Lecture #01 | 프로그래밍과 python 소개

SE213 프로그래밍 (2019)

Written by Mingyu Cho

Edited by Donghoon Shin

오늘 다룰 내용

- 강좌 소개 및 운영에 관한 사항
- 프로그래밍 교과 소개 및 목적
- 컴퓨터 혹은 프로그래밍은 무엇인가?
- python소개와 기초 문법
  - 기본 자료형과 연산
  - 표준 출력 (standard output): print()
  - 변수
  - 표준 입력 (standard input): input()

프로그래밍(SE213) 강좌 소개

- 코스페이지
  - LMS: http://lms.dgist.ac.kr/ (To be updated)

- 프로그래밍 환경
  - elice (https://dgist.elice.io/courses/944/): 프로그래밍 실습 및 과제 제출
  - anaconda (https://www.anaconda.com/distribution/): python 설치
  - pycharm (https://www.jetbrains.com/pycharm/): 일반적인 개발 환경

- 이론과 실습 모두 동시에 수강해야 함 (학점도 동일하게 부여)

프로그래밍(SE213) 강좌 소개

- **과제 및 시험**
  - 과제: 실습시간에 배우는 내용을 기초로 해서, 학기 중에 4-6번의 과제가 제출될 예정입니다.
  - 시험: 중간고사와 기말고사
- 평가 방법
  - 평가 비중

내용	비중
중간고사	30%
기말고사	30%
과제(실습 내용 포함)	40%

  - 출석: 결석 penalty (전체 100점 기준)
    - 이론: $2^{\max(n-2,0)} - 1$, n=결석회수
    - 실습: 2회 1점, 3회 3점, 4회 이상 F

