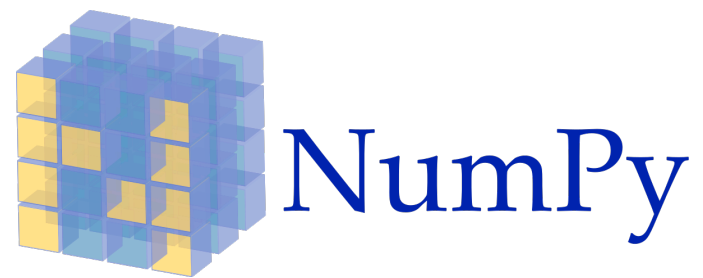




Python 1

Jin Hyun Kim

Machine Learning with Python



Python 설치

- Anaconda 설치 (<https://www.anaconda.com/>)
- Matplotlib 설치 (<https://anaconda.org/conda-forge/matplotlib>)

The screenshot shows the Anaconda website's 'Anaconda Distribution' page. The header includes the Anaconda logo and navigation links: Products, Why Anaconda?, Solutions, Resources, Company, Contact Us, and Download. The main section features the title 'Anaconda Distribution' and the subtitle 'The World's Most Popular Python/R Data Science Platform' with a 'Download' button. Below this, a paragraph describes the open-source distribution as the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. A list of benefits follows, including quick download of 7,500+ packages, library management with Conda, and support for various machine learning frameworks. A grid of logos for various data science libraries is displayed, including Jupyter, Spyder, NumPy, SciPy, Numba, pandas, Dask, Bokeh, HoloViews, Datashader, matplotlib, sklearn, H2O.ai, TensorFlow, and CONDA. At the bottom, there are tabs for Windows, macOS, and Linux. The 'Windows' tab is selected, showing the 'Anaconda 2019.10 for Windows Installer' section. This section offers two download options: 'Python 3.7 version' and 'Python 2.7 version'. Each option has a 'Download' button and links to 64-bit and 32-bit graphical installers with their respective sizes.

ANACONDA

Products Why Anaconda? Solutions Resources Company Contact Us Download

Anaconda Distribution

The World's Most Popular Python/R Data Science Platform Download

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 19 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 7,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

jupyter spyder NumPy SciPy Numba

pandas DASK Bokeh HoloViews Datashader

matplotlib sklearn H2O.ai TensorFlow CONDA

Windows macOS Linux

Anaconda 2019.10 for Windows Installer

Python 3.7 version

Download

64-Bit Graphical Installer (462 MB)
32-Bit Graphical Installer (410 MB)

Python 2.7 version

Download

64-Bit Graphical Installer (413 MB)
32-Bit Graphical Installer (356 MB)

Python 설치

- Matplotlib 설치 (<https://anaconda.org/conda-forge/matplotlib>)

```
Anaconda Prompt (Anaconda3) - conda install -c conda-forge matplotlib

(base) C:\Users\jhhkim>conda install -c conda-forge matplotlib
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

environment location: C:\ProgramData\Anaconda3

added / updated specs:
- matplotlib

The following packages will be downloaded:

package | build | size | channel
-----|-----|-----|-----
ca-certificates-2019.11.28 | hecc5488_0 | 182 KB | conda-forge
certifi-2019.11.28 | py37_0 | 148 KB | conda-forge
conda-4.8.2 | py37_0 | 3.0 MB | conda-forge
openssl-1.1.1d | hfa6e2cd_0 | 4.7 MB | conda-forge
-----|-----|-----|-----
Total: | 8.1 MB |

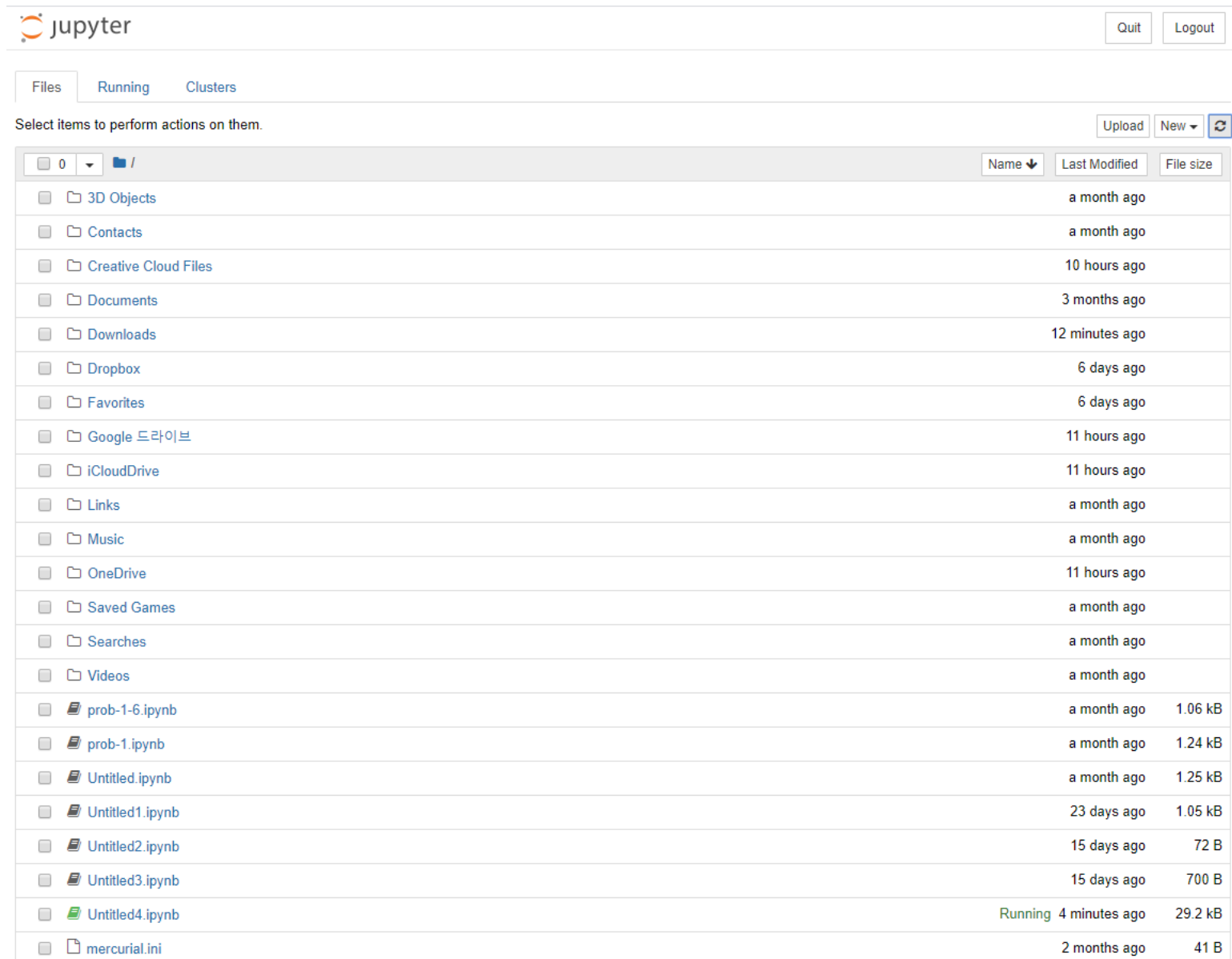
The following packages will be SUPERSEDED by a higher-priority channel:

ca-certificates pkgs/main::ca-certificates-2020.1.1-0 --> conda-forge::ca-certificates-2019.11.28-hecc5488_0
certifi pkgs/main --> conda-forge
conda pkgs/main --> conda-forge
openssl pkgs/main::openssl-1.1.1d-he774522_4 --> conda-forge::openssl-1.1.1d-hfa6e2cd_0

Proceed ([y]/n)? _
```

