네가 만일 네 입으로 예수를 주로 시인하며 또 하나님께서 그를 죽은 자 가운데서 살리신 것을 네 마음에 믿으면 구원을 받으리라 사람이 마음으로 믿어 의에 이르고 입으로 시인하여 구원에 이르느니라 (롬10:9-10)

NOTE: The following materials have been compiled and adapted from the numerous sources including my own. Please help me to keep this tutorial up-to-date by reporting any issues or questions. Send any comments or criticisms to `idebtor@gmail.com` Your assistances and comments will be appreciated.

Welcome to "Python for Machine Learning"!

학습 목표

- 전반적인 파이썬의 개요를 살펴본다.
- 파이썬 문법을 살펴본다.

학습 내용

1. 파이썬 문법 개요
2. Shell 커멘드
3. 파이썬 연습 플랫폼

May I personally extend a warm welcome to you who are willing to come to this class. I hope that we can make this course of time will be memorable and enjoyable with a new spirit of learning together.

This is a short course on Python for students who have some experience of programming in a programming language other than Python. The course assumes that you know no Python whatsoever. Also, those people who already have considerable programming experience in several different programming languages are likely to be bored, and would be better off just working through the notes in their own time, or looking at one of the many on-line Python tutorials.

Python is named after Monty Python's Flying Circus, not the constricting snake.

Programming expectations

All the assignments and labs for this class will use Python and, for the most part, the browser-based IPython notebook format you are currently viewing as well as a command-line mode and Python Tools for Visual Studio when needed. Knowledge of Python is not a prerequisite for this course, **provided you are comfortable learning on your own as needed**. While we have strived to make the programming component of this course straightforward, we will not devote much time to teaching Python syntax itself. Basically, you should feel comfortable with:

- How to look up Python syntax on Google and StackOverflow.
- Basic programming concepts like functions, loops, arrays, dictionaries, strings, and if statements.
- How to learn new libraries by reading documentation.
- Asking questions on StackOverflow or Piazza.

There are many online tutorials to introduce you to Python training courses and scientific Python programming.

Introduction

Python - Easy to learn, but powerful

- Readability
- Simplicity
- Extensibility

Getting Python

You will be using Python 3.x or latest (not 2.x) throughout the course, including many popular 3rd party Python libraries or modules. There are many ways of installing Python.

For the program development environment, you may use any kind of IDE. Personally, I have reviewed or used a few IDE's such as Anaconda, Canopy, Eclipse, and PyCharm. At this point I ended up using Anaconda and 64 bit Python executable since my computer is 64-bit system. It comes with an easy-to-install bundle of Python and 3rd party libraries.

In this first class, we will be using a variety of data types of Python and understand the basic concepts of the language to ensure everything goes smoothly moving forward, While some of this will likely be dull, doing it now will enable us to do more exciting work in the days that follow without getting bogged down in further software development. It is essential that you complete it timely.

Python Installation Guide (if you don't use Anaconda)

- Refer to a web site: www.py4e.com
- https://www.py4e.com/lessons/install

MOOC & Textbook Available (in English and 한글 script)

We are going to use the one of the best Python course available through www.coursera.org or available for free at www.py4e.com.

The book "Python for Everybody" by Charles R. Severance available for free in many different forms including html and pdf.

- https://www.py4e.com/book
- over 200 pages long.

Coursera offers a week for free, and $49 per month when you subscribe it.

- You may earn a Coursera Certificate for this course

"Python for Everybody" is designed to introduce students to programming and software development through the lens of exploring data. This course consists of five parts as shown below:

1. Programming for Everybody (Getting Started with Python)
2. Python Data Structures
3. Using Python to Access Web Data
4. Using Databases with Python
5. Capstone: Retrieving, Processing, and Visualizing Data with Python

In this specialization, I expect you are to finish 4 out of 5 parts. I hope that you take this course along with a Certification specialized in Python.

파이썬 연습 플랫폼 - 파이썬 300문제 인강(한국어)

- https://wikidocs.net/book/922
- 300문제들 중에서 240번까지 주피터 노트북에 풀어보기를 권장합니다.
- 오늘 강의(Overview)가 지겨운 학습자는 파이썬300문제를 할 필요가 없습니다.
- 오늘 강의(Overview)가 무슨 말인지 알아듣지 못하는 사람은 300문제집에 꼭 풀어보아야 합니다.
- 기계학습/인공지능을 배우는 초기에는 파이썬 실력이 참 중요합니다.

