Lecture #1 | 프로그래밍과 python 소개

SE213 프로그래밍 (2018)

오늘 다룰 내용

- 강좌 소개 및 운영에 관한 사항
- 프로그래밍 과목을 배우는 이유 혹은 기대할 것
- 컴퓨터 혹은 프로그래밍은 무엇인가?
- python소개와 기초 문법
  - 기본 자료형
  - 변수
  - python에서 입출력: `print()`, `input()`

프로그래밍(SE213) 강좌 소개

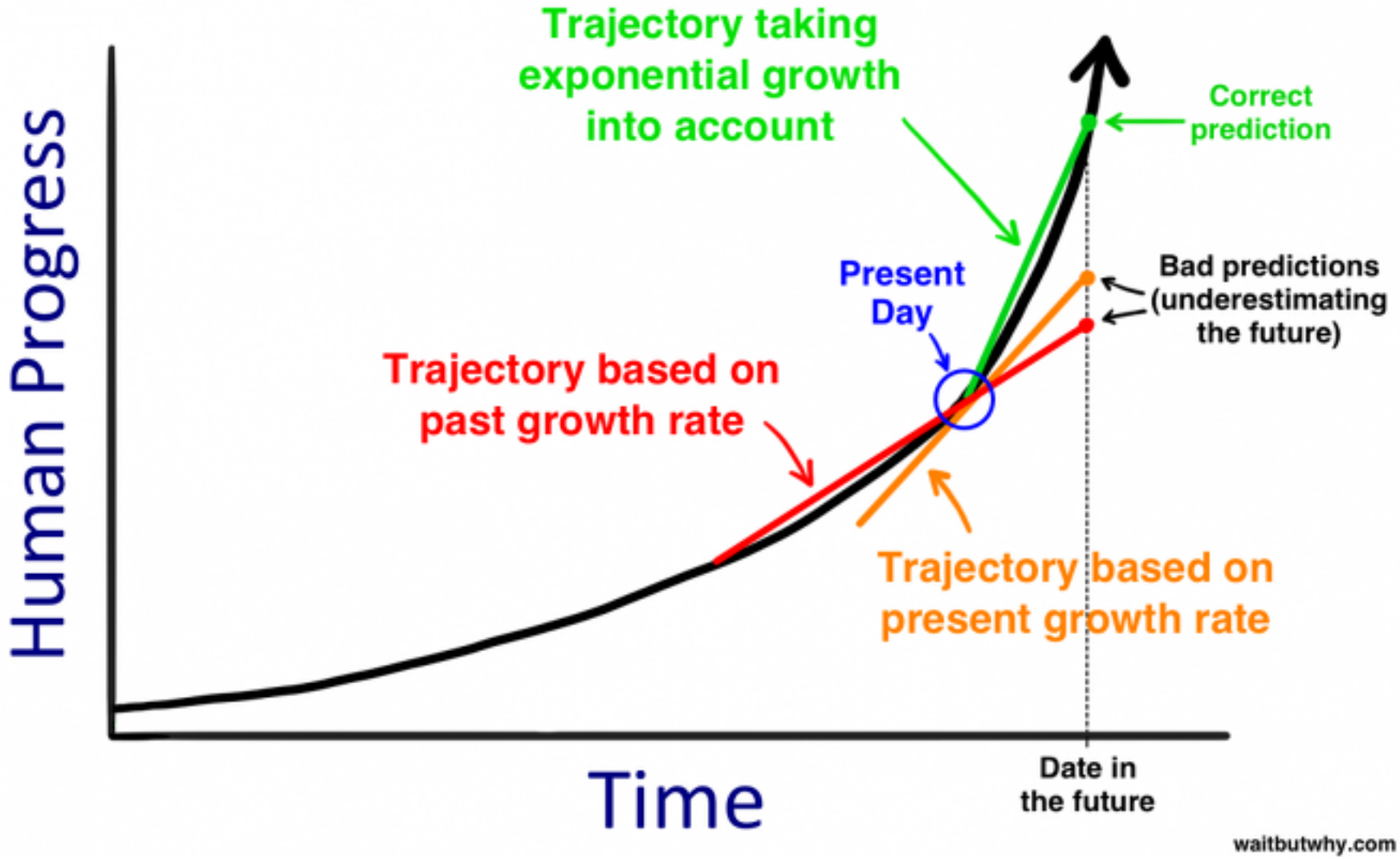
- 코스페이지
  - LMS: http://lms.dgist.ac.kr/
  - Course Home: https://github.com/chomg/se213_2018/
- 프로그래밍 환경
  - elice (https://dgist.elice.io/courses/263/): 프로그래밍 실습 및 과제 제출
  - pycharm (https://www.jetbrains.com/pycharm/): 일반적인 개발 환경
- 이론과 실습 모두 동시에 수강해야 함 (학점도 동일하게 부여)

프로그래밍 강좌의 대상과 목적

- 대상
  - DGIST 융복합대학 2학년
    → 대부분 프로그램 경험 없음
    & 컴퓨터/공학이 전공이 아닐 가능성이 높음

- 컴퓨터를 도구로 사용하는 방법
  - 여러 분야에서 활용되는 컴퓨터 작동 방식에 대한 이해를 높이기
  - 컴퓨터를 '도구'로 활용하는 능력을 키우기
    → 전공과 상관없이 컴퓨터는 폭넓게 사용됨
  - 컴퓨팅 사고(computational thinking)를 통해 문제해결력 높이기

