Python

프로그래밍 언어마다 특징도 다양하고 장단점도 다양하다.

http://www.inven.co.kr/mobile/board/powerbbs.php?come_idx=2097&category=_%EF%BF%BD%EA%B3%97%EF%BF%BD%EF%BF%BD%3E%C2%A0%C2%A0%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%EF%BF%BD%C2%A0%C2%A0%3C%2Fa%3E%20%7C%3Ca%20href%3D&l=832573

http://kstatic.inven.co.kr/upload/2017/08/03/bbs/i14621578868.jpg

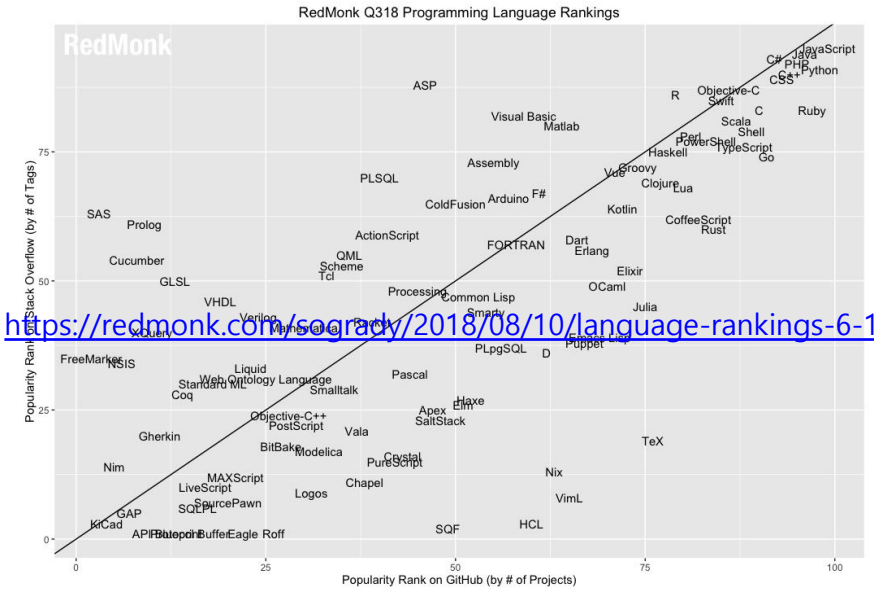
http://redmist.tistory.com/21

http://redmist.tistory.com/30

Worldwide, Sept 2018 compared to a year ago:

Rank	Change	Language	Share	Trend
1	↑	Python	24.58 %	+5.7 %
2	↓	Java	22.14 %	-0.6 %
3	↑	Javascript	8.41 %	+0.0 %
4	↓	PHP	7.77 %	-1.4 %
5		C#	7.74 %	-0.4 %
6		C/C++	6.22 %	-0.8 %
7		R	4.04 %	-0.2 %
8		Objective-C	3.33 %	-0.9 %
9		Swift	2.65 %	-0.9 %
10		Matlab	2.1 %	-0.3 %

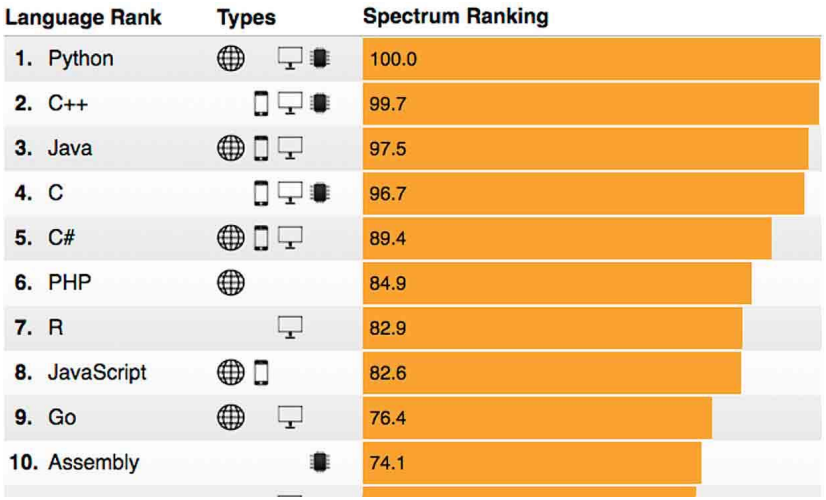
http://pypl.github.io/PYPL.html



https://redmonk.com/sogrady/2018/08/10/language-rankings-6-18/

Sep 2018	Sep 2017	Change	Programming Language	Ratings	Change
1	1		Java	17.436%	+4.75%
2	2		C	15.447%	+8.06%
3	5	↑	Python	7.653%	+4.67%
4	3	↓	C++	7.394%	+1.83%
5	8	↑	Visual Basic .NET	5.308%	+3.33%
6	4	↓	C#	3.295%	-1.48%
7	6	↓	PHP	2.775%	+0.57%
8	7	↓	JavaScript	2.131%	+0.11%
9	**-**	⇈	**SQL**	**2.062%**	**+2.06%**
10	18	⇈	Objective-C	1.509%	+0.00%

https://www.tiobe.com/tiobe-index/



https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages


python™

파이썬은 배우기 쉽고, 강력한 프로그래밍 언어입니다. 효율적인 자료 구조들과 객체 지향 프로그래밍에 대해 간단하고도 효과적인 접근법을 제공합니다. 우아한 문법과 동적 타이핑(typing)은, 인터프리터 적인 특징들과 더불어, 대부분 플랫폼과 다양한 문제 영역에서 스크립트 작성과 빠른 응용 프로그램 개발에 이상적인 환경을 제공합니다.