Python or IPython?

IPython is short for **Interactive Python**. It does all things Python does and more. It provides a rich architecture for interactive computing with:

- A browser-based notebook with support for code, text, mathematical expressions, inline plots and other rich media.
- Support for interactive data visualization and use of GUI toolkits.
- Flexible, embeddable interpreters to load into your own projects.
- Easy to use, high performance tools for parallel computing. Once again all Python scripts presented here should work in IPython seamlessly, but also provides you with more functionalities and rich set of tools.

Overview

In this Jupyter Notebook, we will cover a wide variety of topics about the Python language to give you a broad overview of the data structures it offers, some use cases for them and the kinds of tasks they can be used for.

NOTE: It isn't expected for new comers to be able to follow all the details presented here. This fast-paced overview will be followed with a slower in-depth coverage of all these topics.

NOTE: The following materials have been collected from the numerous sources including my own and my students over the years of teaching and experiences of programming. Please help me to keep this tutorial up-to-date by reporting any issues or questions. Send any comments or criticisms to `idebtor@gmail.com` . Your assistances and comments will be appreciated.

Two kinds of cell

Jupyter Notebook consists of a series of so-called cells. There are two types of cells that contains either code or document. Cells that contains code is called **code cell**, whereas cell that contains document is called **markdown cell**.

By default, the notebook creates a code cell. Let's explore working with code cells. You may convert a code cell into a markdown cell by typing a short-cut `<Esc> m` , `<Esc> y` vice versa.

1. **Code cells** - contains python codes and ipython magic.
2. **Markdown cells** - Contains documents written in Markdown which is a lightweight markup language with plain-text-formatting syntax.

Two kinds of mode

1. **Command mode** - Tapping `<Esc>` let the cell into a **Command Mode**
2. **Edit mode** - Tapping `<Enter>` let the cell into **Edit Mode**.

Cell menu

The "Cell" menu has a number of menu items for running code in different ways. These includes:

- Run and Select Below
- Run and Insert Below
- Run All
- Run All Above
- Run All Below

Running Cells

Jupyter notebook cell is capable of running code in a wide range of languages. However, the default kernel in Jupyter notebook runs Python code. If the cell is a markdown cell, it just randers into webpage.

Run a cell using `<Shift Enter>` or "Run Cell" button in the toolbar above:

There are two other keyboard shortcuts for running code:

- `Shift Enter` runs the current cell and move to the next cell.
- `Ctrl Enter` run the current cell and enters command mode.
- `Alt Enter` runs the current cell and inserts a new one below.

Run OS or magic commands in code cell

IPython uses ! to run shell commands. Jupyter notebook provides shell-like magic functions as well. The followind commands available are shown below:

```
%cd, %cat, %cp, %env, %ls, %man, %mkdir, %more, %mv, %pwd, %rm, %rmdir
```

These commands can be used without the `%` sign if automagic is on. This makes it so that you can almost treat the IPython prompt as if it's a normal shell:

- Use `%` to run the OS command in code cell.
- You even omit `%` since, automagic option is on by default.
- **Some developers** prefer to using `!` instead of `%` since IPython uses `!` for a long time.

Print the current folder name (present working directory).

- Try `pwd` with % and without %, or with !.
- `pwd` means `present working directory` or `print working directory` .
- For example: `!pwd`

In [ ]:

Print the list of files in the current folder.

- Try it with `!ls` and `!dir`
- For example `!dir`

In [ ]:

Print the contents of a file.