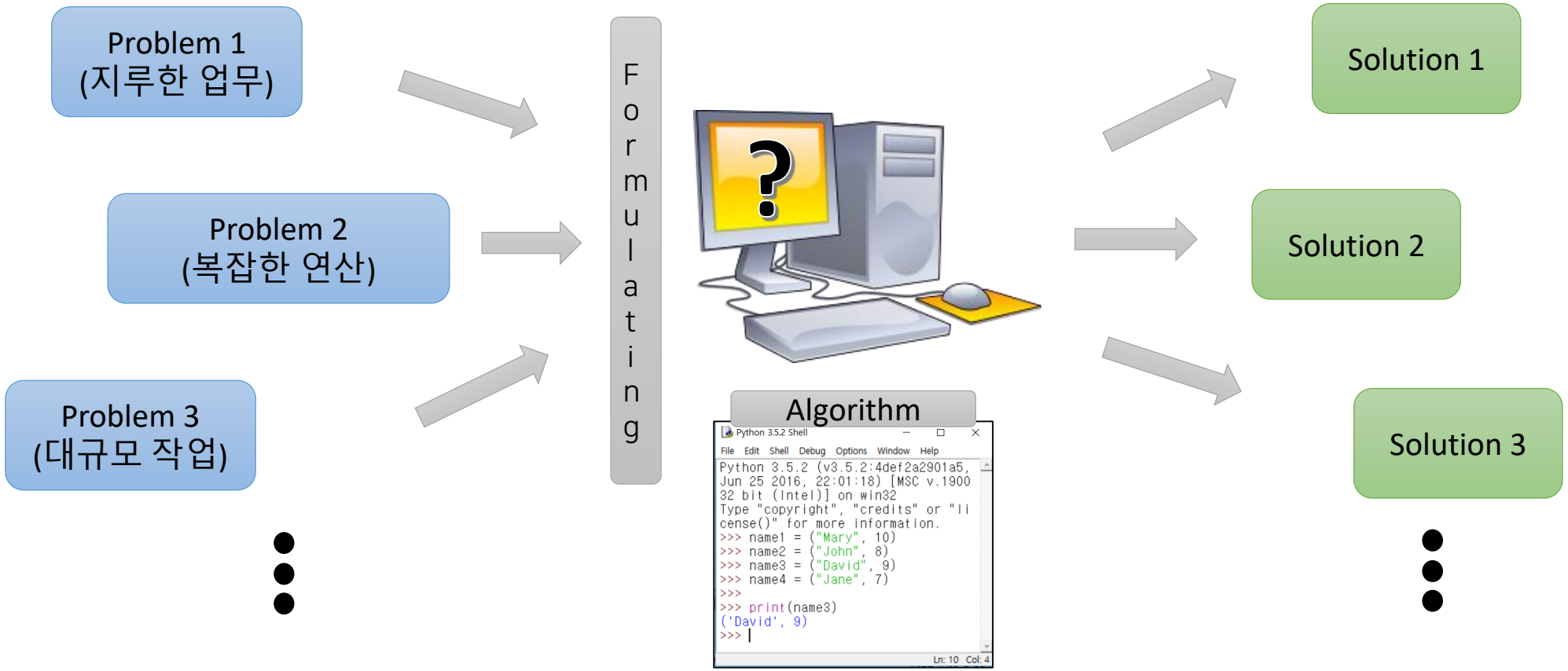
프로그래밍 강좌의 대상과 목적

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

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

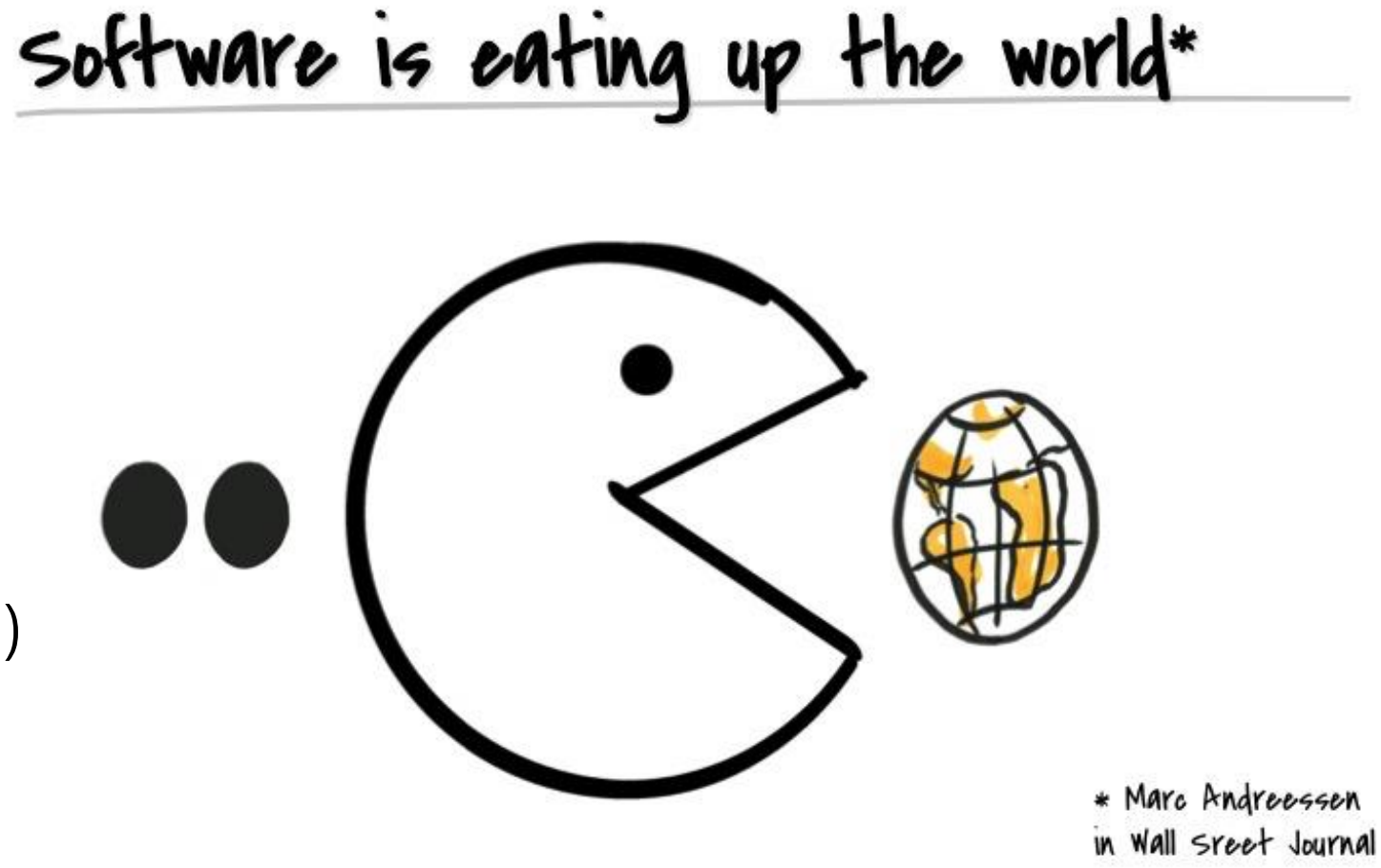
컴퓨터 활용

- 컴퓨터를 **도구**로 활용하는 방법



왜 소프트웨어인가?