파이썬 인터프리터와 풍부한 표준 라이브러리는 소스나 바이너리 형태로 파이썬 웹 사이트, https://www.python.org/, 에서 무료로 제공되고, 자유롭게 배포할 수 있습니다. 같은 사이트는 제삼자들이 무료로 제공하는 확장 모듈, 프로그램, 도구, 문서들의 배포판이나 링크를 포함합니다.

파이썬 인터프리터는 C 나 C++ (또는 C에서 호출 가능한 다른 언어들)로 구현된 새 함수나 자료 구조를 쉽게 추가할 수 있습니다. 파이썬은 고객화 가능한 응용 프로그램을 위한 확장 언어로도 적합합니다.

https://docs.python.org/ko/3/tutorial/index.html
https://docs.python.org/ko/3/tutorial/appetite.html

Python 파이썬

웹 앱에서 인공지능까지 인기가 급상승하며 주목받는 언어

탄생
1991년
만든 사람
Guido van Rossum
주요 용도
웹 앱, 데이터 분석, 인공지능
분류
절차형 · 함수형 · 객체지향형/ 인터프리터

이런 언어

데이터 분석에 장점을 가진 스크립트 언어로 '데이터 사이언티스트'라는 직종에 인기가 있다. 해외에서는 웹 애플리케이션의 개발 언어로도 많이 사용되고 있으며, Python 프로그래머의 평균 연봉이 높은 것이 화제가 되기도 한다.
최근에는 기계학습 등 인공지능의 개발에도 많이 사용되고 있다.
버전 2.x와 3.x은 일부 호환이 되지 않지만 두 버전 모두 이용자가 많다.

인덴트가 중요

많은 언어가 '{' '}' 등으로 블록 구조를 표현하는 반면, Python에서는 인덴트들여 쓰기로 표현한다. "코드는 쓰는 것보다 읽는 것이 더 많다"라는 의미에서 PEP8이라는 코딩 가이드도 제공되고 있다.

다양한 구현이 있다

원래의 처리계인 'CPython'뿐만 아니라 'Jython'나 'PyPy', 'Cython'나 'IronPython' 등 다양한 구현이 있으며, 각각 특징이 있다.

풍부한 통계 라이브러리

데이터 분석과 기계학습에 사용할 수 있는 라이브러리가 풍부하게 갖추어져 있으며 무료로 사용할 수 있다. 'NumPy'나 'Pandas', 'matplotlib' 등이 특히 유명하다.

Column

The Zen of Python

Python 프로그래머가 가져야 할 마음가짐이 정리된 말.
Python으로 소스코드를 작성할 때 '단순'과 '가독성'을 실현하기 위한 19개의 문장으로 구성되어 있다.

python

기억해야 할 키워드

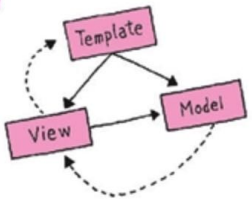
pip

Python 패키지 관리 시스템.
최신 Python이 기본으로 제공되어 검색 및 설치, 업데이트 등도 가능하다.
Git 저장소에서 직접 설치할 수도 있다.

```
[x for x in range(10) if is_prime(x)]
```

리스트 컴프리핸션

리스트를 생성할 때 사용되는 표기 방법.
수학에서 집합을 나타낼 때 사용되는 {x | x는 10 이하의 소수}라는 표현에 가깝다.
for 루프보다 빠르게 처리할 수 있다.



Django

Python으로 사용되는 웹 애플리케이션 프레임워크 중에서도 가장 인기가 있다.
Instagram과 Pinterest의 개발에도 사용되고 있는 것으로 알려져 있다.