Jupyter 실행

- 시작 > Anaconda3 (64-bit) > Jupyter Notebook

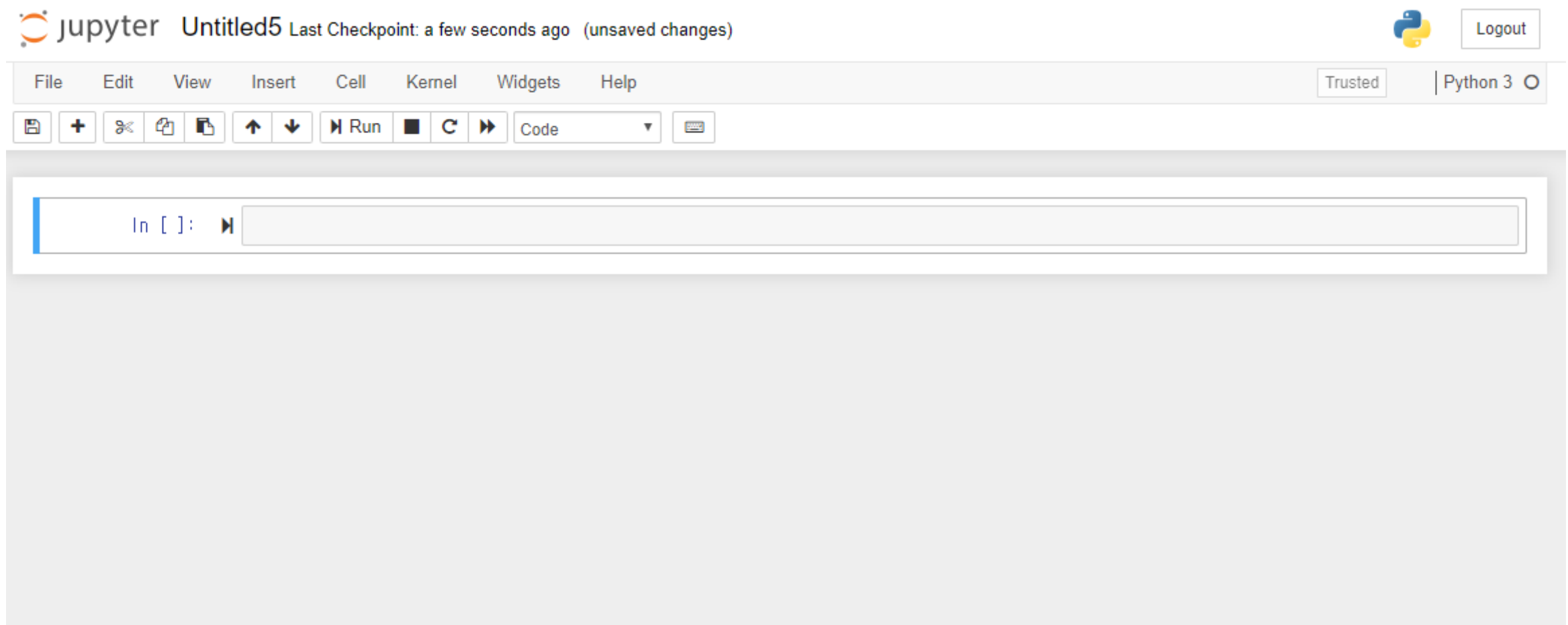


The screenshot displays the Jupyter Notebook web interface. At the top, the 'jupyter' logo is on the left, and 'Quit' and 'Logout' buttons are on the right. Below the logo, there are tabs for 'Files', 'Running', and 'Clusters'. The 'Files' tab is active, showing a file browser. The browser has a search bar with '0' and a dropdown arrow, and a breadcrumb path '/'. To the right of the search bar are buttons for 'Upload', 'New', and a refresh icon. Below these are columns for 'Name', 'Last Modified', and 'File size'. The file list includes various system folders like '3D Objects', 'Contacts', 'Creative Cloud Files', 'Documents', 'Downloads', 'Dropbox', 'Favorites', 'Google 드라이브', 'iCloudDrive', 'Links', 'Music', 'OneDrive', 'Saved Games', 'Searches', and 'Videos'. It also lists several Jupyter Notebook files (.ipynb) with names like 'prob-1-6.ipynb', 'prob-1.ipynb', 'Untitled.ipynb', 'Untitled1.ipynb', 'Untitled2.ipynb', 'Untitled3.ipynb', and 'Untitled4.ipynb'. The 'Untitled4.ipynb' file is highlighted in green and labeled 'Running' with a status of '4 minutes ago' and a size of '29.2 kB'. At the bottom, there is a file named 'mercurial.ini' with a size of '41 B'.