- 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
  - …

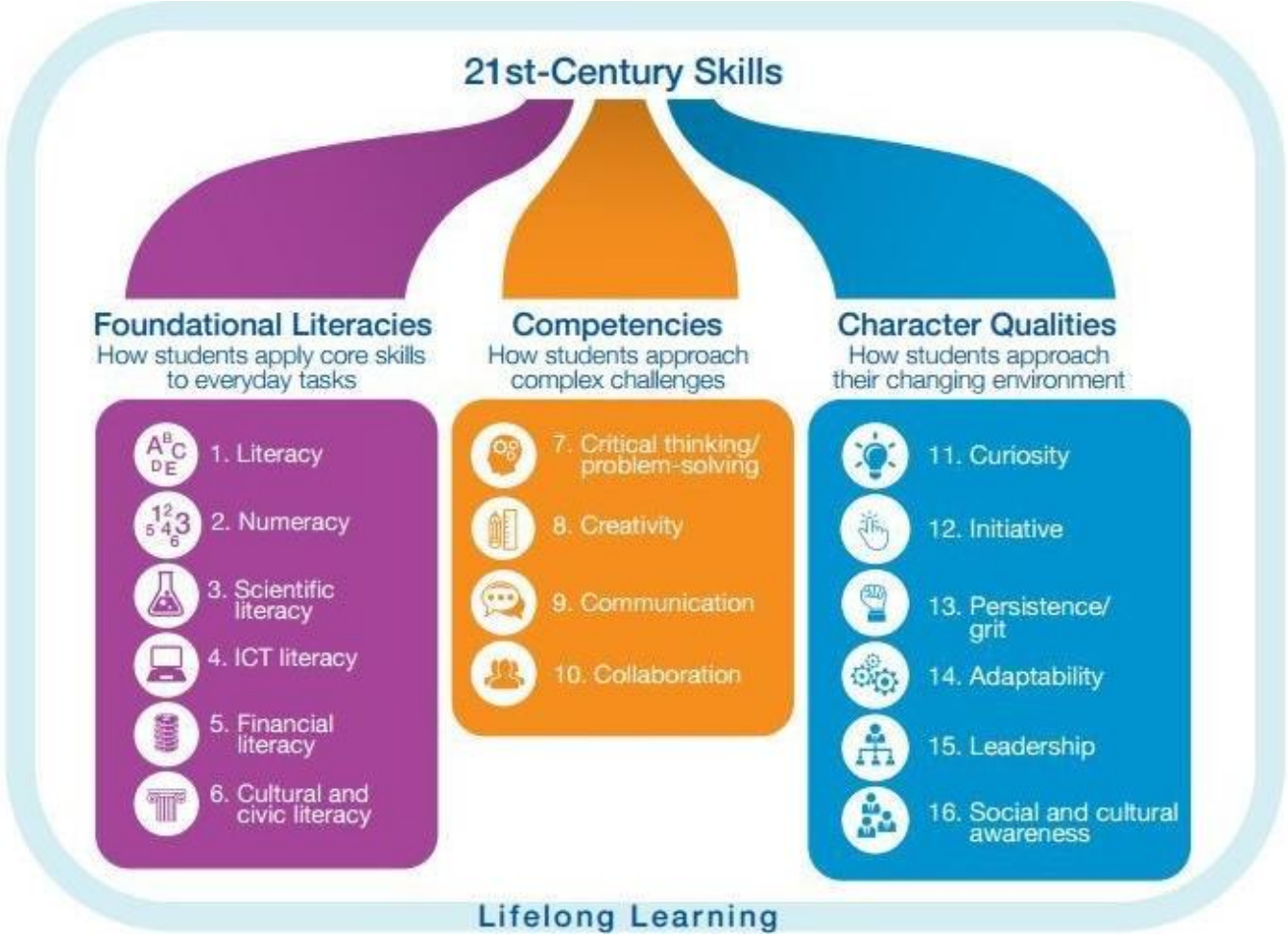


출처: Marc Andreessen on Why software is eating the world

21세기에는 한 분야의 지식보다 복합적 지식/사고력이 요구됨

- 기술의 발전할수록 새로운 기술의 개발하고 그것을 현실 세계에 적용시키기 위해서, 더 복잡한 문제(problem)들을 해결해햐 함

→ 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."