프로그래밍 예제 **하노이의 탑(hanoi.py)**

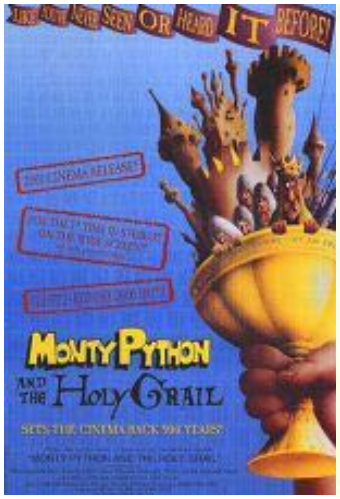
```
# 하노이의 탑
def hanoi(n, from_, to, via):
    if n > 1:
        hanoi(n - 1, from_, via, to)
        print from_ + " -> " + to
        hanoi(n - 1, via, to, from_)
    else:
        print from_ + " -> " + to

# 표준 입력에서 단수를 받아서 실행
n = input()
hanoi(n, "a", "b", "c")
```

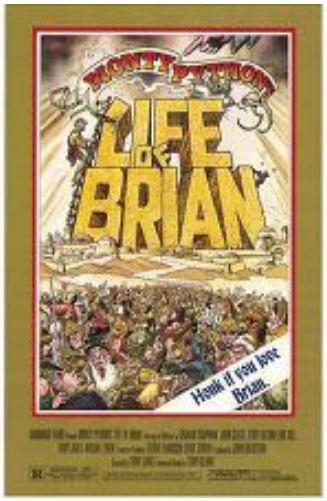
인덴트가 중요

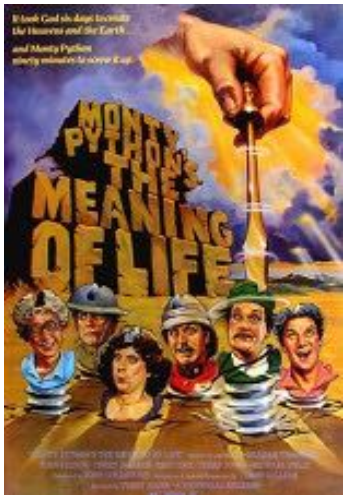
탭 문자 또는 4 문자의 공백이 사용되는 경우가 많다.
하나의 블록은 동일한 들여쓰기로 동일이 필요하다.

From 프로그래밍 언어 도감

Humorous







https://www.youtube.com/watch?v=jiu0lYQlPqE

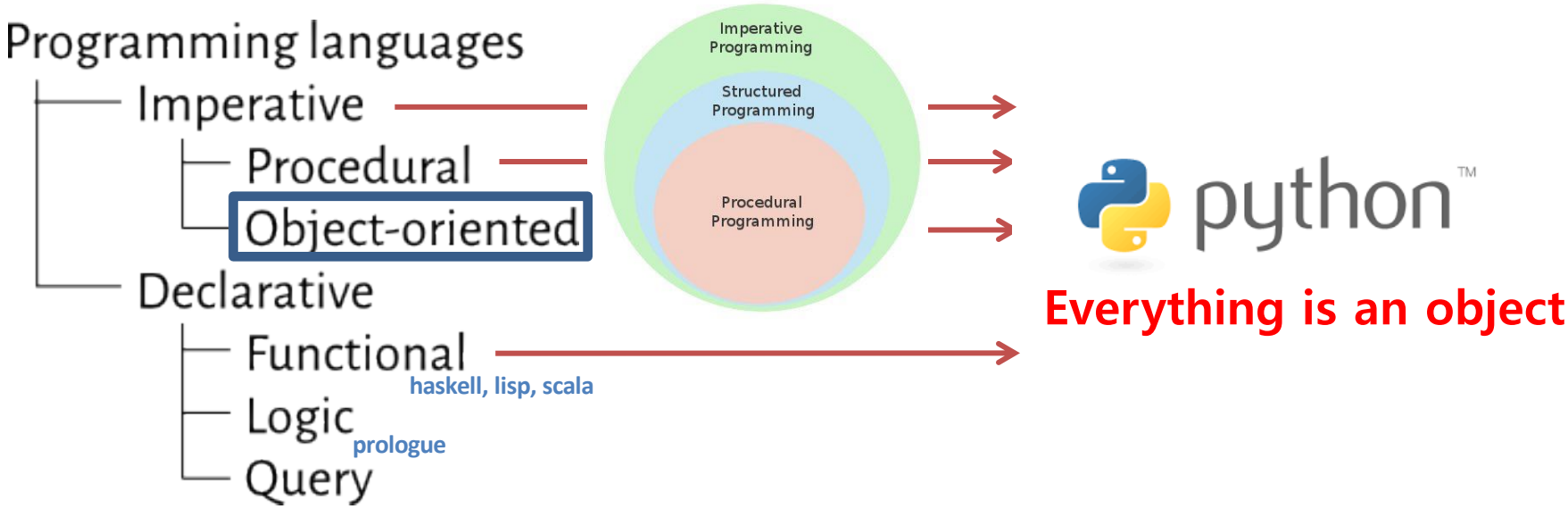
https://gvanrossum.github.io/

Life is too short, You need

생산성

Effective
Efficient

Multi Paradigm



The same problem can be solved and expressed in **different ways**

Glue Language

Language Reference

■ **CPython (c.f : cython)**

- ○ De facto
- ○ Python Software Foundation 관리

■ **대체, 플랫폼 특정 구현체**

- ○ PyPy, Stackless
- ○ IronPython, PythonNet, Jython
- ○ Skulpt, Brython
- ○ MicroPython

https://wiki.python.org/moin/AdvocacyWritingTasks/GlueLanguage
https://www.python.org/doc/essays/omg-darpa-mcc-position/

