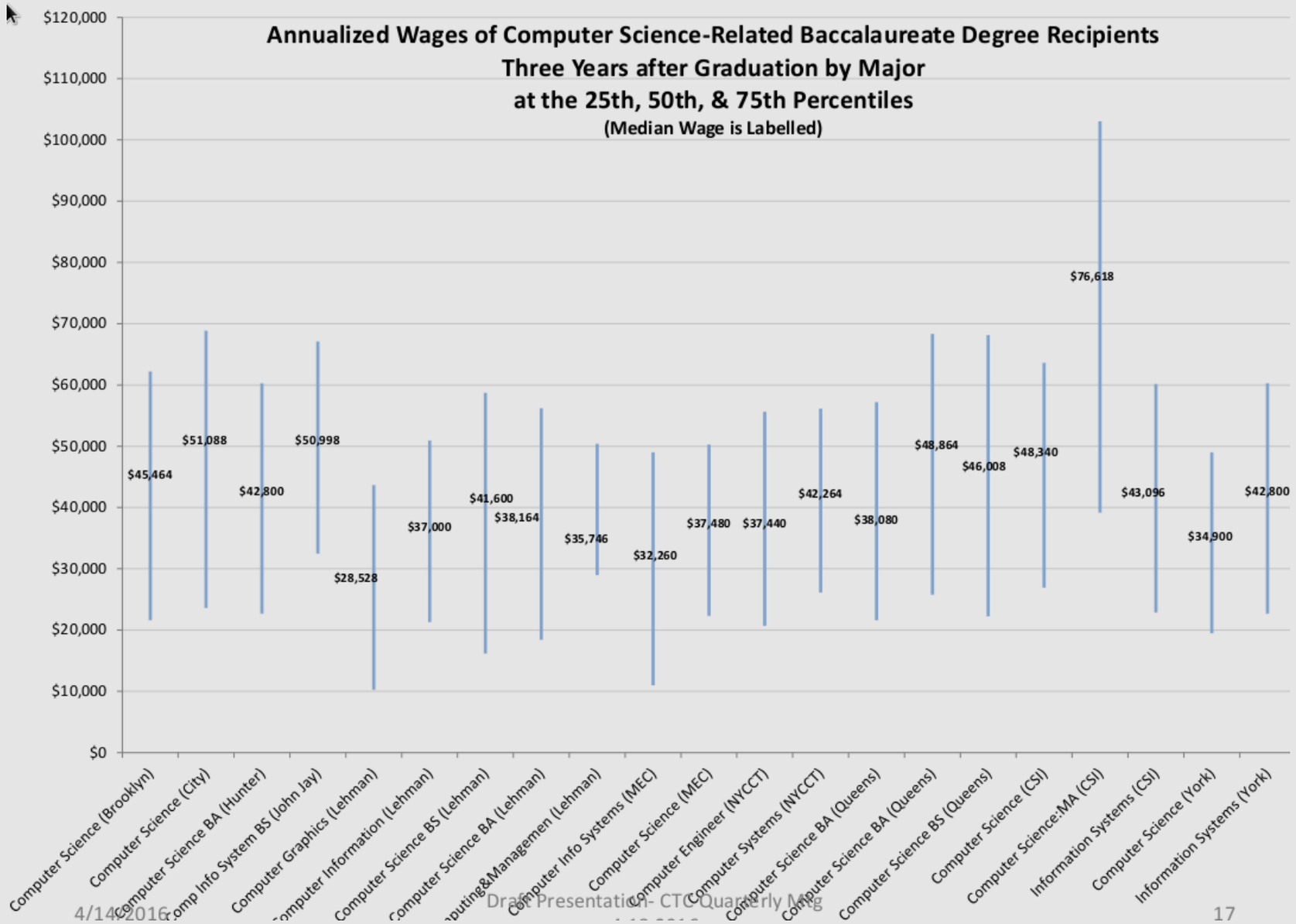
10 Why is all of this important to you

- BLS Statistics
- 2019 median salary is $107,510
- Number of jobs: 1,469,200
- Job growth: 22% (much faster than average)

11 Salaries vs Graduation



12 Salaries Distribution



13 Languages developers and employers want

- The most important job is to know

http://stackoverflow.com/research/developer-survey-2016#technology-most-loved-dreaded-and-wanted

14 Program lifecycle phase

- Edit time
- Compile time
- Distribution time
- Installation time
- Link time
- Load time
- Run time

15 Edit time

- This is while we are editing the program. This is the phase that version control allows us to collaborate in.

16 Compile time

- This is the phase when our program is being translated to machine code. This is when a type system may throw errors.

17 Distribution time

- The time that it takes to distribute a program. Think of downloading from a git repo, CRAN, Ubuntu or Debian repository.

18 Installation time

- The time it takes to install the program on the user's system.

19 Link time

- The time it takes to connect together any dependencies.

20 Load time

- The time that it takes for a stored image to be placed in memory

21 Run time

- The time the program is executing.

22 Programming paradigms

- Imperative
- Procedural
- Object Oriented
- Functional

23 Imperative

In computer science, imperative programming is a programming paradigm that uses statements that change a program's state. In much the same way that the imperative mood in natural languages expresses commands, an imperative program consists of commands for the computer to perform. Imperative programming focuses on describing how a program operates.

24 Procedural

Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call. Procedures, also known as routines, subroutines, or functions (not to be confused with mathematical functions), simply contain a series of computational steps to be carried out. Any given procedure might be called at any point during a program's execution, including by other procedures or itself.

25 Object Oriented

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another.

26 Functional Programming

In computer science, functional programming treats computation as the evaluation of mathematical functions and avoids changing-state and mutable data. It is a declarative programming paradigm, which means programming is done with expressions or declarations instead of statements.

27 Functional Programming II

In functional code, the output value of a function depends only on the arguments that are input to the function, so calling a function f twice with the same value for an argument x will produce the same result f(x) each time.

Created by Evan Misshula.