Name	Last Modified	File size
3D Objects	a month ago	
Contacts	a month ago	
Creative Cloud Files	10 hours ago	
Documents	3 months ago	
Downloads	12 minutes ago	
Dropbox	6 days ago	
Favorites	6 days ago	
Google 드라이브	11 hours ago	
iCloudDrive	11 hours ago	
Links	a month ago	
Music	a month ago	
OneDrive	11 hours ago	
Saved Games	a month ago	
Searches	a month ago	
Videos	a month ago	
prob-1-6.ipynb	a month ago	1.06 kB
prob-1.ipynb	a month ago	1.24 kB
Untitled.ipynb	a month ago	1.25 kB
Untitled1.ipynb	23 days ago	1.05 kB
Untitled2.ipynb	15 days ago	72 B
Untitled3.ipynb	15 days ago	700 B
Untitled4.ipynb	Running 4 minutes ago	29.2 kB
mercurial.ini	2 months ago	41 B

Jupyter 실행

- New > Python 3



Hello World 실행

 jupyter Untitled5 Last Checkpoint: 2 minutes ago (autosaved)

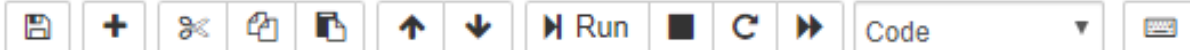


Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3 



```
In [ ]: ▶ print ("Hello World")
```

Matplotlib 실행

jupyter Untitled5 Last Checkpoint: 4 minutes ago (unsaved changes)



Logout

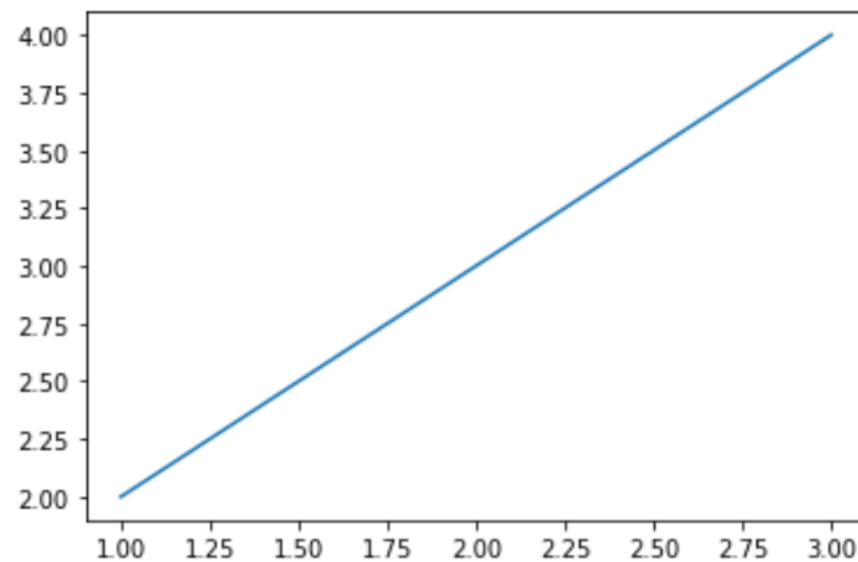
File Edit View Insert Cell Kernel Widgets Help

Trusted

Python 3



```
In [1]: ▶ from matplotlib import pyplot as plt  
plt.plot([1,2,3],[2,3,4])  
plt.show()
```