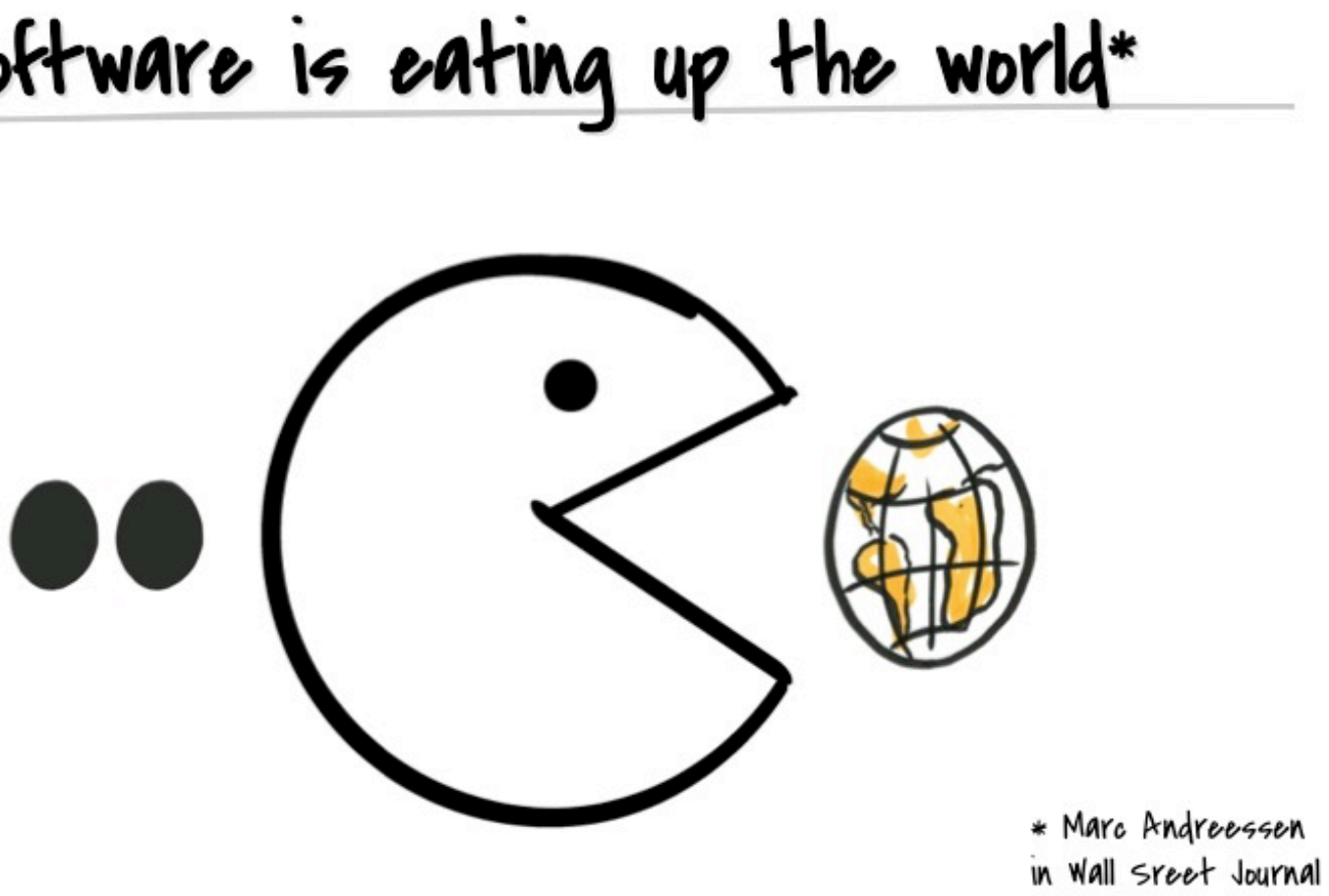
기술의 발전속도는 점점 빨라지고 있음



출처: http://waitbutwhy.com/2015/01/artificial-intelligence-revolution-1.html

특히 소프트웨어는 산업 전반에 끼치는 영향이 매우 큼

- 소프트웨어가 바꾸는 것
  - 생산성
  - 생각하는 방식
  - 일하는 방식
- 소프트웨어/기술 기반의 회사들이 기존 산업 대체
  - 보더스 v.s. 아마존 (서점)
  - 코닥 v.s. 인스타그램 (사진)
  - 콜택시 v.s. 카카오택시, 우버
  - …