- Try it with `%more <filename>`
- For example, `%more _start_jupyter.bat`

Note:
OS나 개발 환경 설정에 관계없이 라인 매직 코멘드 `%more` 를 사용하면 문서(~.txt)파일을 출력해 볼 수 있습니다. `%%cmd` 혹은 `%%bash` 셀 매직을 시작한 다음, 해당 OS에 맞는 명령어를 사용해 보십시오.

```
%%cmd
type _start_jupyter.bat
```

```
%%bash
cat _start_jupyter.bat
```

In [2]:
```
%more _start_jupyter.bat
```

```
rem -- start jupyter notebook here .bat file
pwd
rem --jupyter notebook
jupyter-lab
pause
```

In [12]:
```
%%bash
cat _start_jupyter.bat
```

```
rem -- start jupyter notebook here .bat file
pwd
rem --jupyter notebook
jupyter-lab
pause
```

In [1]:
```
%%cmd
type _start_jupyter.bat
```

```
Microsoft Windows [Version 10.0.22000.493]
(c) Microsoft Corporation. All rights reserved.

C:₩GitHub₩JoyAlx₩jupyter>type _start_jupyter.bat
rem -- start jupyter notebook here .bat file
pwd
rem --jupyter notebook
jupyter-lab
pause

C:₩GitHub₩JoyAlx₩jupyter>
```

Run Python code in code cell

Arithmetic

Let's start simple with some arithmetic...

Example 1:

Add two integer numbers.

In [ ]:
```
```

Example 2:

Add two floating pointer numbers.

In [ ]:
```
```

Example 3:

If you add an integer to a decimal, you will get a floating point number.

In [ ]:

Example 4:

How about computing `2**8, 2**16, 2**32` ?

In [ ]:

Example 5:

True division vs. Floor division

- **Note**: Python 2.x vs Python 3.x - They are not compatable

In [ ]:

Variables (or names)

You can save your value into a name called **'variable(변수)'**.

In [ ]:
```
# Save the values 10 and 5 in the variables 'a' and 'b', respectively.
# print a, b

a = 10
b = 5
print(a + b)
print(a - b)
a * b
```

In [ ]:
```
# Now you can do some math with two variables.
# Add two numbers and save the result to another variable 'c'.
# Subtract two numbers (a - b) and save the result to another variable 'd'.
# print the results c and d
```

Getting an input from the user and save it

The varable 'name' stores a string value, not numeric value.

- use `input()` function to get an user's input.

Ask a name (`What is your name?`) and save it in a variable called `name` . Then, you print `Hello` and the name entered as shown below:

For example,

```
What is your name?
John
```

```
Hello John!
```

In [ ]:
```
print('What is your name?')
None
None
```

List

Python makes it easy to create lists, which can contain any kind of object. We can index these, concatenate them, and find the length, just as we can with strings. For more examples, refter to Python Tutorials

Example 1.

Create two lists, one with raibow colors and the other with a few prime numbers:

In [ ]:
```
colors = ['red', 'orange', 'yellow', 'green', 'blue', 'indigo', 'violet']
primes = [2, 3, 5, 7, 11, 13, 17]
```

Example 2.

Make a new list called `mixed` by adding two lists .

In [ ]:
```
None
None
```

Example 3:

How many elements are in the `mixed` list?

- Use `len()` function.
- The most built-in function returns a value or `None` .

In [ ]:
```

```

Exmaple 4.

What are the first and last elements in the `mixed` ?

In [ ]:
```

```

In [ ]:
```

```

Exmaple 5.

- Append the next prime number at the end of the `mixed` . Use `append()` **method**.
- Append two next prime numbers at the end of the `mixed` . Use `extend()` **method**.

```

```

In [ ]:

Example 6:

- Sort the list.
  - Use `sort()` **method**.
  - Use `sorted()` **function**.
  - What is the difference between them?

In [ ]:

- Reading error messages is a good habit to have to your coding skill. Accept them joyfully.
- help(list)

In [ ]:
```
help(list)
```

In [ ]:
```
colors.sort()
print(colors)
```

Set

Python also provides the **set** data structure, which can be created using curly brackets.

- Set is a collection of unique objects. No duplicate are allowed.

In [ ]:
```
a = {0, 1, 2, 3, 4}
a
```

Example 1.

Add an element such as 0 or -1 in the set `a`. Make an observation.

In [ ]:

Example 2.

ake a new set `b` such that it has the some common elements in the set `a`.

In [ ]:
```
b = {2, 3, 4, 5, 6}
b
```

Example 3.

Intersection Operation

In [ ]:

Example 4.