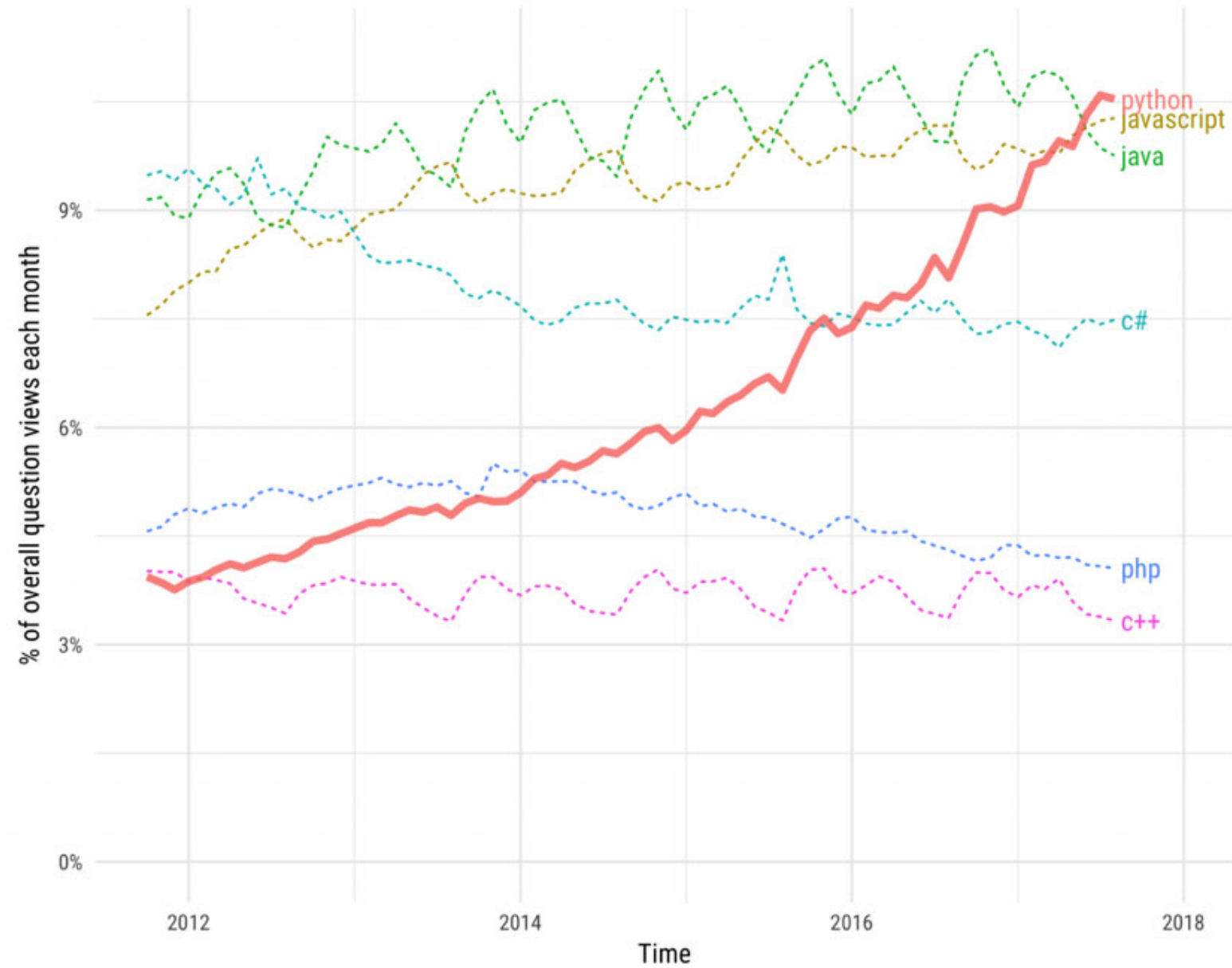
```
In [ ]: ▶
```


Learning Python

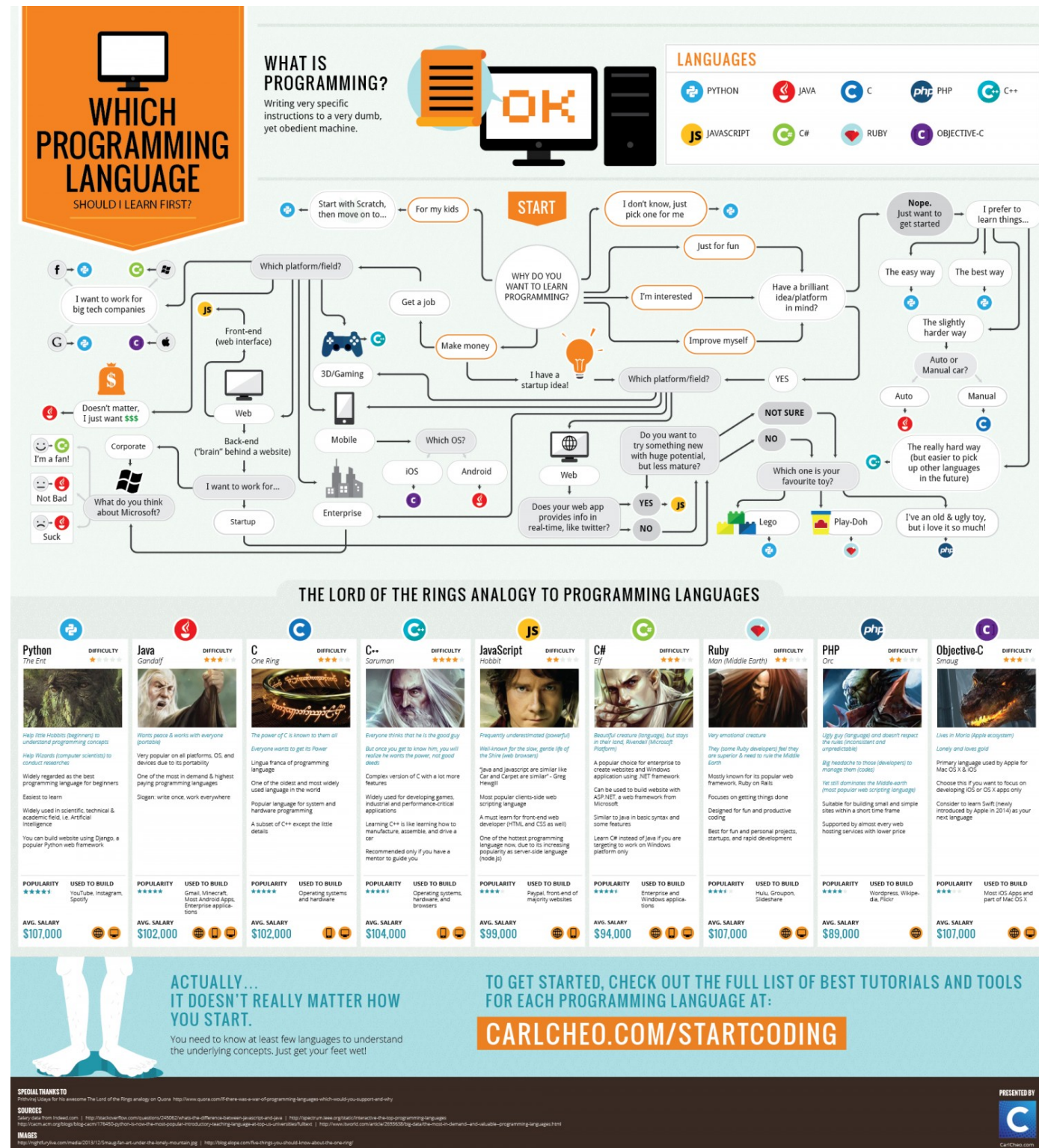
왜 파이썬?

Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



왜 파이썬?



왜 파이썬?

- Easy to Learn, Read, and Maintain
- 풍부한 라이브러리
 - Numpy, Pandas, Matplotlib, Lpython, Tensorflow, Keras, Scykit-learn
- 다양한 플랫폼
- 다양한 개발 응용
 - 시스템 유틸리티, GUI, C/C++ 연동, 웹 프로그래밍, 수치연산, DB,
 - 그러나 시스템 프로그래밍이나 , 모바일 프로그래밍은 아직

프로그래밍 언어의 기본 요소

- 변수, 대입, 조건, 반복 + 함수 (클래스, 내장함수...)

변수와 변수 형태

- 변수 Variable
 - a, B, cde, fg1, h2i
- 파이썬의 변수 이용(선언 + 이용)
 - `a = 1`, `a = 1.0`, `c = a`, `c = "a"`, `str = "abc"`, `str = 'abc'`
- 수 Number, 문자 Char, 문자열 String

변수 형태

Text Type:

`str`

Numeric Types:

`int`, `float`, `complex`

Sequence Types:

`list`, `tuple`, `range`

Mapping Type:

`dict`

Set Types:

`set`, `frozenset`

Boolean Type:

`bool`

Binary Types:

`bytes`, `bytearray`, `memoryview`

산술 연산자

연산 기호	뜻	예시	결과
+	더하기	$7+4$	11
-	빼기	$7-4$	3
*	곱하기	$7*4$	28
/	나누기	$7/4$	1.75
**	제곱 (같은 수를 여러 번 곱함)	$2**3$	8 (2를 세 번 곱함 $2*2*2$)
//	정수로 나누었을 때의 몫	$7//4$	1 (나눗셈의 몫)
%	정수로 나누었을 때의 나머지	$7\%4$	3 (나눗셈의 나머지)
()	다른 계산보다 괄호 안을 먼저 계산	$2*(3+4)$	14

변수선언

- 암묵적 (implicit)

d = 100 # 변수타입은 정수, 변수 d에 값 100을 저장

대입

» 대화형 셀에서 변수를 사용한 예제

```
>>> a = # 변수 a에 3을 저장
>>> # a 값을 확인
3
>>> b = 1.1+2 # 변수 b에 1.1+2의 결과인 3.1을 저장
>>> b # b 값을 확인
3.1
>>> c = a+b # a와 b를 합한 값을 변수 c에 저장
>>> c # c 값을 확인
6.1
>>> d = 2 # 변수 d에 2를 저장
>>> d = d+1 # d에 1을 더한 값을 다시 d에 저장
>>> d # d 값을 확인하면 3임
3
```