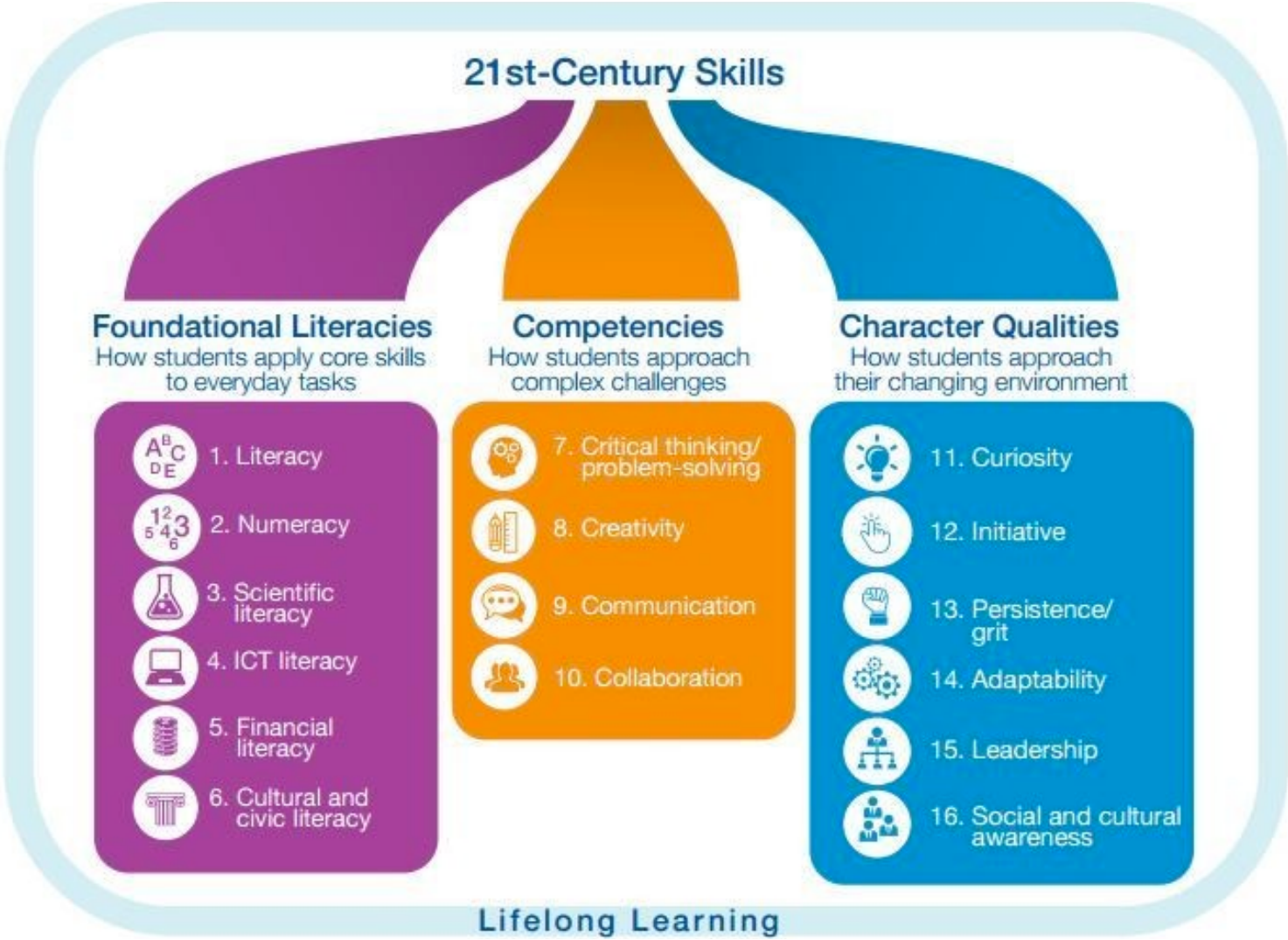


출처: Marc Andreessen on Why software is eating the world

21세기에는 한 분야의 지식보다 복합적 지식/사고력이 요구됨

- 기술의 발전은 기존에 없었던 복잡한 문제(problem)들을 만들게 될 것임

→ 복합적 지식과 사고력에 기반한 문제해결능력 (problem solving skills)이 요구됨



출저: [What are the 21st-century skills every student needs?, WEF]

과학/컴퓨터 교육의 목표는 지식과 함께 사고력을 키우는 것

"Science is more than a body of knowledge.
It is a way of thinking; a way of skeptically interrogating the universe with a fine understanding of human fallibility."

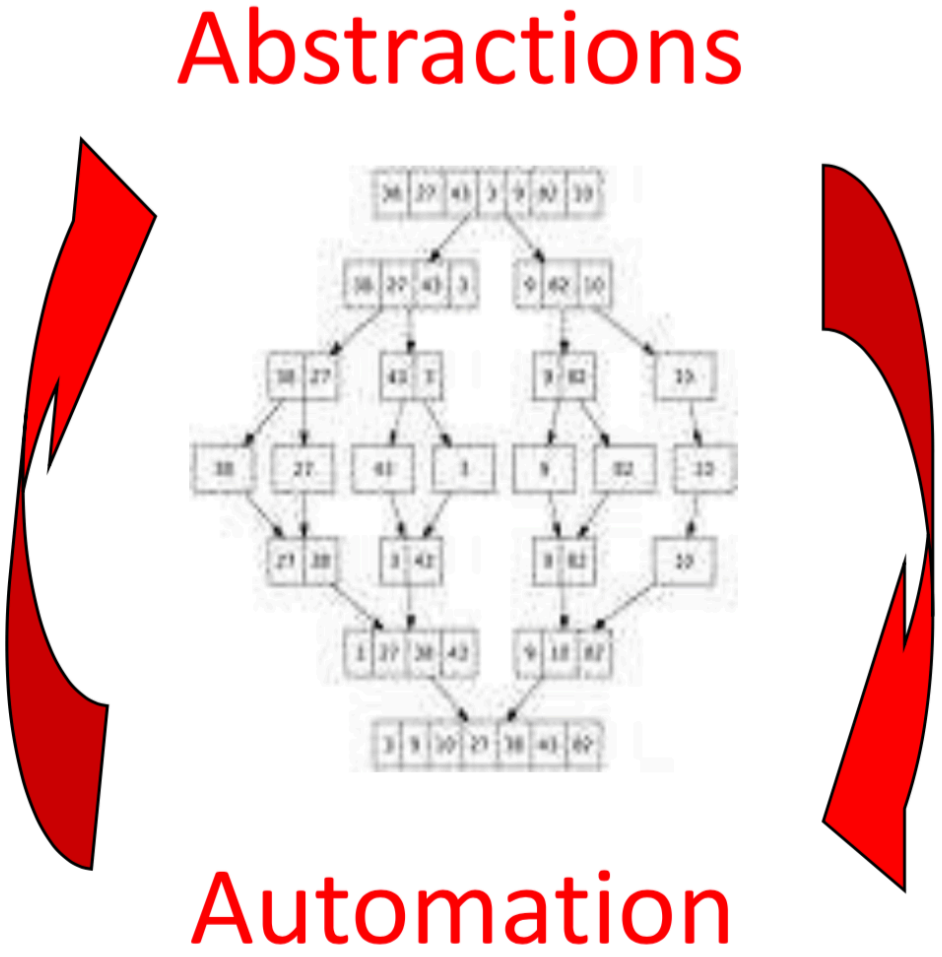
– Carl Sagan

"Everybody in this country should learn to program a computer, because it teaches you how to think"