Union Operation

In [ ]:
```
```

Reading a web page

In [ ]:
```
import requests
response = requests.get('http://www.handong.edu')
response.ok
```

Display the first 500 characters from the response. Use `response.text[500]`

In [ ]:
```
response.text[:500]
```

Read a remote file and process the text

In [ ]:
```
from urllib.request import urlopen
book = urlopen('http://www.gutenberg.org/10/10-0.txt')
#book = urlopen('http://composingprograms.com/shakespeare.txt')
```

Make a list words out of the book read from url.

In [ ]:
```
wlist = book.read().decode().split()
```

Example 1:

Get the number of words in `wlist`

Sample Run:

```
824533
```

In [ ]:
```
```

- Print the first 10 words in `wlist`
- Print the last 10 words in `wlist`

Sample Run:

```
['\ufeffThe', 'Project', 'Gutenberg', 'eBook', 'of', 'The', 'King',
'James', 'Bible', 'This']
['subscribe', 'to', 'our', 'email', 'newsletter', 'to', 'hear',
'about', 'new', 'eBooks.']
```

In [ ]:
```
```

Example3.

Sort words in `wlist`.

- Use the list's method called `sort()`
- To find methods available, Use after the class/variable name

Sample Run:

```
['"Defects,"', '"Information', '"Plain', '"Plain', '"Project',
'"Project', '"Project', '"Project', '"Project', '"Right']
```

In [ ]:

Example 4:

Counter the number of `life` in `wlist`.

- Use `count()` method.
- help(list.count)

Sample Run:

```
life: 197
truth: 97
```

In [ ]:
```
None
help(list.count)
```

Example 5.

Remove duplicated words from `wlist` and call it `ulist`.

Count the number of words in `ulist`. Convert `ulist` which is a set object into a list object `wlist`. Now sort `wlist` in increasing order, and print the first 10 words.

- Use `set()` construct returns unique elements in a set type which is unordered or a kind of bag. Change it back to a list using list() construct.
- Therefore, the set data type does not support indexing.

Sample Run

```
34024
['"Defects,"', '"Information', '"Plain', '"Project', '"Right', '#10]',
'$5,000)', "'AS-IS",", '("the', '($1']
```

In [ ]:

Example 6:

Try to print the first 10 words from `ulist`.

- Reading error messages is a good habit to have to improve your coding skills.
- Make an every effort to understand the error message if possible.

Sample Run:

```
---------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-32-e7a363738b60> in <module>
----> 1 ulist[:10]     # set is not subscriptable; read the error
message

TypeError: 'set' object is not subscriptable
```

In [ ]:
```
# print(ulist[:10]) # if ulist is a set, it fails
# set is not subscriptable; read the error message
```


- Reading error messages is a good habit to have to improve your coding skills.
- Make an every effort to understand the error message if possible.

Example 7:

Count a specific word `truth` and `life` in `wlist` again. Make sure that there is only one each.

Sample Run:

```
life: 1
truth: 1
```

In [ ]:
```

```

Functions

Now let's take a look at defining and using functions in Python. We will start by defining a function to evaluate a polynomial.

Defining your own function

- Define a function if you are using the same set of code more than once.
- Be careful when using `:` and `<tab>` characters.

Example 1.

Do the following task three times:

Say `Hello` to him/her whose name is passed as an argument.

In [ ]:
```
def greet(name):
    None
```

In [ ]:
```
greet(None)
greet(None)
greet(None)
```

- Do the same thing as shown above using a for loop.

In [ ]:
```

```

Example 2.

Draw a linear graph `y = x` or `x = 1, 2, 3, 4` and `y = 1, 2, 3, 4`

In [ ]:
```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 2, 3, 4])
plt.show()
```

Example 3.

Plot a polynomial from `x = -10` and `x = 10` .