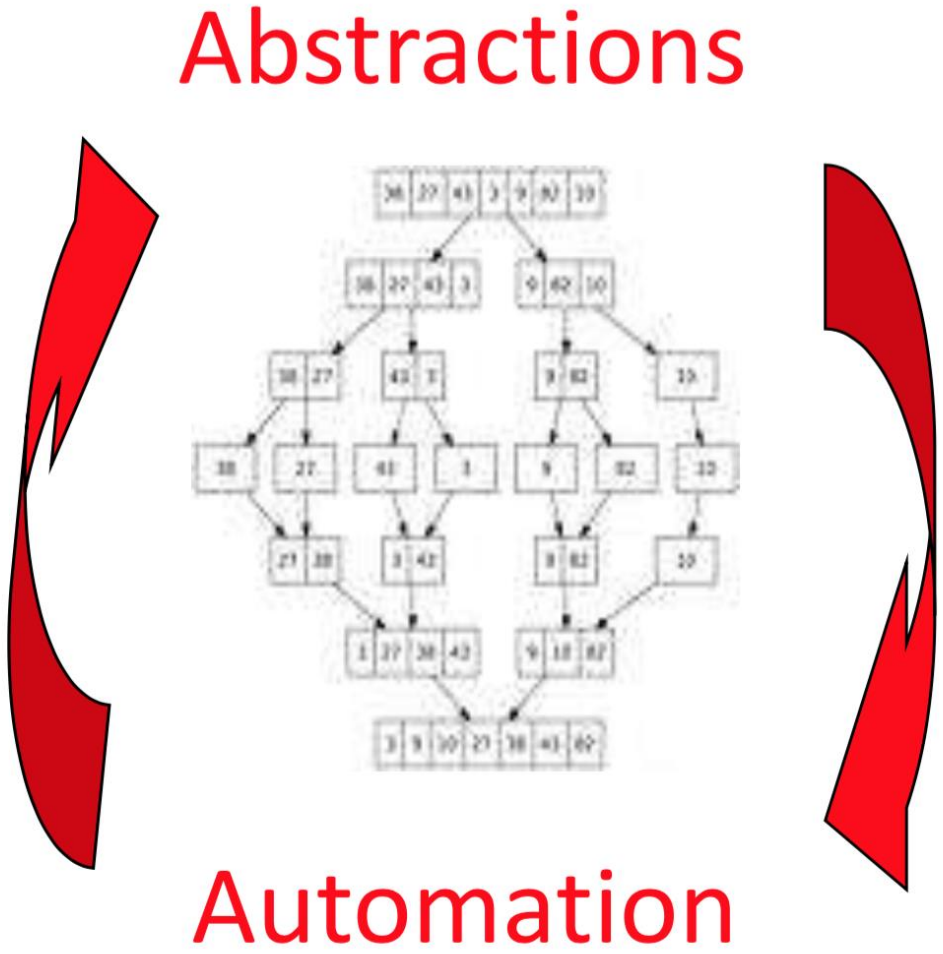
–Carl Sagan

"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

- 연산적 사고(Computational Thinking)의 정의
  - 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, …

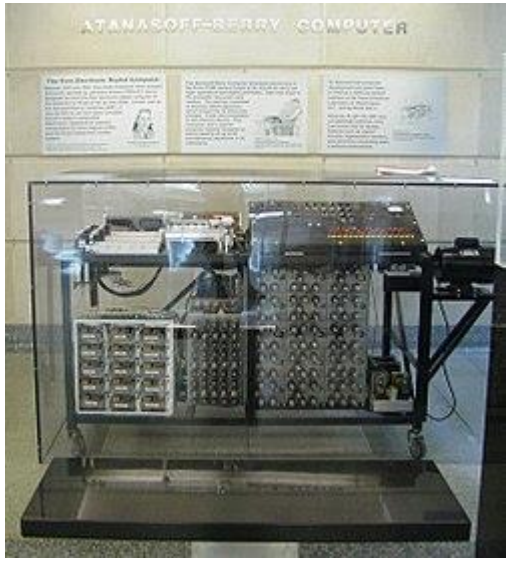
프로그래밍, 혹은 다른 분야를 잘 하기 위해 필요한 것들

- 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…)

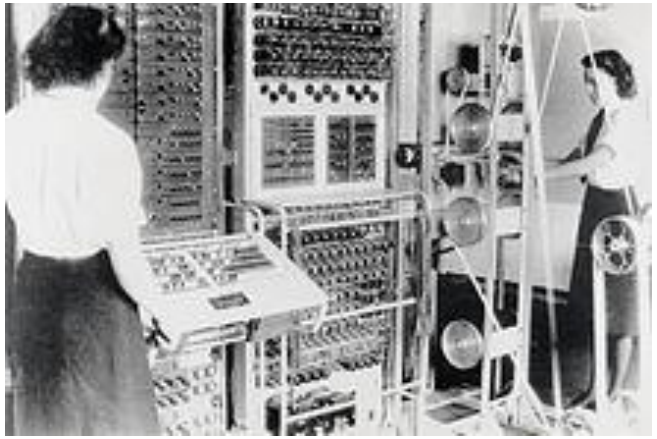
컴퓨터로 할 수 있는 것들...?

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

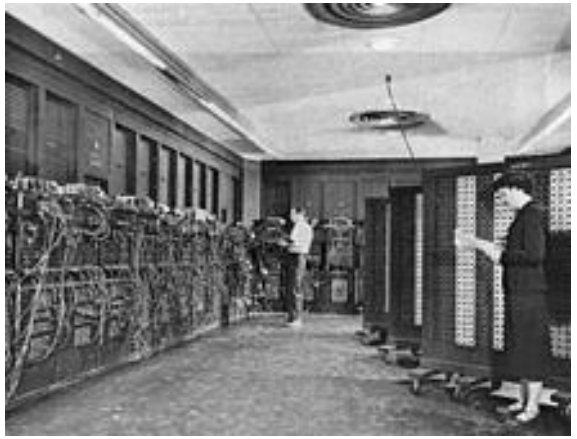
최초의 컴퓨터(computer)



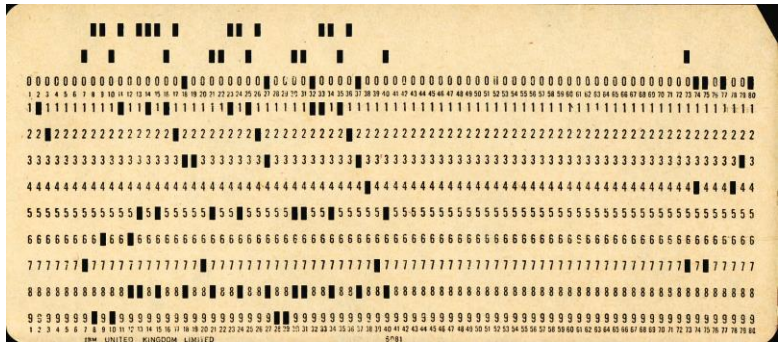
Atanasoff-Berry



Colossus



ENIAC



출처: [Wikipedia]