Library & Tool

■ 다양한 종류의 수많은 **standard library** 기본 탑재

플랫폼에 상관없음

■ 다양한 종류의 수많은 **open-source libraries**

the Python Package Index: https://pypi.python.org

pip

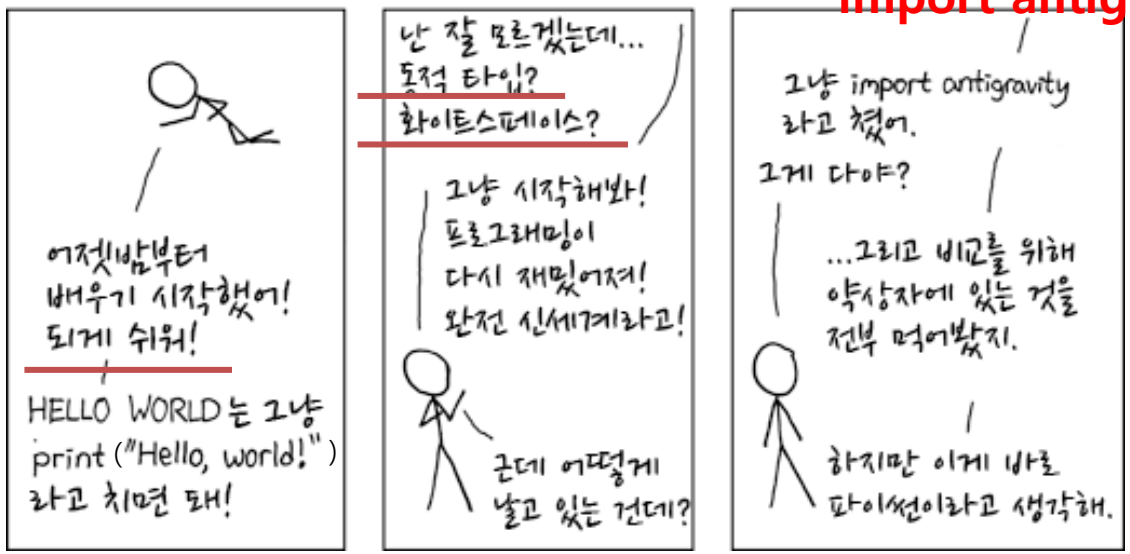
the de facto

default package manager

버전과 라이브러리에 대한 의존성 문제?

General Purpose

import antigravity



C.f)
Domain-specific

https://en.wikipedia.org/wiki/List_of_Python_software
https://github.com/vinta/awesome-python

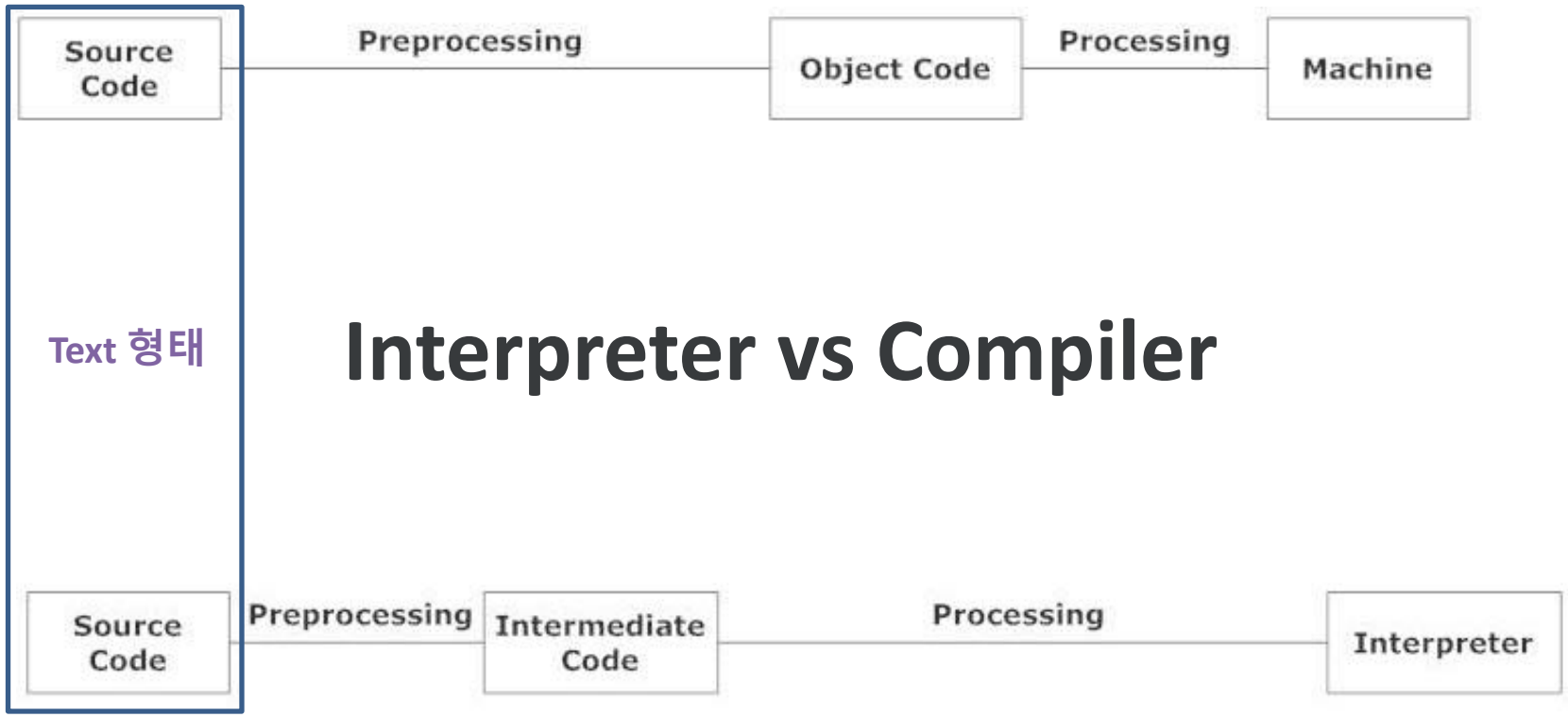
c.f) 모바일 지원에 대한 약점?

