

DMU*ai*

동양미래대학교 인공지능소프트웨어학과

인공지능 서비스
전문 소프트웨어 개발자 인재양성

Dongyang Mirae University
Dept. Of Artificial Intelligence

- 인공지능소프트웨어학과 학과장 교수
 - 연락처: 02-2610-1941
 - 연구실: 2호관 706호
 - E-mail: hskang@dongyang.ac.kr
 - Github Homepage
 - <https://github.com/ai7dnn>



수강생 목록

- 수강생 여러분께 칭찬의 말씀을 드립니다.

파이썬을 위한 다양한 개발환경 (07/12~07/15) 수강생 명단					
연번	학과	성명	학번	학년	전화번호
1	인공지능소프트웨어학과	강승희	20221655	1학년	010-2955-5691
2	인공지능소프트웨어학과	권민정	20222592	1학년	1082174130
3	인공지능소프트웨어학과	김호민	20222596	1학년	1020906625
4	인공지능소프트웨어학과	양희원	20221664	1학년	010-4104-1347
5	인공지능소프트웨어학과	이찬송	20221665	1학년	1084803613
6	컴퓨터정보공학과	김민재	20221578	1학년	1023399551
7	컴퓨터정보공학과	이조은	20221638	1학년	1050916371
8	컴퓨터소프트웨어공학과	김영중	20221053	1학년	1029294432
9	컴퓨터소프트웨어공학과	이하림	20222928	1학년	1055389364
10	컴퓨터소프트웨어공학과	배홍범	20180569	2학년	010-2107-5307
11	컴퓨터소프트웨어공학과	이재웅	20190818	2학년	1040059642
12	컴퓨터소프트웨어공학과	최창정	20181824	2학년	1023735526
13	기계공학과	김태현	20152321	2학년	1055861146
14	기계설계공학과	안태현	20223595	1학년	1075840306
15	로봇공학과	이정형	20222061	1학년	1067165858
16	로봇공학과	최한울	20220535	1학년	1032398278
17	로봇공학과	천성원	20191538	2학년	010 2595 3700
18	로봇공학과	천성원	20191538	2학년	010 2595 3700
19	자동화공학	장성민	20220498	1학년	1091655174
20	전기공학과	장지우	20210541	2학년	1077053785
21	정보전자공학과	박신영	20202176	1학년	1091924228
22	정보전자공학과	이건우	20200620	1학년	1045269066

파이썬을 위한 다양한 개발환경 소개

- 파이썬의 다양한 개발환경

- 표준 개발환경
 - IDLE 쉘
- 통합개발환경
 - Pycharm, Spyder 등
- 웹 개발환경, 주피터노트북 계열
 - ipython 기반, 파일 확장자 ipynb
 - 주피터 노트북
 - 코랩 노트북
 - 캐글 노트북
- 전문 편집기(에디터) 개발환경
 - 비주얼 스튜디오 코드
 - 서블라임 텍스트, 아톰 등

- 가상 환경

- 독립적인 파이썬 개발 환경
 - 하나의 컴퓨터에 여러 개 생성 가능

수업 개요

- 2022-7/12(화)에서 7/15까지 4일 동안 총 24시간
 - 매일(6시간), 오프라인 수업
- 참고자료
 - 교재
 - 여러분을 위한 참고자료
 - 파이썬 알고리즘 인터뷰: 95가지 알고리즘 문제 풀이로 완성하는 코딩 테스트
 - 코딩 참조:
 - 왕초보를 위한 파이썬
 - <https://wikidocs.net/book/2>
 - 데이터사이언스 스쿨
 - <https://datascienceschool.net/intro.html>
 - Realpython.com
 - 파이썬으로 시작하는 컴퓨터과학입문
 - 파이썬프로그래밍개론(An Introduction to Programming using Python)

수업 일정

- 1일 6시간

- 09:00 ~ 12:00, 13:00 ~ 16:00

	오전	오후
1일차	파이썬 표준 개발 환경, 모듈과 패키지	아나콘다 설치, 주피터 노트북 사용
2일차	모듈 기본, 파이참 설치	가상환경 개요와 생성
3일차	직접 가상환경 생성과 파이참에서 생성	주피터 노트북과 파이참 심화
4일차	구글 코랩	비주얼 스튜디오 코드