Print

- 기본 출력

```
1
2 x = 1
3 y = 2.8
4 z = 1j
5
6 print(1)
7 print(1.0)
8 print(x)
9 print(type(x))
10 print(y)
11 print(type(y))
12 print(z)
13 print(type(z))
```

```
$python main.py
1
1.0
1
<type 'int'>
2.8
<type 'float'>
1j
<type 'complex'>
```

조건분기

```
a = 33
b = 200
if b > a:
    print("b is greater than a")
```

```
a = 33
b = 200
if b > a:
    print("b is greater than a") # you will get an error
```

잠깐...파이썬에서는

- 들여쓰기 (indentation) 으로 구역을 구분

```
s = 0
x = 1
while x <= 10:
    s = s + x
    print("x:", x, " sum:", s)
    x = x + 1
```

```
x = input("12+23 = ")
a = int(x)
if a == 12+23:
    print("천재!")
else:
    print("바보?")
```

조건분기

```
a = 33
b = 200
if b > a:
    print("b is greater than a") # you will get an error
```

```
3 a = 33
4 b = 33
5 if b > a:
6     print("b is greater than a")
7 elif a == b:
8     print("a and b are equal")
9 else:
10    print("b is less than a")
```

비교 연산자

연산자	설명	예
==	양쪽이 같다(같으면 True, 다르면 False).	$3 == 3 \rightarrow \text{True}$ $1 == 7 \rightarrow \text{False}$
!=	양쪽이 다르다(다르면 True, 같으면 False).	$3 != 3 \rightarrow \text{False}$ $1 != 7 \rightarrow \text{True}$
<	왼쪽이 오른쪽보다 작다.	$3 < 7 \rightarrow \text{True}$ $3 < 3 \rightarrow \text{False}$
>	왼쪽이 오른쪽보다 크다.	$7 > 3 \rightarrow \text{True}$ $7 > 7 \rightarrow \text{False}$
<=	왼쪽이 오른쪽보다 작거나 같다.	$3 <= 7 \rightarrow \text{True}$ $3 <= 3 \rightarrow \text{True}$
>=	왼쪽이 오른쪽보다 크거나 같다.	$7 >= 3 \rightarrow \text{True}$ $7 >= 7 \rightarrow \text{True}$

논리 연산자

```
if 1 < 2 and 4 > 2:  
    print("condition met")  
  
if 1 > 2 and 4 < 10:  
    print("condition not met")  
  
if 4 < 10 or 1 < 2:  
    print("condition met")
```

```
x = False  
if not x :  
    print("condition met")  
else:  
    print("condition not met")
```


중첩조건분기

```
x = 41

if x > 10:
    print("Above ten,")
    if x > 20:
        print("and also above 20!")
    else:
        print("but not above 20.")
```

반복

- while

```
i = 1
while i < 6:
    print(i)
    i += 1
```

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

range

`range(5)` : 0, 1, 2, 3, 4로 값을 다섯 개 가짐.

`range(0, 5)` :

`range(a, b)`의 값은 a에서 시작해서b 바로 앞의 값까지 1씩 늘리면서 반복.

```
for x in range(5) :  
    print(x)
```

`range(0, 5)` :

0부터 시작해서 5 바로 앞의 값까지 반복
0, 1, 2, 3, 4를 출력.

`range(1, 11)` :

1에서 시작해서 11 바로 앞(10)까지를 반복
1, 2, 3, 4, 5, 6, 7, 8, 9, 10을 출력.

`range(1, 4)` : 1에서 4 바로 앞(3)까지 반복
1, 2, 3을 출력.

간단한 실습

- 1부터 10까지 숫자의 합계를 구하는 프로그램

```
s = 0
x = 1
while x <= 10:
    s = s + x
    print("x:", x, " sum:", s)
    x = x + 1
```

for + range

```
for x in range(5):  
    print(x)
```

range(5)로 0, 1, 2, 3, 4까지 다섯 번 반복
변수 x 값을 출력

```
for x in range(1, 11):  
    print(x)
```