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, R, …
- **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)

1954 1955 1956 1957 1958 1959 1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001

Fortran — Once one of the most widely used languages in science and engineering.
Fortran IV — Created for the IBM 7090/94
Fortran 90 ISO/IEC
Flow-Matic — Originally B-O. One of the first English-like data-processing languages.
Cobol — The culprit behind many Y2K bugs. Also known as Common Business-Oriented Language.
Cobol 61
ANSI Cobol 74
ANSI Cobol 85
OO Cobol
IPL — Possibly the first list-processing language used in AI guru Ed Feigenbaum's PhD thesis.
Lisp — Invented by John McCarthy at MIT, Lisp has an unusual syntax made up of lots of nested parantheses. Still popular with AI researchers.
George Radin's attempt to combine the best features of Fortran, Cobol, and Algol 60.
A Lisp-like language designed by MIT researcher Seymour Papert to make programming accessible to children through graphics and simple syntax.
A Lisp offshoot that's still used in introductory programming classes at universities.
Scheme
Scheme 84
Scheme IEEE
Scheme R5RS
Haskell 1.0
Haskell 1.1
Logo
Object Logo
Clos — Common Lisp Object System.
Common Lisp
ANSI Common Lisp
PL/1
ANSI PL/1
Self — A simple object-oriented language.
Self 4.0
PL/M — Programming Language/Microcomputers
The version Xerox released to the public.
NetRexx
Initially called Oak. Still one of the fastest-growing languages around, despite the standards fued between Sun and Microsoft. Somewhat like C++, Java allows for "write once, run anywhere" portability across the Net.
Smalltalk — The first language to show off the power of object-oriented code. Developed by Software Concepts Group, Xerox PARC, led by Alan Kay.
Smalltalk-80
Algol 58 — The hoped-for Esperanto of the computing world. Designed by an international Zurich-based committee as a universal language, it was one of the first attempts at making software more portable. Initially called International Algebraic Language.
Algol 68
Algorithmic Language, designed as a portable language for scientific computations. Algol 68's complexity was either ahead of its time or rightfully doomed, depending on whom you ask.
Rex 1.00
Rex 2.00 — Restructured Extended Executor
An object-oriented language.
Object Rexx
Simula — Popular in Europe during the '70s, Simula introduced the now-standard concept of object-oriented, rather than procedural, programming. Desgined by Kristen Nygaard and Ole-Johan Dahl.
Simula 67 — All-purpose version of Simula
A simple language (A = argument, B = basic varue, C = ?).
Interactive object-oriented language.
Java
Java 2(v1.3)
Prolog — Programming en Logique for natural-language processing. Popular for AI programs.
Clu — Called Cluster, it's an object-oriented academic language created to teach rigid engineering skills.
Marco and text-processing language used by Unix admins to manipulate large config files programmatically, rather than editing them by hand.
Objective-C
Eiffel
Eiffel 3 — Microsoft's answer to Java, it is a key component of the Microsoft.Net platform for Web services.
New AWK, a pattern scanning and processing language.
ABC
Sather 0.1
Ruby — An object oriented language.
Python
Python 1.6
AWK
NAWK — A popular language among Web site builders, it includes features missing from Perl.
ECMA Script
Possibly the most commen language today. Adds object-oriented features to C.
Tool Command Language, "tickle." The duct tape of programming (a scripting language for patching together different languages).
Cmm
C--, a scripting language.
JavaScript
JScript — Microsoft's version of JavaScript. Can't they share anything? :-)
C#
One of the most widely deployed languages today. Windows and Unix OSes are written mostly in C and its descendants.
C with Classes
C++
Internet C++
Concurrent C
Tcl
Tcl/Tk
Tcl/Tk 8.1
BCPL — Basic Combined Programming Language
B — Used primarily for non-numeric programming.
C
ANSI C (C89)
ISO C (C95)
Updates C++ for the Net with a Java-like virtual machine, so code can run on any kind of computer.
ADA — The US Department of Defense's effort to craft a standard object-oriented language for its work. Named after Ada Lovelace, arguably the world's first computer programmer, and created by Jean Ichbiah's team of Honeywell.
ADA 83
ADA 95
Modula 3
Found in millions of Web pages. Originally dubbed LiveScript, it was renamed by Netscape marketers who licensed the name to ride Java's buzz. It has little in common with that language.
A teaching language named for French mathematician Blaise Pascal. Designed by legendary Swiss programmer Niklaus Wirth for simplicity, in reaction to the complexity of Algol 68.
Modula — Modular Language
Modula 2 ISO
Object-oriented Pascal, designed for simplicity.
Object Pascal
Oberon — Another creation of Niklaus Wirth. Later modified by Robert Griesemer for numerical apps on supercomputers.
Delphi
Pascal
Csh — The swiss Army Knife of programming (a k a Practical Extraction and Report Language), used for patching together different languages. Spawned a quasi-literary culture that writes Perl haiku
Perl 1.000
Perl 4.000
String-processing language for text and formula manipulation, common in text processors. Invented by David Farber, Ralph Griswold, and Ivan Polonsky at Bell Labs.
sh
Ksh — Frame-based language.
Programmer's easy street - with this descendant of Snobol, there's no need to know the underlying OS.
A general-purpose language known for its beauty and grace. Designed by Ralph Griswold to be successor to Snobol4.
C-Shell, a scripting language and command-shell interpreter. Written by C programmers to make Unix command lines more like C's.
Standard ML, a general-purpose language.
The kitchen sink of command-line programming features for Unix, based on the original Bourne shell (a k a Bourne Again Shell).
Bash
King of the one-liners, also referred to as A Programming Language. An algebra-like language with a special set of characters for writing complex programs as short formulas. Designed by Kenneth Iverson at Harvard.
Snobol
Snobol4
Icon
ML — Meta Language
SML
SML 90
SML 97
Word-based language first used to guide the National Radio Astronomy Observatory telescope at Kitt Peak in Arizone.
Page-description language for printers and graphics sistems; 75 percent of all commercial documents are produced on PostScript printers.
Caml
Caml 2-6.1
Objective Caml
O Caml 2
O Caml 3.00
Forth
PostScript
PostScript Level 2
J
Categorical Abstract Machine Language
Although mocked by "real" programmers for its limited abilities, Basic has outlived many more advanced languages, as well as the RadioShack TRS-80 computers that made it a household word. Stands for Bigenner's All-Purpose Symbolic Instruction Code.
APL
APL 96
Microsoft Basic
Basic
MS Basic 2.0
Visual Basic — The Rodney Dangerfield of programming languages. Popular for building Web sites with Microsoft Visual Studio tools.