– Steve Jobs

추상화를 통한 문제해결과 자동화가 컴퓨팅사고의 핵심

- 컴퓨팅 사고(Computational Thinking)의 정의들
  - 문제를 표현(formulate)하고, 그 답을 컴퓨터(사람 혹은 기계)가 효율적으로 처리할 수 있는 답으로 표현하는 사고 단계
  - 컴퓨터 과학의 기본적인 개념들을 활용하여 문제를 해결하고, 시스템을 설계하고, 사람의 행동을 이해하는 방법
  - 추상화를 활용하여, 기계를 이용한 자동화가 가능한 알고리즘을 만드는 문제해결 방법



출저: Jeannett M. Wing, Aman Yadav

분해, 패턴인식, 추상화, 알고리즘의 단계를 걸쳐 문제를 해결

분해 Decomposition	큰 문제를 해결 가능한 작은 문제로 나누는 것, 경우에 따라 여러 번 반복
패턴인식 Pattern recognition	(작게 나뉘어진) 주어진 문제에서 공통된 요소(패턴)을 찾음
추상화 Abstraction	문제 해결에 있어 중요한 공통된 부분만을 놔두고, 다른 구체적인 사항들을 없앰
알고리즘 Algorithms	어떤 입력에도 원하는 결과를 얻을 수 있도록 잘 정의된 연산 단계 예: 덧셈, 뺄셈, 곱셈, 나눗셈, …

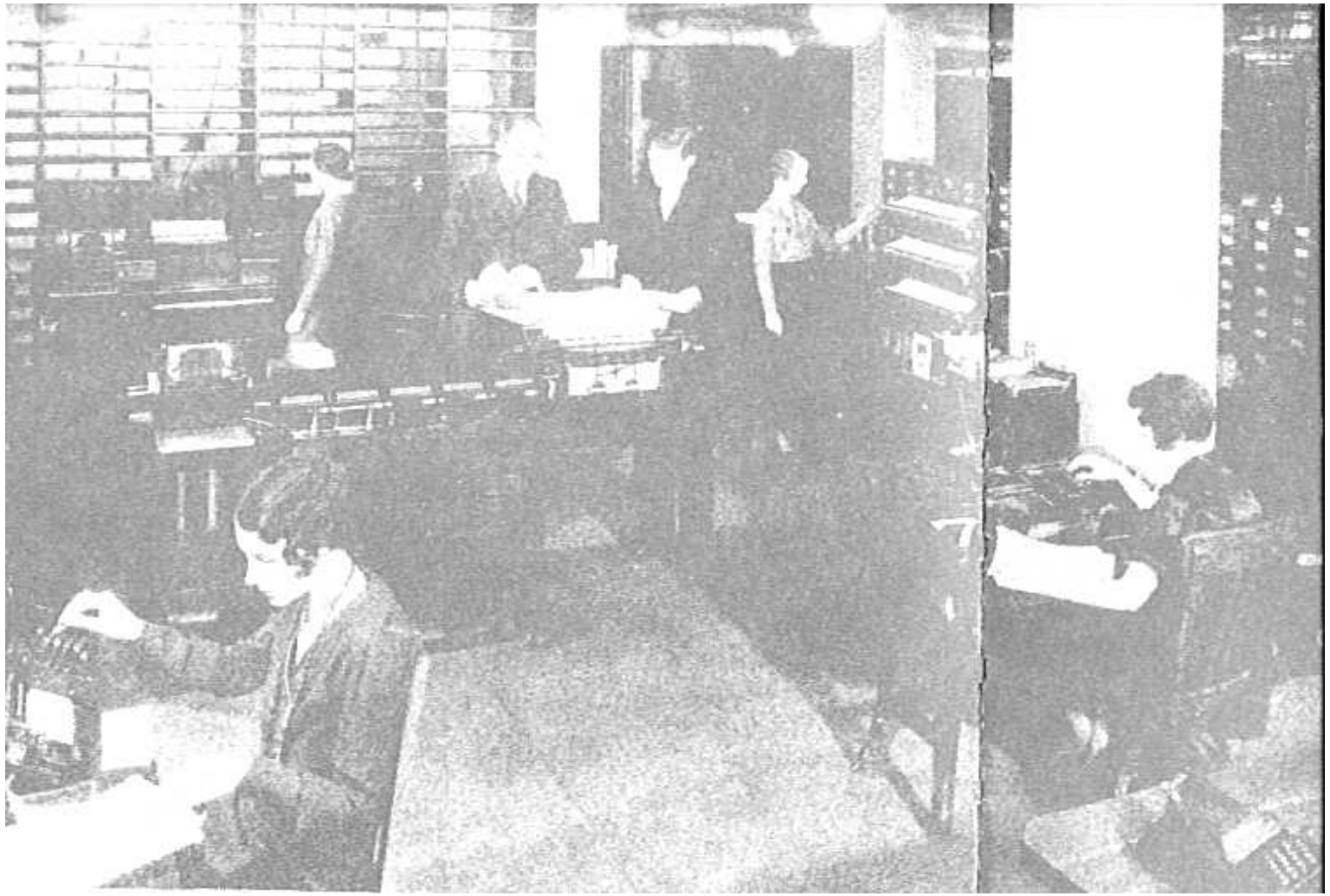
프로그래밍, 혹은 다른 분야를 잘 하기 위해 필요한 것들

- 관련된 분야의 지식 – 수학, 과학, 경영, …
- 학습 능력
- 질문 – 무엇이 중요한가, 왜 중요한가, 다른 문제와 공통점과 차이는 무엇인가, 어떻게 문제를 다르게 볼 수 있을까, …
- 호기심
- 영어 (예전보다 조금 덜 중요할 수도 있음…)

컴퓨터로 할 수 있는 것들...?

- 오버워치/롤/…
- 카톡/페이스북/트위터/인스타그램/…
- 동영상 시청
- 웹서핑
- 전자상거래
- 숙제
- 프로그래밍
- …

최초의 컴퓨터(computer)



출처: http://www.computersciencelab.com

컴퓨터가 하는 일?