1, 2, ..., 10까지 열 번 반복(11은 제외).
변수 x 값을 출력

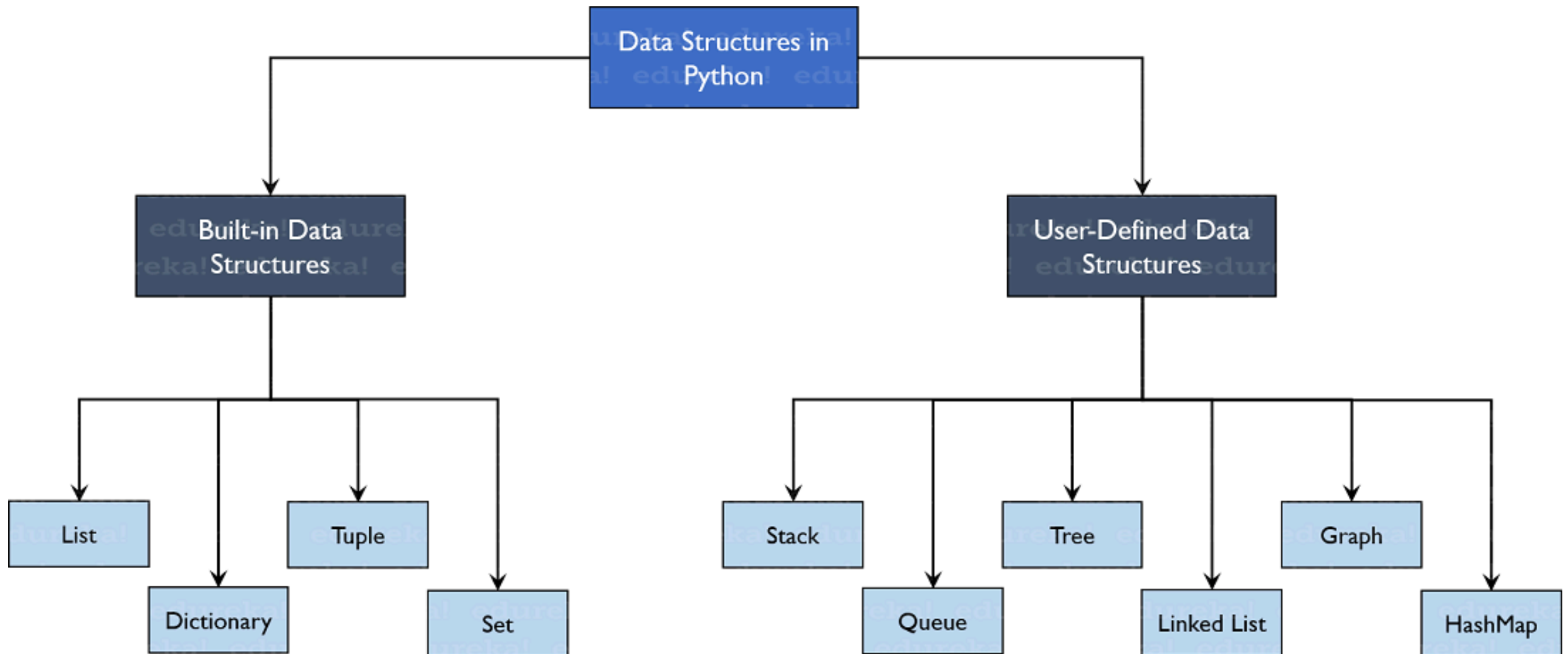
➤ range 명령어를 쓸 때 두 가지는 꼭 기억

- ① range(5) : 0부터 시작해서 4까지 다섯 번 반복한다(5는 제외한다).
- ② range(1, 11) : 1부터 시작해서 10까지 열 번 반복한다(11은 제외한다).

지금까지 ...

- 프로그램 언어의 기본 요소
 - 변수, 대입, 조건, 반복 + 함수 (클래스, 내장함수...)

파이썬의 자료구조



가장 유용한 자료구조: 리스트

- 리스트 만들기

```
1 my_list = [] #create empty list
2 print(my_list)
3 my_list = [1, 2, 3, 'example', 3.132] #creating list with data
4 print(my_list)
```

- 리스트 추가하기

```
1 my_list = [1, 2, 3]
2 print(my_list)
3 my_list.append([555, 12]) #add as a single element
4 print(my_list)
5 my_list.extend([234, 'more_example']) #add as different elements
6 print(my_list)
7 my_list.insert(1, 'insert_example') #add element i
8 print(my_list)
```


가장 유용한 자료구조: 리스트

- 리스트 지우기

```
1 my_list = [1, 2, 3, 'example', 3.132, 10, 30]
2 del my_list[5] #delete element at index 5
3 print(my_list)
4 my_list.remove('example') #remove element with value
5 print(my_list)
6 a = my_list.pop(1) #pop element from list
7 print('Popped Element: ', a, ' List remaining: ', my_list)
8 my_list.clear() #empty the list
9 print(my_list)
```

- 리스트 접근하기

```
1 my_list = [1, 2, 3, 'example', 3.132, 10, 30]
2 for element in my_list: #access elements one by one
3     print(element)
4 print(my_list) #access all elements
5 print(my_list[3]) #access index 3 element
6 print(my_list[0:2]) #access elements from 0 to 1 and exclude 2
7 print(my_list[::-1]) #access elements in reverse
```

가장 유용한 자료구조: 리스트

- 기타

```
1 my_list = [1, 2, 3, 10, 30, 10]
2 print(len(my_list)) #find length of list
3 print(my_list.index(10)) #find index of element that occurs first
4 print(my_list.count(10)) #find count of the element
5 print(sorted(my_list)) #print sorted list but not change original
6 my_list.sort(reverse=True) #sort original list
7 print(my_list)
```

입력처리

- 이름을 입력받아 Hello와 함께 보여 주는 프로그램

```
name = input("Your name? ")      # 이름을 입력받아 name 변수에 저장
print("Hello", name)             # Hello와 함께 name을 출력
```

- 실행결과

Your name? 김길벗

Hello 김길벗

입력처리

- 숫자 두 개를 입력 받아 곱하는 프로그램

```
x = input("?") # 변수 x에 첫 번째 입력을 받습니다. x = 문자열  
a = int(x)      # 문자열 x의 값을 정수(int)로 바꿔서 a에 넣음
```

```
x = input("?") # 변수 x에 두 번째 입력을 받습니다. x = 문자열  
b = int(x)      # 문자열 x의 값을 정수(int)로 바꿔서 b에 넣음
```

```
print(a * b)      # a와 b를 곱한 결과를 출력
```

- 실행결과

```
? 3  
? 7  
21
```

입력처리

- 속으로 20초를 세어 맞히는 프로그램

```
import time
```

```
input("엔터를 누르고 20초를 셉니다.")  
start = time.time()
```

```
input("20초 후에 다시 엔터를 누릅니다.")  
end = time.time()          # end 시간에서 start 시간을 빼면 실제 걸린 시간을 계산할 수 있음
```

```
et = end - start  
print("실제 시간 :", et, "초")  
print("차이 :", abs(et - 20), "초")
```

- 실행결과

```
엔터를 누르고 20초를 셉  
20초 후에 다시 엔터를 누름  
실제 시간 : 20.608863830566406 초  
차이 : 0.6088638305664062 초
```

초보자를 위한 노트

- 데이터 입력에 굳이 많은 노력을 기울이지 마라.
 - 하드코딩을 적극활용
- 복잡한 문법을 사용하지 마라.
 - 정확히 아는 문법과 의미 syntax and semantics 만을 사용하라.
- 실행 가능한 부분이 만들어지는 즉시 테스트하여 확인하라.
 - 점증적 프로그래밍 Incremental programming

정리

- 오늘...
 - 파이썬을 배운데 기본 요소...
 - 변수, 대입, 조건, 반복
- 다음 시간...
 - 파이썬의 함수
 - 수학을 위한 파이썬 이용

Python Part 2

In the last class,

- 변수, 대입, 조건, 반복 + 함수 (클래스, 내장함수...)
- Types, Assignments, Conditonal execution, Loop