Dynamic Language

https://wiki.python.org/moin/Why%20is%20Python%20a%20dynamic%20language%20and%20also%20a%20strongly%20typed%20language

Interpreted Language

script

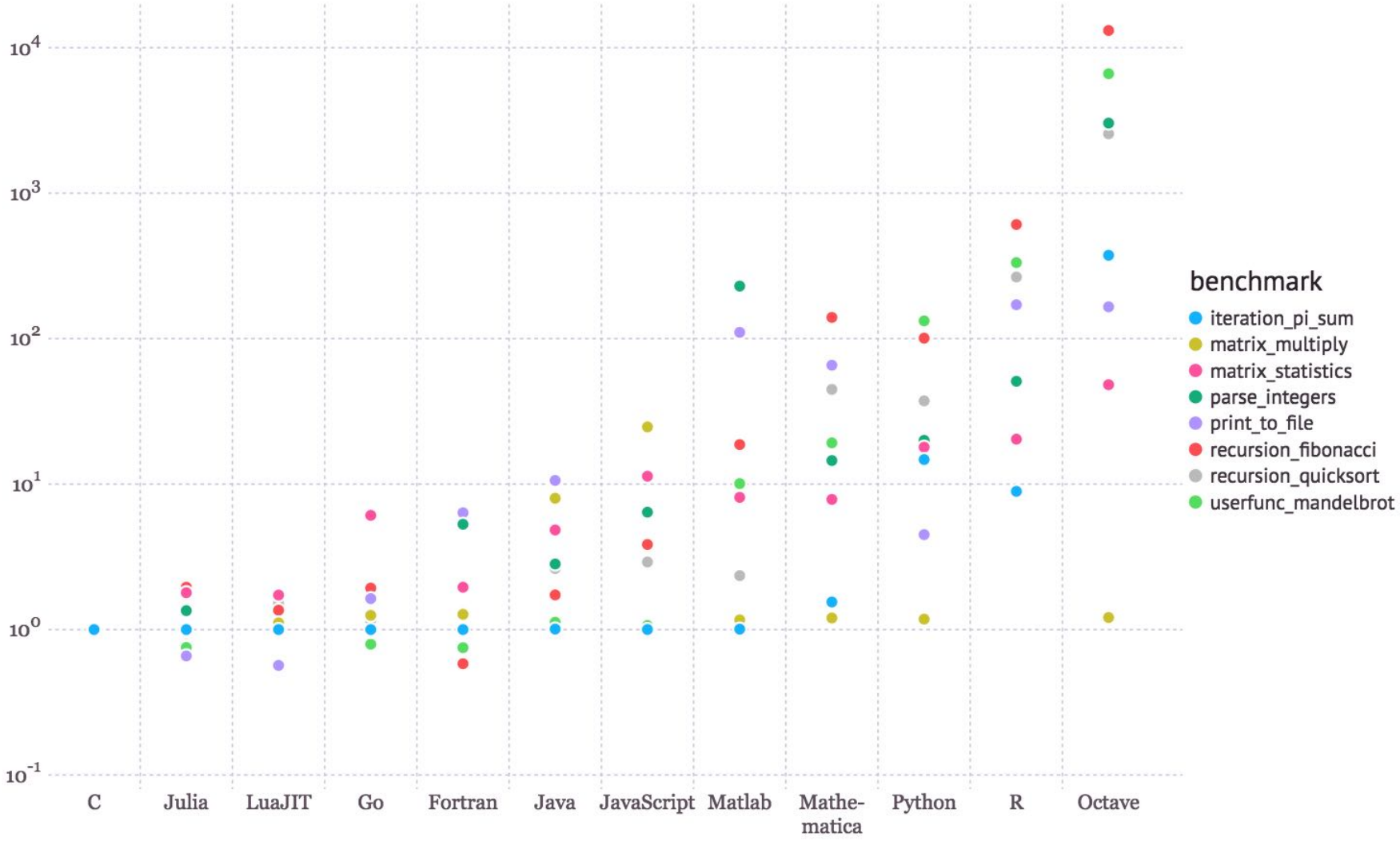


Interpreter vs Compiler

python™

JIT 없는 dynamic 인터프리터 언어

Why python?