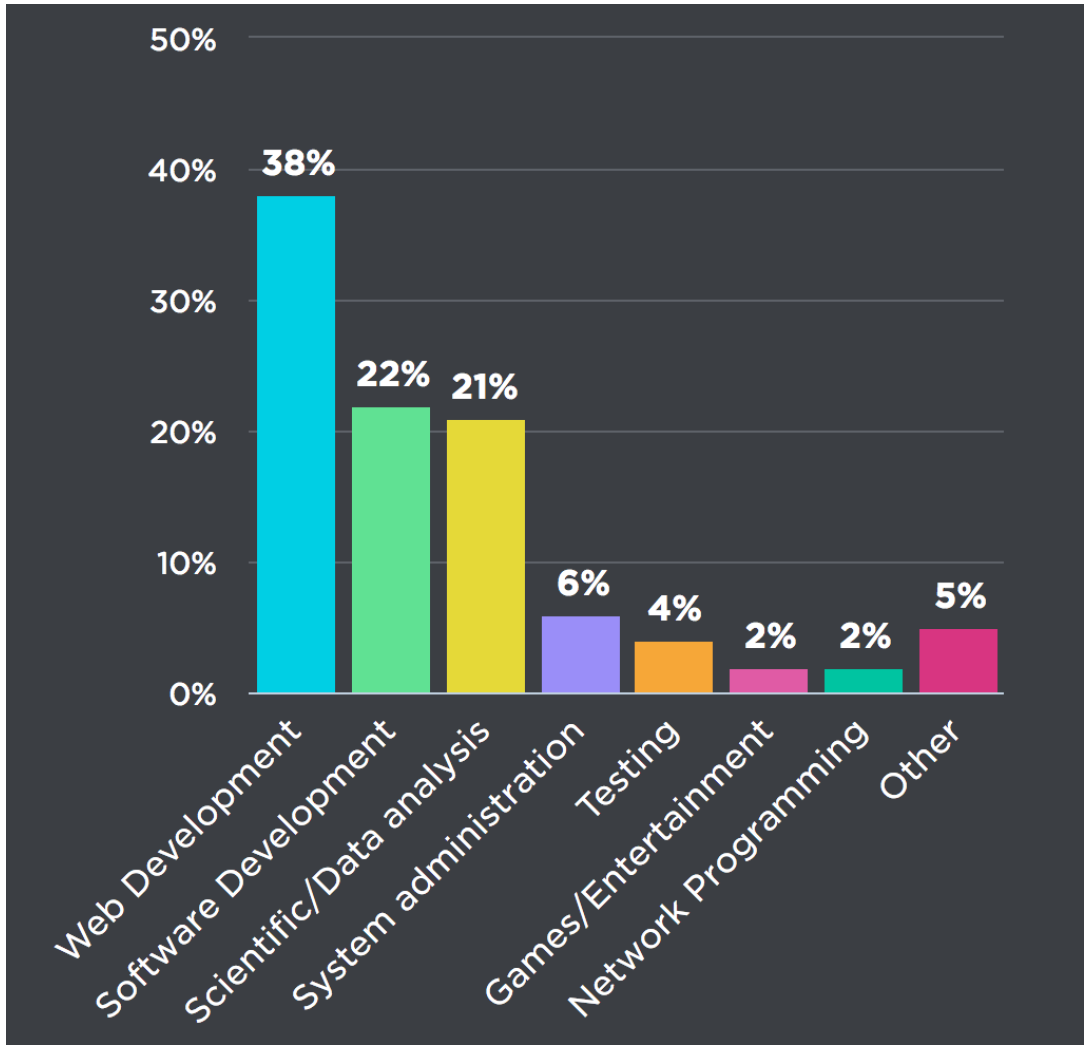
Sources: Paul Boutin; Brent Hailpern, associate director of computer science at IBM Research; The Retrocomputing Museum; Todd Proebsting, senior researcher at Microsoft; Gio Wiederhold, computer scientist, Stanford University