- 연산: +, -, ×, ÷, …
  - 이것들의 조합으로 복잡한 연산을 수행

- 저장: 숫자 (기본적으로는 2진수, 0 or 1)
  - 문자열, 이미지, 소리, 동영상, …

- 입출력: 숫자, 문자, 그림, 소리, …

컴퓨터 과학/공학은 무엇인가?

- 학과 이름
  - 컴퓨터 공학과, 전산학과, 정보통신공학과, 정보통신융합전공, …
  - Computer Science, Computer Engineering,
    Information Science, Informatics,
    Computer Science & Engineering,
    Information and Communication Engineering, …
- 컴퓨터 과학/공학이란?
  - ‘정보’를 얻고, 표현하고, 처리하고, 저장하고, 통신하고, 접근하는 ‘방법론’의 가능성, 구조, 표현, 구현에 관한 학문
  - ‘지능Intelligence’을 구현하는 방법에 대한 학문

프로그래밍 언어(Programming Language)란?

- **필요성**: 컴퓨터와 사람이 커뮤니케이션 방식이 다름
  - 컴퓨터가 이해하는 것: 0, 1
  - 사람이 이해하는 것: 자연어 (한국어, 영어, 중국어, …)
- **목적**
  - 컴퓨터와 사람이 이해할 수 있는 공통의 규칙
  - 컴퓨팅 사고에 따른 사람들의 생각을 더 명확하고 편리하게 표현하기 위한 규칙
- **종류**: 필요에 따라 수많은 언어가 만들어졌음
  - C, C++, Java, python, Objective-C, swift, go, JavaScript, scala, php, html, css, Haskell, Lisp, Shakespeare, …
- **역사**
  - http://alumni.cs.ucr.edu/~vladimir/cs181/PLchart.png
  - http://www.cs.toronto.edu/~gpenn/csc324/PLhistory.pdf

Mother Tongues

Tracing the roots of computer languages through the ages

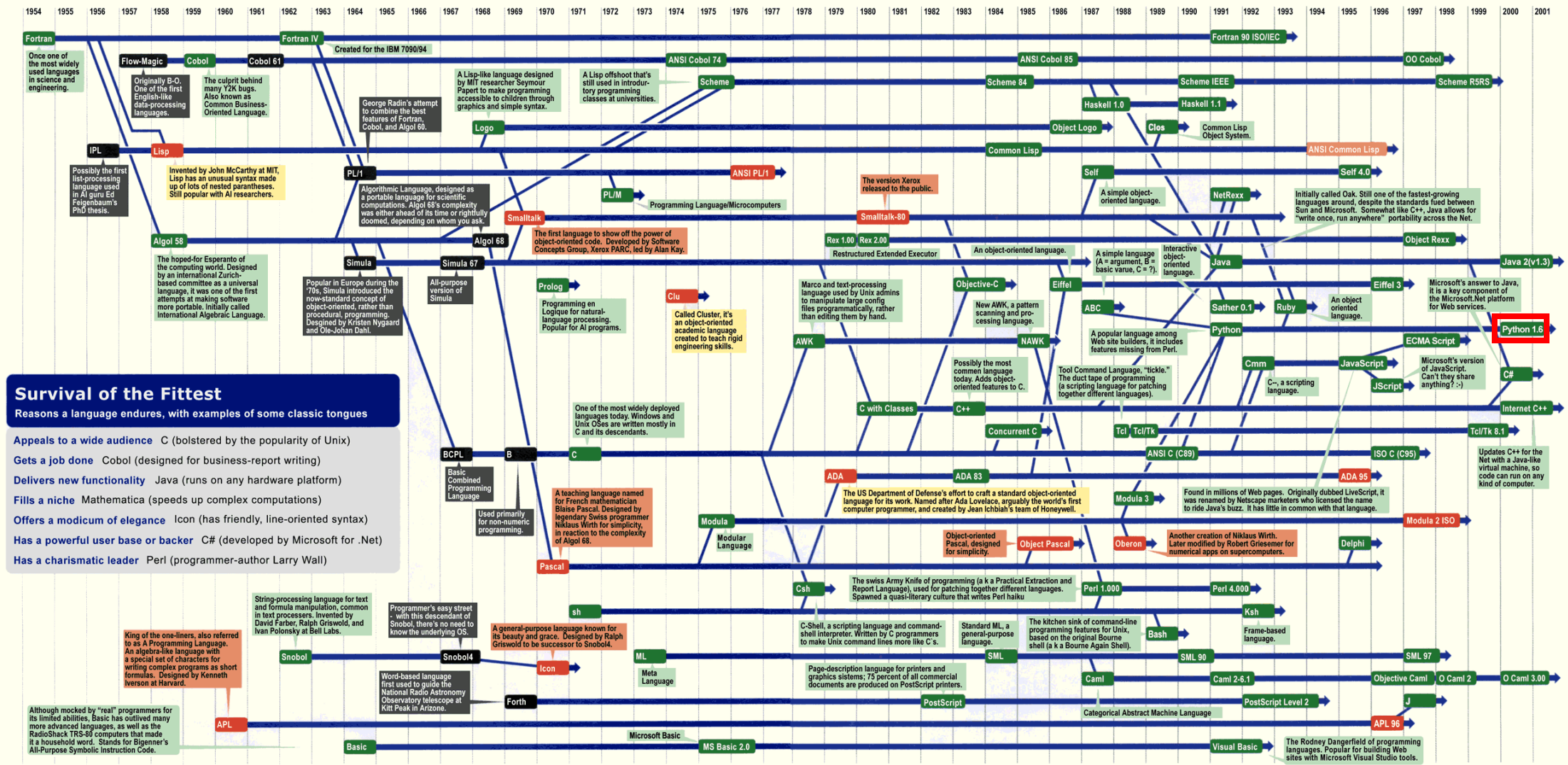
Just like half of the world's spoken tongues, most of the 2,300-plus computer programming languages are either endangered or extinct. As powerhouses C/C++, Visual Basic, Cobol, Java and other modern source codes dominate our systems, hundreds of older languages are running out of life.