10^4
10^3
10^2
10^1
10^0
10^{-1}
C
Julia
LuaJIT
Go
Fortran
Java
JavaScript
Matlab
Mathe-
matica
Python
R
Octave
benchmark
iteration_pi_sum
matrix_multiply
matrix_statistics
parse_integers
print_to_file
recursion_fibonacci
recursion_quicksort
userfunc_mandelbrot

Python 개발 환경

2020년까지 bugfix만, 지원 종료 예정

https://www.python.org/dev/peps/pep-0373/

REPL vs IDE vs Text Editor

don't need
to use
the print() function

Interactive Mode
playground

https://en.wikipedia.org/wiki/Read–eval–print_loop

REPL

IDE

Text Editor

vi, emacs...

■ Online

- ○ Python.org Online Console: www.python.org/shell
- ○ Python Fiddle: pythonfiddle.com
- ○ Repl.it: repl.it
- ○ Trinket: trinket.io
- ○ Python Anywhere: www.pythonanywhere.com
- ○ codeskulptor : http://www.codeskulptor.org/viz/index.html, http://py3.codeskulptor.org/
- ○ http://www.pythontutor.com/

■ iOS (iPhone / iPad)

- ○ Pythonista app

■ Android (Phones & Tablets)

- ○ Pydroid 3

■ pip

○ 모듈이나 패키지를 쉽게 설치할 수 있도록 도와주는 도구 (De Facto)

- 설치된 패키지 확인
  pip list / pip freeze
  pip show '패키지 이름'
- 패키지 검색
  pip search '패키지 이름'
- 패키지 설치
  pip install '패키지 이름'
- 패키지 업그레이드
  pip intall –upgrade '패키지 이름' pip install –U '패키지 이름'
- 패키지 삭제
  pip uninstall '패키지 이름'

■ conda

■ 윈도우 라이브러리

○ https://www.lfd.uci.edu/~gohlke/pythonlibs/

■ Distribution = a software bundle

○ 용량 문제, 관리 문제, 충돌 문제, 버전 호환성 등의 문제를 해결 가능

○ Python interpreter (Python standard library) + package managers

Anaconda

ActivePython

Enthought Canopy

python(x,y)

■ 가상환경

○ Application-level isolation of Python environments

- virtualenv / venv
  - VirtualenvWrapper
- buildout
  - http://www.buildout.org/en/latest/

○ System-level environment isolation

- docker
  - https://djangostars.com/blog/what-is-docker-and-how-to-use-it-with-python/
  - https://www.pycon.kr/2015/program/71

■ Packing isolation

- pipenv
  - dependencies를 간단하게 관리

■ **REPL 실행**

○ 대화형 모드 - 인터프리터

○ $ python3

■ **file.py 파일 실행**

○ $ python3 file.py

■ **"print('hi')" 문 실행**

○ $ python3 -c "print('hi')"

■ **Execute the file.py file, and drop into REPL with namespace of file.py:**

○ $ python3 -i file.py

■ **json/tool.py 모듈 실행**

○ $ python3 -m json.tool

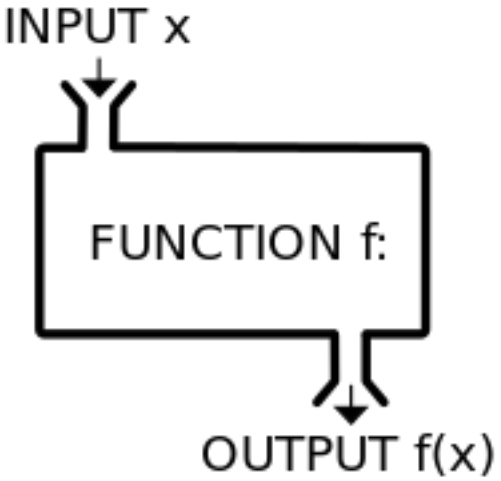
■ 대화형 환경에서는, 마지막에 인쇄된 표현식은 변수 _ 로 표현 가능

○ tax = 12.5 / 100
price = 100.50
price * tax 12.5625
price + _ 113.0625
round(_, 2) 113.06