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: python은 인터프리터 기반이기 때문에, 컴파일 기반 언어(예: C, C++)보다 일반적으로 실행속도가 느리지만, 개발속도는 더 빠름

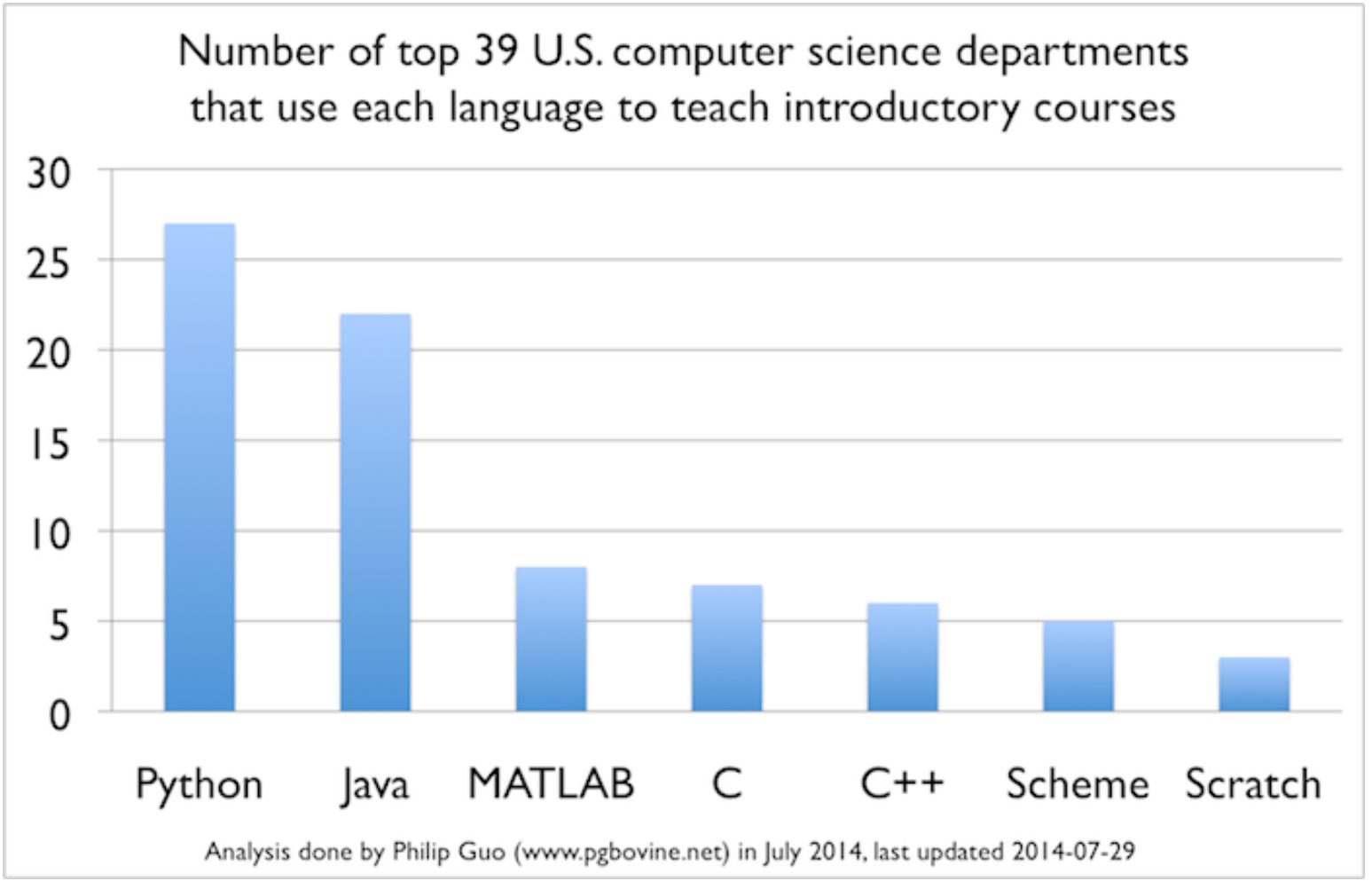
python은 교육, 웹개발, 데이터 분석 등 다양한 분야에 쓰임

- 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. 2019)	Top programming languages on github (Sep. 2016)	The top programming languages by IEEE (Jul. 2016)
1	Java	JavaScript	C
2	C	Java	Java
3	**Python**	**Python**	**Python**
4	C++	Ruby	C++
5	Visual basic .Net	PHP	R
6	JavaScript	C++	C#
7	C#	CSS	PHP
8	PHP	C#	JavaScript
9	SQL	C	Ruby
10	Objective-C	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 (참고. 최신 버전은 3.7임, 2019.2. 기준)

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!"