An ad hoc collection of engineers-electronic lexicographers, if you will-aim to save, or at least document the lingo of classic software. They're combing the globe's 9 million developers in search of coders still fluent in these nearly forgotten lingua frangas. Among the most endangered are Ada, APL, B (the predecessor of C), Lsp, Oberon, Smalltalk, and Simula.

Code-raker Grady Booch, Rational Software's chief scientist, is working with the Computer History Musuem in Silicon Valley to record and, in some cases, maintain languages by writing new compilers so our ever-changing hardware can grok the code. Why bother? "They tell us about the state of software practice, the minds of their inventors, and the technical, social, and economic forces that shaped history at the time," Booch explains. "They'll provide the raw material for software archaeologists, historians, and developers to learn what worked, what was brilliant, and what was an utter failure." Here's a peek at the strongest branches of programming's family tree. For a nearly exhaustive rundown, check out the Language List at *HTTP://www.informatik.uni-freiburg.de/Java/misc/lang_list.html.* - Michael Mendeno

Key

- 1954 Year Introduced
- **Active:** thousands of users
- **Protected:** taught at universities; compilers available
- **Endangered:** usage dropping off
- **Extinct:** no known active users or up-to-date compilers
- **Lineage continues**



Survival of the Fittest

Reasons a language endures, with examples of some classic tongues

- **Appeals to a wide audience** C (bolstered by the popularity of Unix)
- **Gets a job done** Cobol (designed for business-report writing)
- **Delivers new functionality** Java (runs on any hardware platform)
- **Fills a niche** Mathematica (speeds up complex computations)
- **Offers a modicum of elegance** Icon (has friendly, line-oriented syntax)
- **Has a powerful user base or backer** C# (developed by Microsoft for .Net)
- **Has a charismatic leader** Perl (programmer-author Larry Wall)

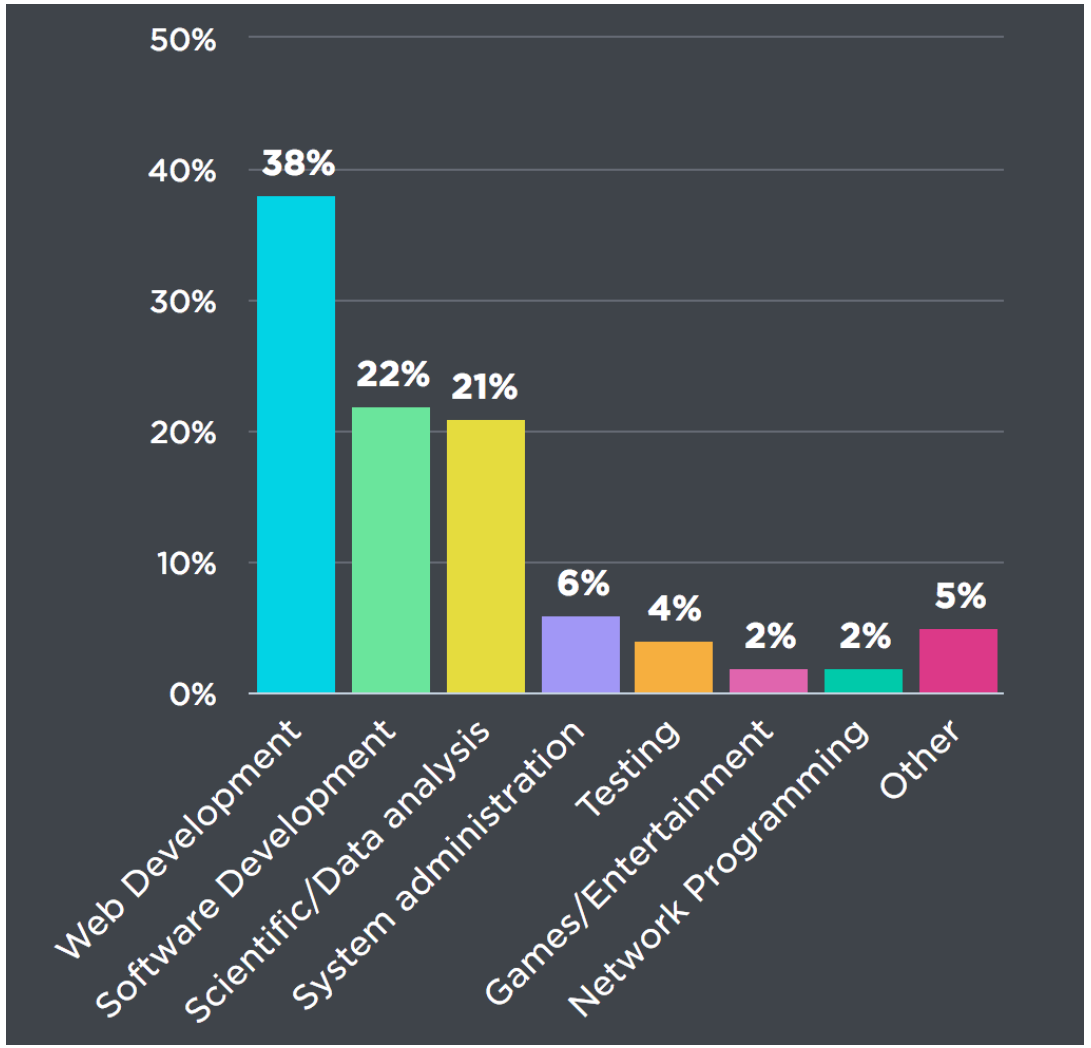
Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

python는 배우기 쉽고, 강력한 언어

- 공식 소개글
  - python은 **배우기 쉽고**, **강력한** 프로그래밍 언어입니다. python은 **효율적인 고수준 데이터 구조**를 갖추고 있으며, 간단하지만 효과적인 **객체 지향 프로그래밍** 접근법 또한 갖추고 있습니다. 우아한 문법과 동적 타이핑, 그리고 인터프리팅 환경을 갖춘 python은 **다양한 분야**, **다양한 플랫폼**에서 사용될 수 있는 최적의 스크립팅 RAD(rapid application development) 언어입니다
- 참고: 인터프리터 기반이기 때문에, 컴파일 기반 언어보다 일반적으로 실행속도가 느림. 단, 일반적으로 개발속도는 훨씬 더 빨라질 수 있음