This Class

- Function Implementation in Python

Referemces

- 모두를 위한 파이썬 – 이승찬, 길벗 – 2018

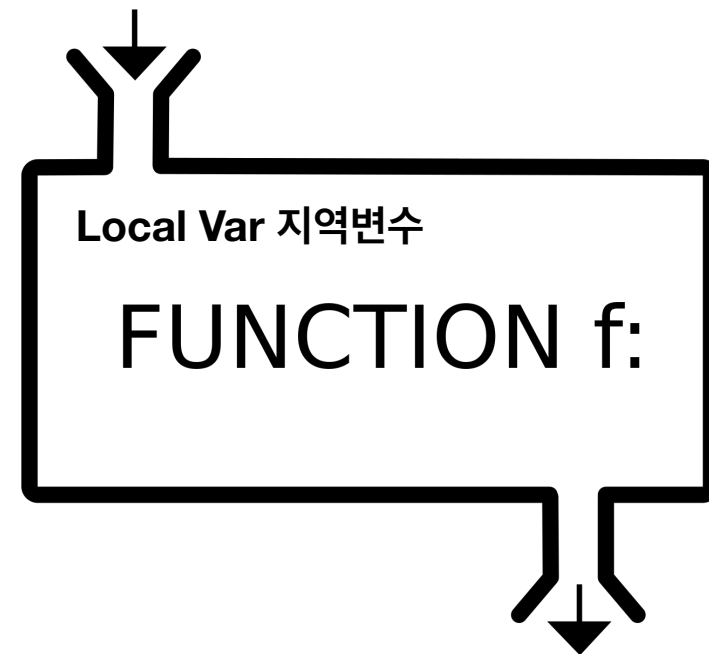
Funtions for What?

- Reuse of pre-defiend codes
 - Library
- Structure codes

Functions

$$y = f(x)$$

INPUT x Parameter 매개변수

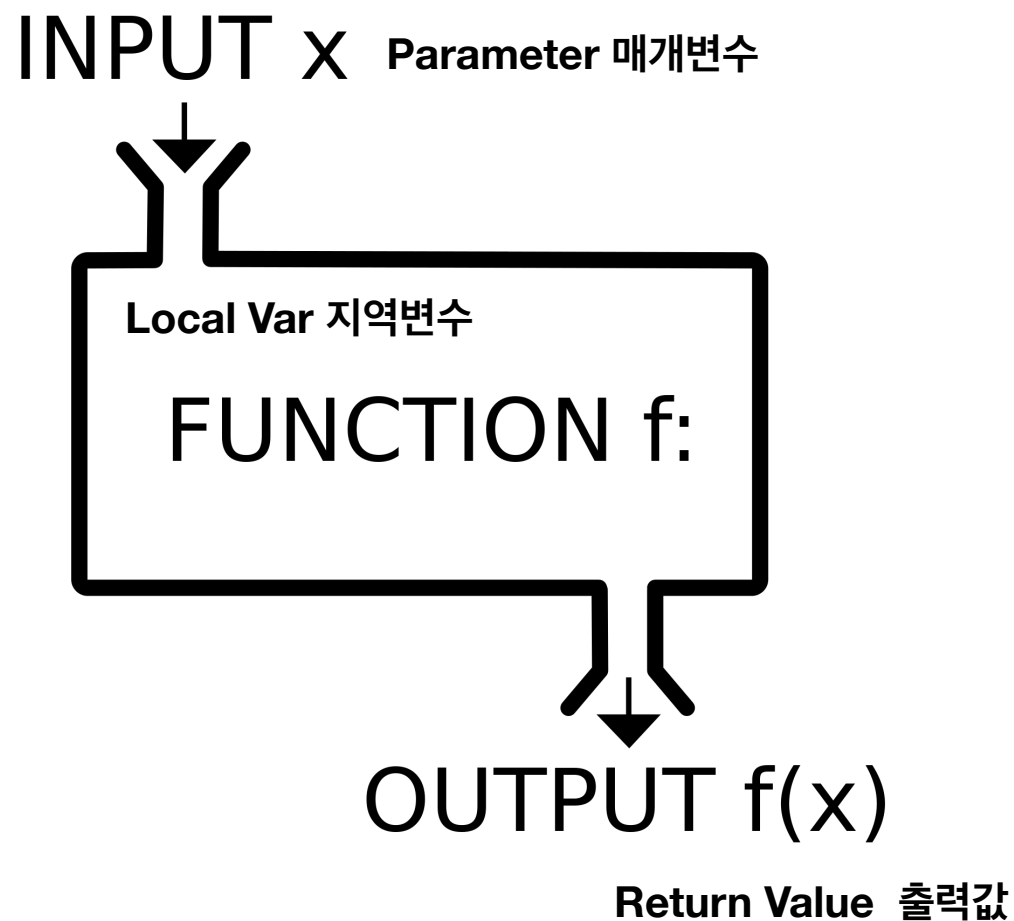


OUTPUT $f(x)$

Return Value 출력값

[https://en.wikipedia.org/wiki/Function_\(mathematics\)](https://en.wikipedia.org/wiki/Function_(mathematics))

Functions



- Every variable has a type
- Functions return values, of which type is called **“function type”**
- In other program languages, functions must be typed, i.e., their return values must be typed

Functions

- A function is defined 정의 to code how it works
- A function is declared 선언 so that it can be known to other functions
- A function is called 호출 for a function to use it

Function in Python

```
def hi():  
    print("Hello Python!")
```

hi() function definition
hi() function body

```
hi()  
hi()  
hi()
```

hi() is called

Function with Parameters

```
def hi(name):  
    print("Hi!" , name)
```

name is given

```
hi("Justin")
```

“Justin” is copied to the local

variable name and used in the function

```
hi("John")
```

```
hi("Mike")
```

Return

```
def square(a):  
    c = a * a  
    return c           # c is return to the function that calls square(a)  
  
def triangle(a, h):    # a, h are (formal) parameters,  
    c = a * h / 2  
    return c  
  
s1 = 4  
s2 = square(s1)        # square(s1) returns the output c ia s2. s1 is called real  
                        # parameter  
print(s1, s2)  
  
print(triangle(3, 4))  # print() shows the results from triangle(3,4) on display
```

Call by Assignment

- Basically, the parameter of Python functions is passed by copy.

```
def foo(a):  
    a = a + 1  
  
def foo_mu(b):  
    for i in range(len(b)):  
        b[i] = b[i] + 1  
  
a = 3  
foo(a)  
print(a)  
  
b = [1, 2]  
foo_mu(b)  
print(b)
```

3

[2, 3]

- Immutalbe object:
Variables in int, float, tuples types are passed by values
- Mutable object: Variables in list, dict, set types are passed by refereces

Conclusions

- Types
- Assignments
- Conditional execution
- Loop
- Functions

Mathematics with Python

What Can We with Python?