연산자(operator)

- 연산자는 데이터/값들에 대한 연산을 정의함
  - 자료형에 따라 연산자가 정의됨
  - 같은 연산자가 자료형에 따라 다른 동작을 하기도 함
  - 연산자는 하나 혹은 그 이상의 피연산자(operand)를 가질 수 있음
- 예제
  - 수에 대한 사칙연산
  - 문자열에 대한 연산: 문자열의 연결(concatenation)

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

연산자 우선 순위 (operator precedence)

Operator	Description
lambda	
if – else	
or	
and	
not x	
in, not in, *is, is not,* <, <=, >, >=, !=	
|	*Bitwise OR*
^	*Bitwise XOR*
&	*Bitwise AND*
<<, >>	*Shifts*
+, –	Addition and subtraction
*, @, /, //, %	Multiplication, *matrix multiplication*, division, remainder
+x, –x, ~x	Positive, negative, *bitwise NOT*
**	Exponentiation
await x	*Await expression*
x[index], x[index:index], x(arguments...), x.attribute	Subscription, slicing, call, attribute reference
(expressions...), [expressions...], {key: value...}, {expressions...}	Binding or tuple display, list display, dictionary display, set display

$2^{max(n-2,0)} - 1$, n=3회 결석
→ 2**max(3 - 2, 0) – 1
→ 2**max(1, 0) - 1
→ 2**1 - 1
→ 2 - 1
→ 1

Low

High

Expression

- Expression: 하나의 값을 계산(evaluate)하기 위한 다음 요소들의 조합
  - 값
  - 연산자
  - 변수
  - 함수
- 예제
  - 42
  - 42 + 23
  - max(42, 23)

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

- 변수가 필요한 이유
  - 동일한 값을 여러 번 사용할 필요가 있음
  - 중간 결과를 나타내기 위해서 사용함
  - 수학에서 사용하는 변수와 유사함
- 정의
  - 어떠한 값(value) 혹은 객체(object)에 대한 이름(alias)
  - 값 혹은 객체를 저장할 수 있는 이름이 붙어있는 메모리 상의 공간
- 변수에 값 혹은 객체를 대입(assignment)한 이후, 대입된 값 혹은 객체 대신에 변수의 이름을 사용할 수 있음

변수 이름의 규칙

- 규칙
  - 첫 글자: 알파벳 문자 혹은 밑줄(_)
  - 나머지 글자: 문자, 밑줄, 숫자
  - 대/소문자를 구별
  - python keyword[1]는 제외
- 예
  - 좋은 예: i, name_2_3
  - 나쁜 예: 2things, two words, my-name, >a1b2_3
- 일반적으로 python에서는 변수이름은 모두 소문자, 단어 사이에서는 밑줄을 사용
- 중요: 변수 이름은 어떠한 값들이 저장되는지 알 수 있도록 지정하는 것이 필요함

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

대입(assignment)

- *variable_name = expression*
  - 의미: 오른쪽 expression의 결과값을 왼쪽 변수에 대입하는 명령
    - 주의: 변수 이름이 항상 왼쪽에 있어야 함
    - 참고: 등식은 ==로 나타냄
  - 참고: 일부 언어에서 대입문으로 사용하는 기호: :=, <-
- 예제

```
greeting = 'Hello, world!'
n = 42
pi = 3.14159265
r = 3.0
```

주의: 문자열과 변수

- 문자열: 따옴표(‘ 또는 “)로 둘러싸여 있는 문자들
- 변수: 어떠한 값 또는 객체를 나타내는 이름
- 예제

```
n = 42
var1 = n
var2 = 'n'
var3 = n + 1
var4 = 'n' + 1
```

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

- 표준 입력 (standard input)
  - 프로그램으로 들어가는 데이터 스트림
  - 일반적으로 키보드를 뜻함
- input()
  - 인자에 있는 경우, 인자의 내용을 화면에 출력
  - 키보드로 입력받은 내용을 문자열로 반환
    → 변수에 대입하면 그 내용이 변수에 저장됨

예제: python에서 키보드 입력

```
name1 = input()
print(name1)

name2 = input('당신의 이름은? ')
print('당신은', name2, '이군요.')

name3 = input('당신의 이름은? ')
age = input(name3 + '의 나이는? ')
print(name3, '의 나이는', age, '세 입니다.')
```

```
홍길동
홍길동

당신의 이름은? 홍길동
당신은 홍길동 이군요.

당신의 이름은? 홍길동
홍길동의 나이는? 20
홍길동 의 나이는 20 세 입니다.
```

참고: 오른쪽 출력결과의 파란색은 키보드 입력을 의미함

읽을 거리

- Marc Andreessen on Why software is eating the world, WSJ
  - 원문: https://on.wsj.com/2lDLhKk (번역)

- 컴퓨팅 사고: Wing M. Jeannette. (2006). Computational Thinking. “Communications of the ACM”, 49(3), 33-35 (번역)

ANY QUESTIONS?