프로그래밍의 구성 요소

프로그래밍

문법(키워드, 식, 문)을 이용해서
값을 **입력**받고, **계산/변환**하고, **출력**하는 **흐름을 만드는 일**



The Zen of Python (PEP 20)

import this

프로그래밍 언어의 문법
- 생각을 표현해내는 도구인 동시에, 생각이 구체화되는 틀
- 언어가 지향하고자 하는 **철학**에 따라 고안

문법에 대한 올바른 이해는 프로그래밍을 위한 필수적인 과정

BDFL(Benevolent Dictator For Life!)
자비로운 종신 독재자 : Guido van Rossum, 파이썬의 창시자
https://en.wikipedia.org/wiki/Benevolent_dictator_for_life

- 아름다운 것이 보기 싫은 것보다 좋다.
- 명시적인 것이 암묵적인 것보다 좋다.
- 간단한 것이 복합적인 것보다 좋다.
- 복합적인 것이 복잡한 것보다 좋다.
- 수평한 것이 중첩된 것보다 좋다.
- 희소한 것이 밀집된 것보다 좋다.
- 가독성이 중요하다.
- 규칙을 무시할 만큼 특별한 경우는 없다.
- 하지만 실용성이 순수함보다 우선한다.
- 에러가 조용히 넘어가서는 안된다.
- 명시적으로 조용히 만든 경우는 제외한다.
- 모호함을 만났을 때 추측의 유혹을 거부해라.
- 하나의 — 가급적 딱 하나의 — 확실한 방법이 있어야 한다.
- 하지만 네덜란드 사람(귀도)이 아니라면 처음에는 그 방법이 명확하게 보이지 않을 수 있다.
- 지금 하는 것이 안하는 것보다 좋다.
- 하지만 안하는 것이 이따금 지금 당장 하는 것보다 좋을 때가 있다.
- 설명하기 어려운 구현이라면 좋은 아이디어가 아니다.
- 설명하기 쉬운 구현이라면 좋은 아이디어다.
- 네임스페이스는 아주 좋으므로 더 많이 사용하자!

False	elif	lambda
None	else	nonlocal
True	except	not
and	finally	or
as	for	pass
assert	from	raise
break	global	return
class	if	try
continue	import	while
def	in	with
del	is	yield

import keyword
identifier로 사용하지 않음

Expression vs Statement

표현식 vs 구문

Declaration vs Assignment

선언 vs 할당(대입)

■ 표현식(expression) = 평가식 = 식

○ **값**

○ **값들과 연산자를 함께 사용해서 표현**한 것

○ 이후 "평가"되면서 하나의 특정한 결과값으로 축약

- 수
  - 1 + 1
    - » 1 + 1 이라는 표현식은 평가될 때 2라는 값으로 계산되어 축약
  - 0 과 같이 값 리터럴로 값을 표현해놓은 것
- 문자열
  - 'hello world'
  - "hello" + ", world"
- 함수
  - lambda (일반적으로 Functional Paradigm에서 지원)

○ 궁극적으로 "평가"되며, 평가된다는 것은 결국 하나의 값으로 수렴한다는 의미

- **python에서는 기본적으로 left-to-right로 평가**

■ 구문(statement) = 문

○ 예약어(reserved word, keyword)와 표현식을 결합한 패턴

○ 컴퓨터가 수행해야 하는 하나의 단일 작업(instruction)을 명시.

- **활당(대입, assigning statement)**
  - python에서는 보통 '바인딩(binding)'이라는 표현을 씀, 어떤 값에 이름을 붙이는 작업.
- **선언(정의, declaration)**
  - 재사용이 가능한 독립적인 단위를 정의. 별도의 선언 문법과 그 내용을 기술하는 블럭 혹은 블럭들로 구성.
    - » Ex) python에서는 함수나 클래스를 정의
  - 블럭
    - » 여러 구문이 순서대로 나열된 덩어리
    - » 블럭은 여러 줄의 구문으로 구성되며, 블럭 내에서 구문은 위에서 아래로 쓰여진 순서대로 실행.
    - » 블럭은 분기문에서 조건에 따라 수행되어야 할 작업이나, 반복문에서 반복적으로 수행해야 하는 일련의 작업을 나타낼 때 사용하며, 클래스나 함수를 정의할 때에도 쓰임.
- **조건(분기)** : 조건에 따라 수행할 작업을 나눌 때 사용.
  - Ex) if 문
- **반복문** : 특정한 작업을 반복수행할때 사용.
  - Ex) for 문 및 while 문
- **예외처리**

■ 값에 대한 type이 중요함

■ 값에 따라 서로 다른 기술적인 체계가 필요

○ 지원하는 연산 및 기능이 다르기 때문

■ 컴퓨터에서는 이진수를 사용해서 값을 표현하고 관리

○ 정확하게 표현하지 못할 수가 있음

○ 숫자형 = numeric

- 산술 연산을 적용할수 있는 값
- 정수 : 0, 1, -1 과 같이 소수점 이하 자리가 없는 수. 수학에서의 정수 개념과 동일. (int)
- 부동소수 : 0.1, 0.5 와 같이 소수점 아래로 숫자가 있는 수. (float)
- 복소수 : **Python에서 기본적으로 지원**