- Representation
- Computation
- Validation
- Reasoning

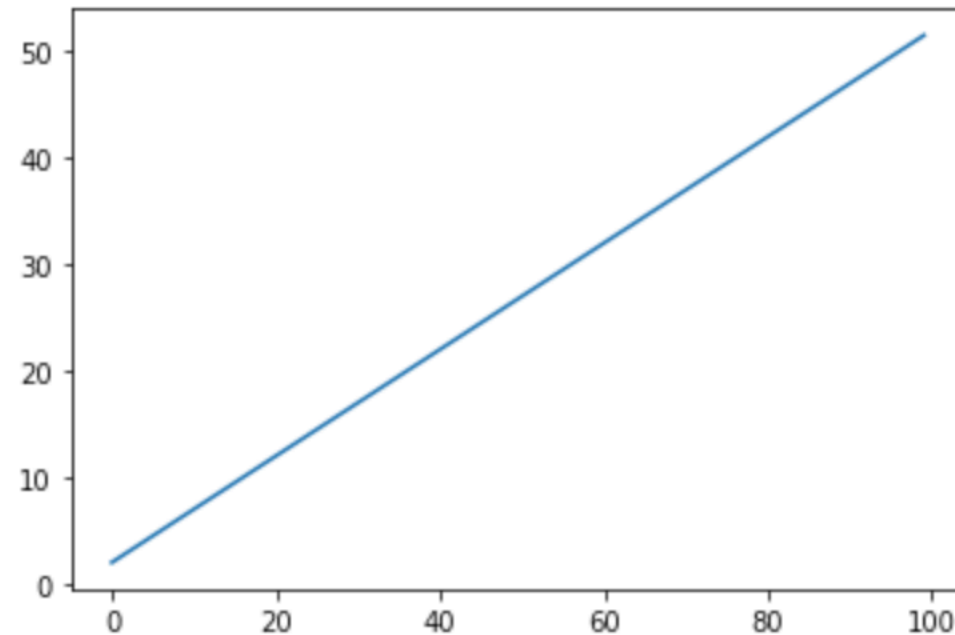
Functions

$$y = \frac{1}{2}x + 2$$

```
from matplotlib import pyplot

# y = 1/2*x + 2
x = []
y = []
for i in range(100):
    x.append(i)
    y.append((1/2)*i+2)

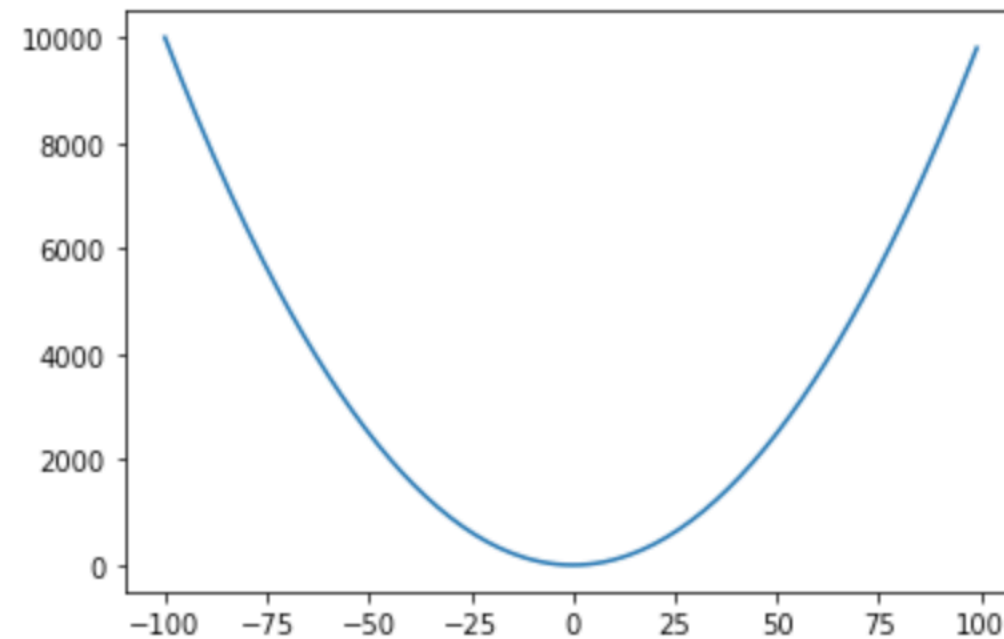
pyplot.plot(x, y)
pyplot.show()
```



Functions

$$y = x^2$$

```
# y = x**2  
  
x.clear()  
y.clear()  
  
for i in range(-100,100,1):  
    x.append(i)  
    y.append(i**2)  
  
pyplot.plot(x, y)  
pyplot.show()
```



Prime factorization

소인수분해

- For a given, the prime factorization computes **which prime numbers** multiply together to make the original number.

```
x = 45
d = 2

while d <= x:
    if x % d == 0:
        print(d)
        x = x / d
    else:
        d = d + 1
```

3
3
5

Probability of Throwing Dice

- Compute the probability of each number on dice

```
import random

total = 10000000
e1 = e2 = e3 = e4 = e5 = e6 = 0

for i in range(total):
    num = random.randint(1,6)
    if num == 1:
        e1 += 1
    if num == 2:
        e2 += 1
    if num == 3:
        e3 += 1
    if num == 4:
        e4 += 1
    if num == 5:
        e5 += 1
    if num == 6:
        e6 += 1
print("1:", e1/total)
print("2:", e2/total)
print("3:", e3/total)
print("4:", e4/total)
print("5:", e5/total)
print("6:", e6/total)
```

Assignment #1

- (15 Points) Write a Python program that finds the least common multiple (LCM) and the greatest common divisor for more than two numbers 2 개 이상의 수가 주어질 때, 최소공배수와 최대공약수를 구하는 프로그램을 작성하라.
- (10 Points) Use functions as many as possible 함수를 이용하여 프로그램을 작성하라.
- Use Jupyter and annotate each line of your codes with your explanation 주피터를 이용하되 각 라인마다 설명을 통해서 왜 그 라인이 필요한지 기술하라.
- (5 Points) 위의 문제에서 2개가 아닌 n개의 숫자를 받아들일 수 있도록 함수를 만들어 보아라

Assignment #2

- (10 points) For a quadratic equation, build a Python program that computes the solution of the quadratic equation, using the quadratic formula, 근의 공식을 이용한 2차 방정식의 해를 구하는 프로그램을 작성하라.
- Plot the quadratic equation using matplotlib and check if your solution from the above program is okay. 주어진 2차 방정식을 matplotlib로 그려보고 위에서 근의 공식으로 해결한 것과 같은 값을 가지는지 확인하라.

In Next Class

- Real Number and Complex number

In Next Class

- Introduction to ML, particularly, Perceptron
- Mathematics for ML