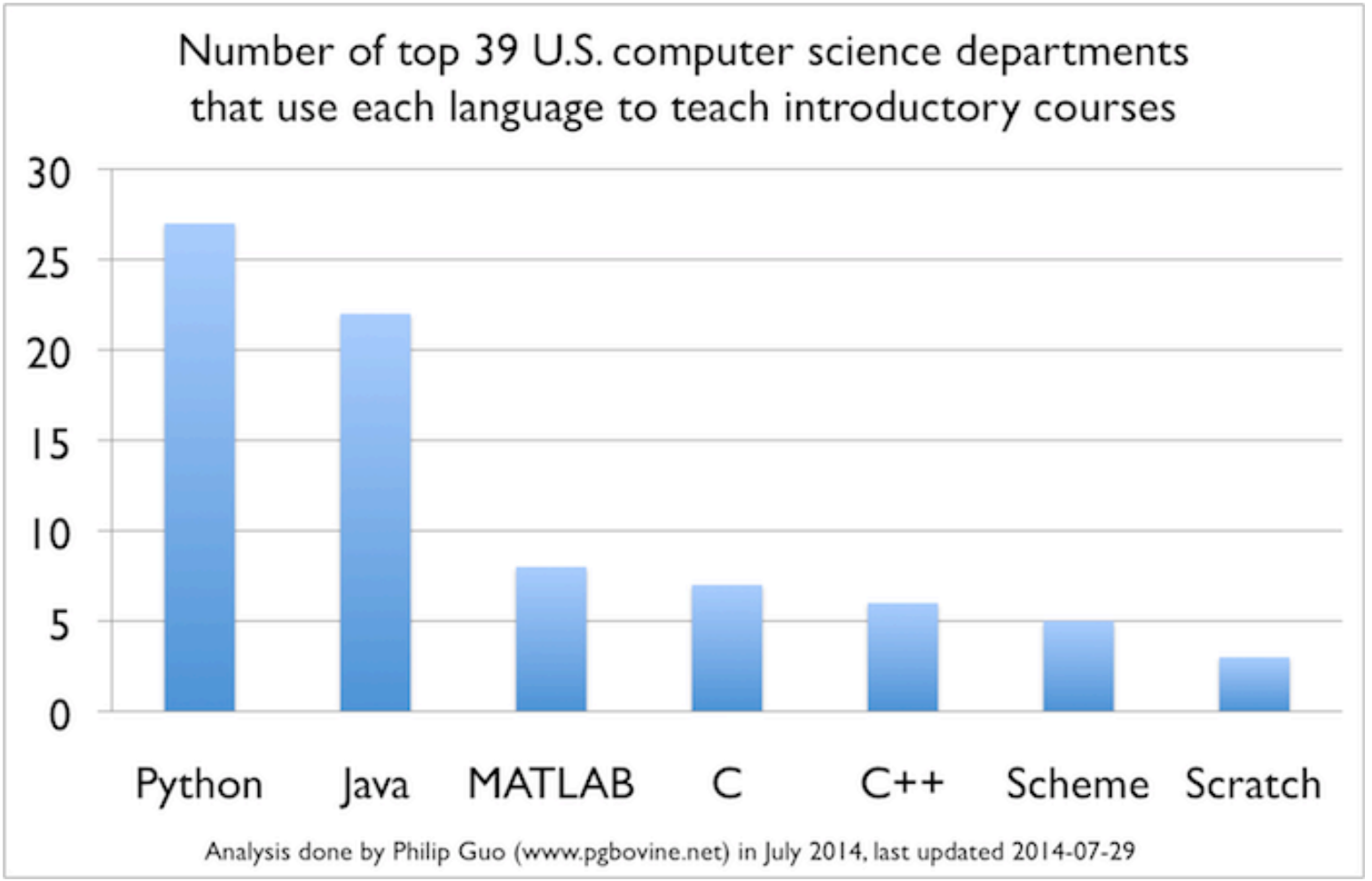
python은 교육, 웹개발, 데이터 분석 등 다양한 분야에 쓰임

- 교육용
- 웹개발: django, flask 등의 프레임워크를 이용한 개발
- Linux/윈도우/맥용 소프트웨어 개발
- 데이터 분석: big data, machine learning, deep learning, …
- 시스템 관리
- 게임 개발
- 모바일 앱 개발
- 임베이드 시스템 개발: IoT 등에 적용
- …



출처: JetBrains

python은 교육용 언어로 가장 널리 사용되고 있음



출처: http://bit.ly/popular-py

뿐만 아니라, 개발용 언어로도 널리 사용되고 있음

순위	TIOBE Index (Feb. 2017)	Top programming languages on github (Sep. 2016)	The top programming languages by IEEE (Jul. 2016)
1	Java	Java script	C
2	C	Java	Java
3	C++	Python	python
4	C#	Ruby	C++
5	Python	PHP	R
6	PHP	C++	C#
7	Java script	CSS	PHP
8	Visual basic	C#	Javascript
9	Delphi/object pascal	C	Ruby
10	Perl	Go	Go

출처: TIOBE index, github, IEEE

python 2 vs python 3

- python 3*
  - 2008.12.3. 공개
  - python 2의 문제점을 해결하기 위해 하위호환성을 포기
  - python 2는 2020년까지만 지원
  - python 2가 아직 좀 더 널리 쓰이나, python 3가 점차 주류로 자리 잡고 있음

- 참고자료
  - 파이썬3 우려인가 위협인가? 파이썬 컴퓨팅의 미래를 준비하며: http://bit.ly/1L6WL4N
  - The key differences between Python 2.7.x and Python 3.x with examples: http://bit.ly/1p3xC12

* 이번 강의에서는 python 3.6을 기준으로 진행함

python 2 vs python 3 (cont.)