- Its coefficients are `a = 1, b = 2, c = 3`
- Define a polynomial, a `poly(x, a, b, c)` function, `y = a * x^2 + b * x + c` that returns y

In [ ]:
```
def None
    return None
```

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

x = None           # np.arange()  from -10 to 10
y = None
plt.plot(x, y)

plt.show()
```

Example 4.

Draw `cos(x)` function from `x = 0` and `x = 2 * pi` .

- To check x values, evaluate x after generating numbers using np.arange()

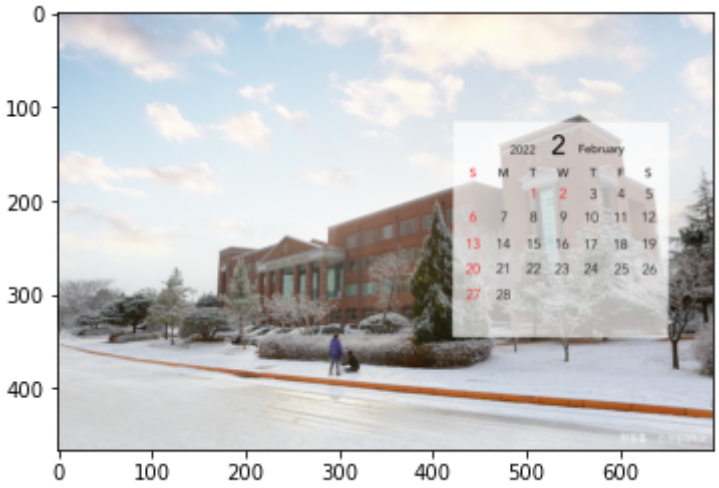
In [ ]:
```
x = None
y = None
plt.plot(x, y
plt.show()
```

Image Display

The pictures and more exotic objects can also be displayed, as long as their representation supports the IPython display protocol.

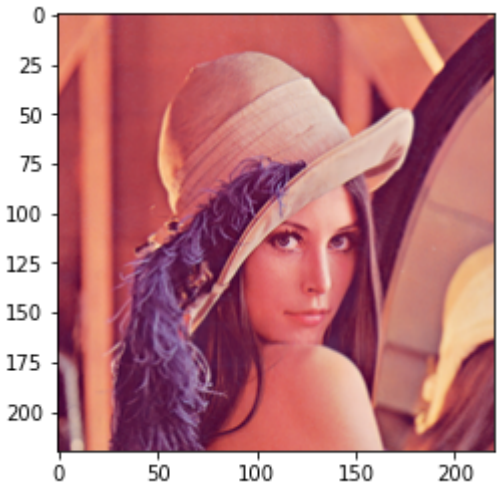
Case 1: If the image exists in local drive, it is much simpler.

In [1]:
```
# if the file exists in local, it is much simpler.
# just specify an image name with its folder name at imread() function
import matplotlib.pyplot as plt
import matplotlib.image as image

img = image.imread('./images/2202.png')            #img = imread('./images/lenna.png'
plt.imshow(img)
plt.show()
```



Case 2: If the image exists in the cloud, use the url to access the image.

In [1]:
```
import matplotlib.pyplot as plt
import urllib.request
import PIL

# add '?raw=true' to the end of the image filename if it is from GitHub
url = 'https://github.com/idebtor/KMOOC-ML/blob/master/ipynb/images/joyai/lenna.png?ra
img = PIL.Image.open(urllib.request.urlopen(url))
plt.imshow(img)
plt.show()
```



Class and Object

Now lets create a class. In Python, every class should derive from object. Our class will describe a person, with a name and an age. We will supply a constructor, and a method to get the full name.

- How to create a class: Refer to Wikibook

Example 1.

Create a Person class.

- It has `first` , `last` , and `age` as instance variables.
- It has a constructor that takes `first` , `last` and `age` as an initial parameters.
- It has a method `full_name` that returns the `first` and `last` name with a space between them.

In [ ]:
```
class Person(object):
    def __init__(self, first, last, age):
        self.first = first
        self.last = last
        self.age = age

    def full_name(self):
        return self.first + ' ' + self.last
```

Example 2.

Instantiate an instance, `student` . Now we can create an instance of a Person, and work with the attributes of the class.

In [ ]:
```
student = None
student.first
```

Example 3.

What is `student` 's last name?

In [ ]:
```

```

Example 4.

What is `student` 's full name?

In [ ]:
```

```

Example 5.

Change the last name to `Kim` .

In [ ]:
```

```

In Python, we can add new class attributes on the fly, even if they have not been defined yet.

In [ ]:
```
student.smoke = False
student.smoke
```

Writing a code cell into a file

You may use **%%writefile** cell magic command. You have to place the command at the very first line of the code cell.