○ 문자, 문자열

- 숫자 "1", "a", "A" 와 같이 하나의 낱자를 문자라 하며, 이러한 문자들이 1개 이상있는 단어/문장와 같은 텍스트
- Python에서는 낱자와 문자열 사이에 구분이 없이 기본적으로 str 타입을 적용
  - byte, bytearry

○ 불리언 = boolean

- 참/거짓을 뜻하는 대수값. 보통 컴퓨터는 0을 거짓, 0이 아닌 것을 참으로 구분
- True 와 False 의 두 멤버만 존재 (bool)
- **Python에서는 숫자형의 일부**

○ Compound = Container = Collection

- 기본적인 데이터 타입을 조합하여, 여러 개의 값을 하나의 단위로 묶어서 다루는 데이터 타입
- 논리적으로 이들은 데이터 타입인 동시에 데이터의 구조(흔히 말하는 자료 구조)의 한 종류. 보통 다른 데이터들을 원소로 하는 집합처럼 생각되는 타입들
- Sequence
  - list : 순서가 있는 원소들의 묶음
  - tuple : 순서가 있는 원소들의 묶음. 리스트와 혼동하기 쉬운데 단순히 하나 이상의 값을 묶어서 하나로 취급하는 용도로 사용
  - range
- Lookup
  - mapping
    - » dict : 그룹내의 고유한 이름인 키와 그 키에 대응하는 값으로 이루어지는 키값 쌍(key-value pair)들의 집합.
  - set : 순서가 없는 고유한 원소들의 집합.

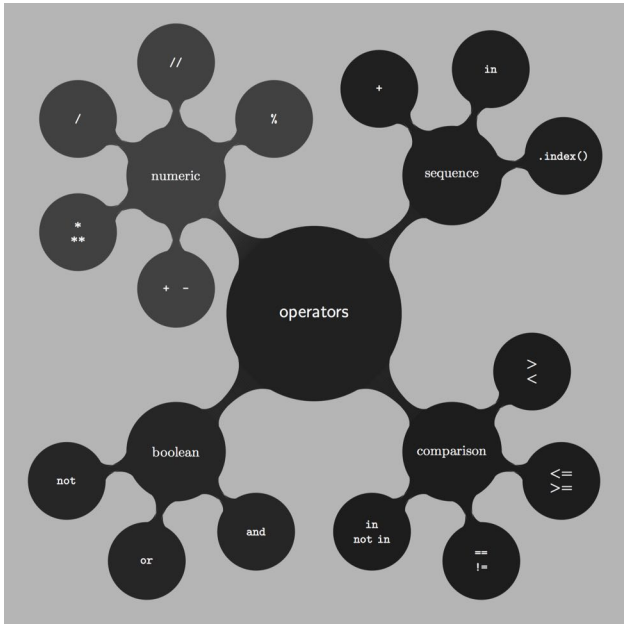
○ None

- 존재하지 않음을 표현하기 위해서 "아무것도 아닌 것"을 나타내는 값
- 어떤 값이 없는 상태를 가리킬만한 표현이 마땅히 없기 때문에 "아무것도 없다"는 것으로 약속해놓은 어떤 값을 하나 만들어 놓은 것
- None 이라고 대문자로 시작하도록 쓰며, 실제 출력해보아도 아무것도 출력되지 않음.
- 값이 없지만 False 나 0 과는 다르기 때문에 어떤 값으로 초기화하기 어려운 경우에 쓰기도 함

Literal vs Type(Object-oriented)

값, 기호로 고정된 값을 대표
간단하게 값 생성

Operation



■ 산술연산

○ 계산기

■ 비교연산

○ 동등 및 대소를 비교. 참고로 '대소'비교는 '전후'비교가 사실은 정확한 표현

○ 비교 연산은 숫자값 뿐만 아니라 문자열 등에 대해서도 적용할 수 있음.

■ 비트연산

■ 멤버십 연산

○ 특정한 집합에 어떤 멤버가 속해있는지를 판단하는 것으로 비교연산에 기반을 둠

○ is, is not : 값의 크기가 아닌 값 자체의 정체성(identity)이 완전히 동일한지를 검사

○ in, not in : 멤버십 연산. 어떠한 집합 내에 원소가 포함되는지를 검사 ('a' in 'apple')

■ 논리연산

○ 비교 연산의 결과는 보통 참/거짓. 이러한 불리언값은 다음의 연산을 적용. 참고로 불리언외의 타입의 값도 논리연산을 적용

	Operator	Description
lowest precedence	or	Boolean OR
	and	Boolean AND
	not	Boolean NOT
	in, not in	membership
	==, !=, <, <=, >, >=, is, is not	comparisons, identity
	|	bitwise OR
	^	bitwise XOR
	&	bitwise AND
	<<, >>	bit shifts
	+, -	addition, subtraction
	*, /, //, %	multiplication, division, floor division, modulo
	+x, -x, ~x	unary positive, unary negation, bitwise negation
highest precedence	**	exponentiation

PEMDAS :

Parentheses - **E**xponentiation - **M**ultiplication - **D**ivision - **A**ddition - **S**ubtraction