구분	python 2	python 3
print	`>>> print 'abc'` `abc`	`>>> print('abc')` `abc`

나눗셈	`>>> 3 / 2`
	`1`
	`>>> 3 / 2.0`
	`1.5`
		`>>> 3 / 2`
		`1.5`
		`>>> 3 // 2`
		`1`

문자열	Latin 문자가 기본	유니코드가 기본 (UTF-8) - 한글, 한자 등 표시가 더 간편

python의 기본 자료형 (built-in types)

- python에는 여러 가지 자료형들(data types)이 정의되어 있다.
- 부울 (`bool`): `True`, `False`
- 수
  - 정수 (`int`): `1`, `2`, `0`, `-1`, `-2`, `100000`, …
  - 실수 (`float`): `3.14`, `1.0`, `6.02e23` (=$6.02*10^{23}$), …
  - 복소수 (`complex`): `1 + 2j`, `j`, `1+4j`
- 문자열 (`str`): `'Hello, world!'`, `"Hello, World!"`

수에 관한 연산자들 (일부)

연산자	설명	예	결과
+	더하기	5 + 8	13
-	빼기	90 - 10	80
*	곱하기	4 * 7	28
/	부동소수점 나누기	7 / 2	3.5
//	정수 나누기	7 // 2	3
%	나머지	7 % 2	1
**	지수	3 ** 4	81

print(): 화면에 글자를 출력하는 함수

- 참고: python에서 함수
  - 재사용이 가능한, 특정한 작업(계산, 입출력 등)을 수행하는 명령어의 집합
  - 인자(argument)를 가질 수 있다 → 수학에서 인자와 동일
  - 반환값(return value)을 가질 수 있다 → 수학에서 함수값과 동일
- print()
  - 인자에 있는 내용을 화면에 출력하고 줄을 바꿈
  - 인자가 여러 개 있는 경우, 여러 인자 사이에 빈 칸을 추가하고 출력함

예제: print()를 이용하여 계산기처럼 사용

```
print(3)   # 정수
print(3.0) # 실수
print(2 + 3)
print(2 - 3)
print(2 * 3)
print(2 / 3)
print(2 // 3) # 정수형으로 나눗셈
print(2 ** 3) # 2 * 2 * 2
print('Hello, World!') # 문자열
print("Hello, World!") # 문자열
print(1, 2, 3) # 여러 개의 인자
print(2 * 2 * 2, 2 ** 3)
```

```
3
3.0
5
-1
6
0.6666666666666666
0
8
Hello, World!
Hello, World!
1 2 3
8 8
```

변수: data를 저장하는 공간으로 이름이 붙어 있음

- 정의: 이름(변수명)이 붙어있는 정보(data)를 저장할 수 있는 공간
- 변수명의 규칙
  - 첫문자: 알파벳 문자 혹은 밑줄(_)
  - 나머지: 문자, 밑줄, 숫자
  - 대/소문자를 구별
  - python keyword[1]는 제외
- 좋은 예: `i, name_2_3`
- 나쁜 예: `2things, two words, my-name, >a1b2_3`
- 일반적으로 변수이름은 모두 소문자, 단어 사이에서는 밑줄을 사용

1 https://docs.python.org/3/reference/lexical_analysis.html?#keywords

예제: python에서 변수 사용

```
# 변수 pi를 정의하고, 값을 대입
pi = 3.1415926534
print(pi)

# 변수 r을 정의하고, 값을 대입
r = 3
print(pi * r ** 2)

# 이미 만들어진 변수 r의 값을 변경
r = 5
print(pi * r ** 2)
```

```
3.1415926534
28.2743338806
78.539816335
```

input(): 키보드로부터 입력을 받는 함수

- input()
  - 인자에 있는 경우, 인자의 내용을 화면에 출력
  - 키보드로 입력받은 내용을 문자열형으로 반환값으로 돌려줌
    → 변수에 대입하면 그 내용이 변수에 저장됨

`int()`, `float()`: 정수 혹은 실수로 형변환

- 형변환(type conversion)
  - 한 자료형(data type)에서 다른 자료형으로 변경하는 것
  - 원하는 연산(예: 사칙연산)을 수행하기 위하여 형변환이 필요
- 형변환 함수: 바뀔형의 이름을 함수처럼 사용
  - `int()`: 인자를 정수형으로 형변환
  - `float()`: 인자를 실수형으로 형변환
- 참고: 형변환이 가능하지 않은 경우에는 오류가 발생함
  - 예: 문자열 'abc'는 정수 혹은 실수로 변환되지 않는다

예제: python에서 키보드 입력

```
name = input('너의 이름은? ')
print('당신은', name, '로군요')

pi = 3.1415926534
# 키보드로부터 값을 입력받음
r = input('Enter a radius: ')
r = float(r) # 연산을 위해 형변환
print(pi * r ** 2)

# input()의 반환값을 float()의 인자로 사용
r = float(input('Enter a radius: '))
print(pi * r ** 2)
```

```
너의 이름은? 미츠하
당신은 미츠하 로군요
Enter a radius: 3
28.2743338806
Enter a radius: 3
28.2743338806
```

참고: 오른쪽 출력결과의 파란색은 키보드 입력을 의미함

읽을 거리

- Marc Andreessen on Why software is eating the world, WSJ (번역)
- 컴퓨팅 사고: Wing M. Jeannette. (2006). Computational Thinking. “Communications of the ACM”, 49(3), 33-35 (번역)

ANY QUESTIONS?