```
%%writefile mycode.py
```

Loading external codes

- Drag and drop a .py in the dashboard
- Use **%load** with any local or remote url: the Matplotlib Gallery http://matplotlib.org/gallery.html

```
%load mycode.py
```

In this notebook we've kept the output saved so you can see the result, but you should run the next cell yourself (with an active internet connection).

That gives you a quick tour of the capabilities of IPython.

We covered a lot of ground here, but this should give you an idea of the power that Python puts at your fingertips. In the later lectures, we will give you a more in-depth tour of most of these features.

A simple way to start Jupyter

A simple way to start Jupyter-Lab or Jupyter notebook in Windows

To start an Jupyter Notebook or Jupyter-lab, you create a console window first, then you type "jupyter notebook" or "jupyter-lab' in the command-line prompt. Then you still need to navigate to file folders where your Jupyter files(~.ipynb) are. It is cumbersome.

I created a batch file (`_start_jupyter.bat`) as shown below and place it in the folder where I want to start Jupyter files from. In my case, I placed it in folders where my notebooks exist such as `~/jupyter` or `~/ipynb` . Then starting jupyter is **a double clicks away**. Enjoy it.

The batch filename I use is `_start_jupyter.bat` since the file is usually displayed at the top in the folder view because of the prefix `_` . The following script simply starts 'jupyter-lab' on a console. Edit the following to start 'jupyter notebook' instead of 'jupyter-lab'.

```
rem  start jupyter notebook here .bat file
rem  If ~.py file to be written as well, then use jupyter notebook --
script
pwd
```

```
rem jupyter notebook
jupyter-lab
pause
```

Shell

In computing, a shell is a user interface for access to an **operating system's services**. In general, operating system shells use either a command-line interface (CLI) or a graphical user interface (GUI), depending on a computer's role and particular operation. It is named a **shell** because it is the outermost layer around the operating system.

One problem we are facing when using Shell is that every OS has its own shell. They are all different, but there are aliases available we can use the same shell command-like across the OS's at convenience.

Some shell commands are often used.

- `cd mydir` changes directory to `mydir`.
- `cd ..` moves to the parent directory of current directory, or the directory one level up from the current directory. ".." represents parent directory.
- `cd ~` moves to directory to the home directory.
- `/` on its own is the root directory of the whole file system.
- `..` means 'the directory above the current one'; . on its own means 'the current directory'.
- A relative path specifies a location starting from the current location.
- An absolute path specifies a location from the root of the file system.
- Directory names in a path are separated with `/` on Unix/Linux/macOS, but `\` on Windows.

- `cp old new` copies a file.

- `mkdir path` creates a new directory.
- `mv old new` moves (renames) a file or directory.
- `rm path` removes (deletes) a file.
- `*` matches zero or more characters in a filename, so `*.txt` matches all files ending in `.txt`.
- `?` matches any single character in a filename, so `?.txt` matches `a.txt` but not `any.txt`.
- The shell does not have a trash bin: once something is deleted, it's really gone.

Exercise: Creating and deleting a folder and file.

1. Check the current directory and know where you are now.
2. Create a `myfolder` directory.
3. Change directory to `myfolder`.
4. Check the current working directory
5. Create a file called `hello.txt` which contains "Hello World".
6. Check that you have the file just created.
7. Check the file contents just created.

8. Delete the file and directory you just created.
9. Remove the directory `myfolder`.
    - `cd ..` first since you cannot remove the folder, then remove the folder.

In [ ]:
```
# a solution of Exercise shown above:

None
```

Closing and Tips on Jupyter-Lab or Jupyter Notebook

That gives you a quick tour of the capabilities of Jupyter-Lab or Jupyter notebook.

We covered a lot of ground here, but this should give you an idea of the power that Python puts at your fingertips. In the later lectures, we will give you a more in-depth tour of most of these features.

학습정리

1. 파이썬 문법 개요
2. Shell 커멘드
3. 파이썬 연습 플랫폼

참고자료

- python overview

Be joyful always! 1 Thes.5:16