강의 홈페이지

- <https://github.com/ai7dnn/IDE-python>

파이썬 개발환경과 가상환경

인공지능소프트웨어학과
강 환수 교수

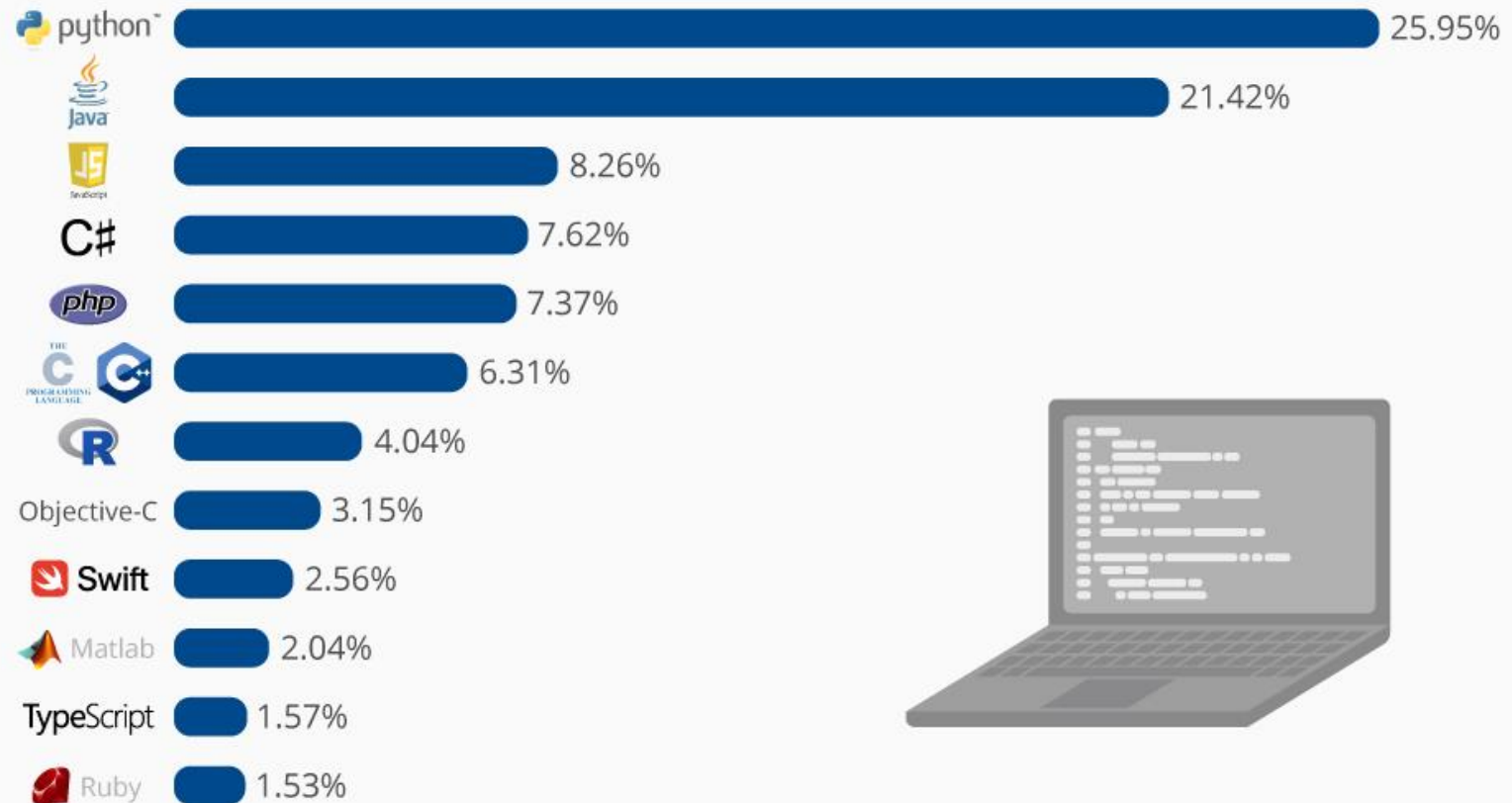
파이썬 표준 개발환경, 모듈과 패키지

파이썬 개발환경과 가상환경 수업 개요

파이썬 언어의 인기

The Most Popular Programming Languages

Share of the most popular programming languages in the world*



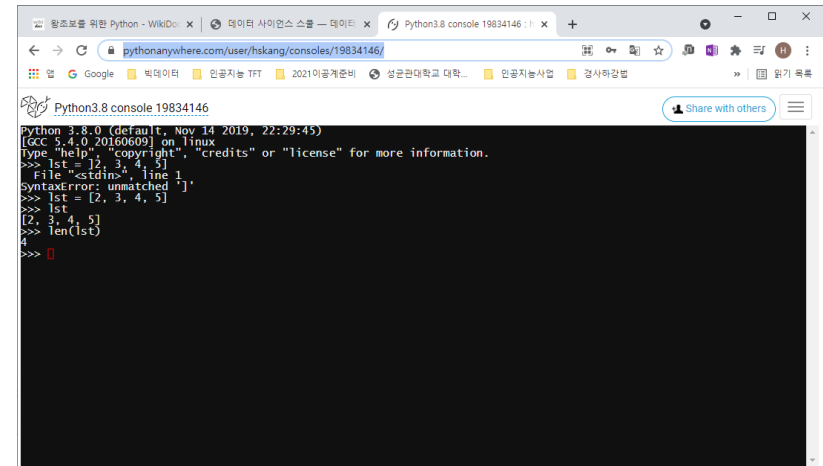
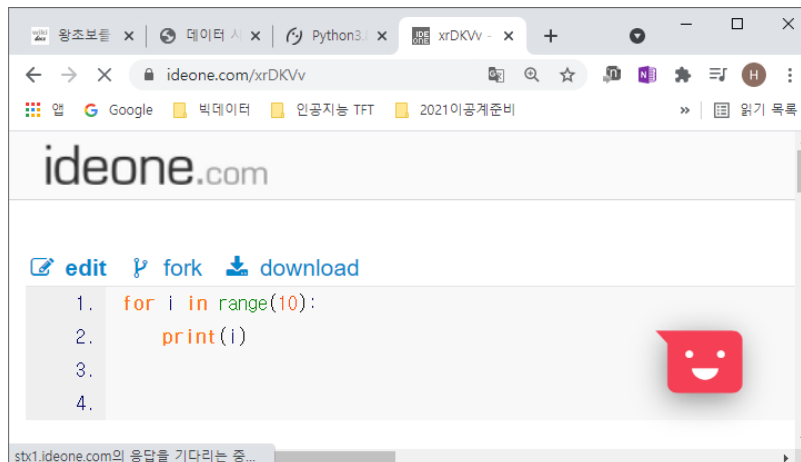
파이썬의 활용 영역

- 딥러닝/데이터과학 학습 순서



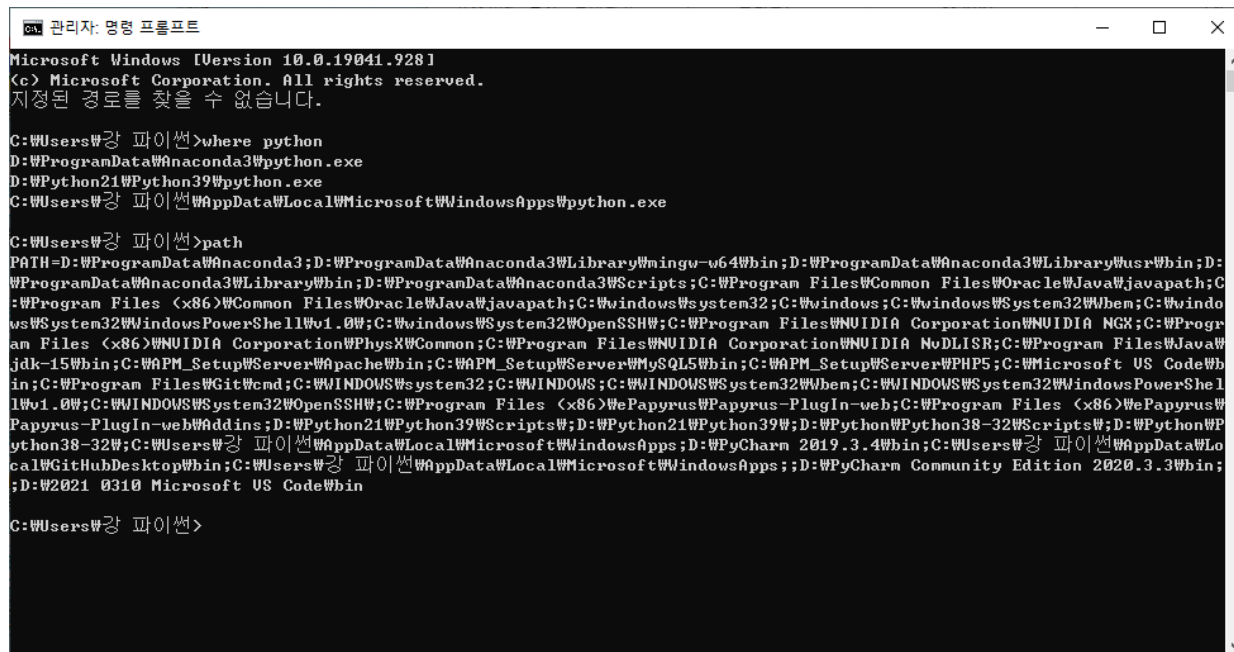
온라인 파이썬 개발 환경

- 표준 파이썬
 - <https://python.org>
- **programiz**
 - <https://www.programiz.com/python-programming/online-compiler/>
- **ideone**
 - <https://ideone.com>
- 구글 코랩
- 캐글



표준 파이썬 설치

- Python.org
- 설치 전/후 '명령 프롬프트'에서 확인
 - Path
 - Where python



```

관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.
지정된 경로를 찾을 수 없습니다.

C:\Users\강 파이썬>where python
D:\ProgramData\Anaconda3\python.exe
D:\Python21\Python39\python.exe
C:\Users\강 파이썬\AppData\Local\Microsoft\WindowsApps\python.exe

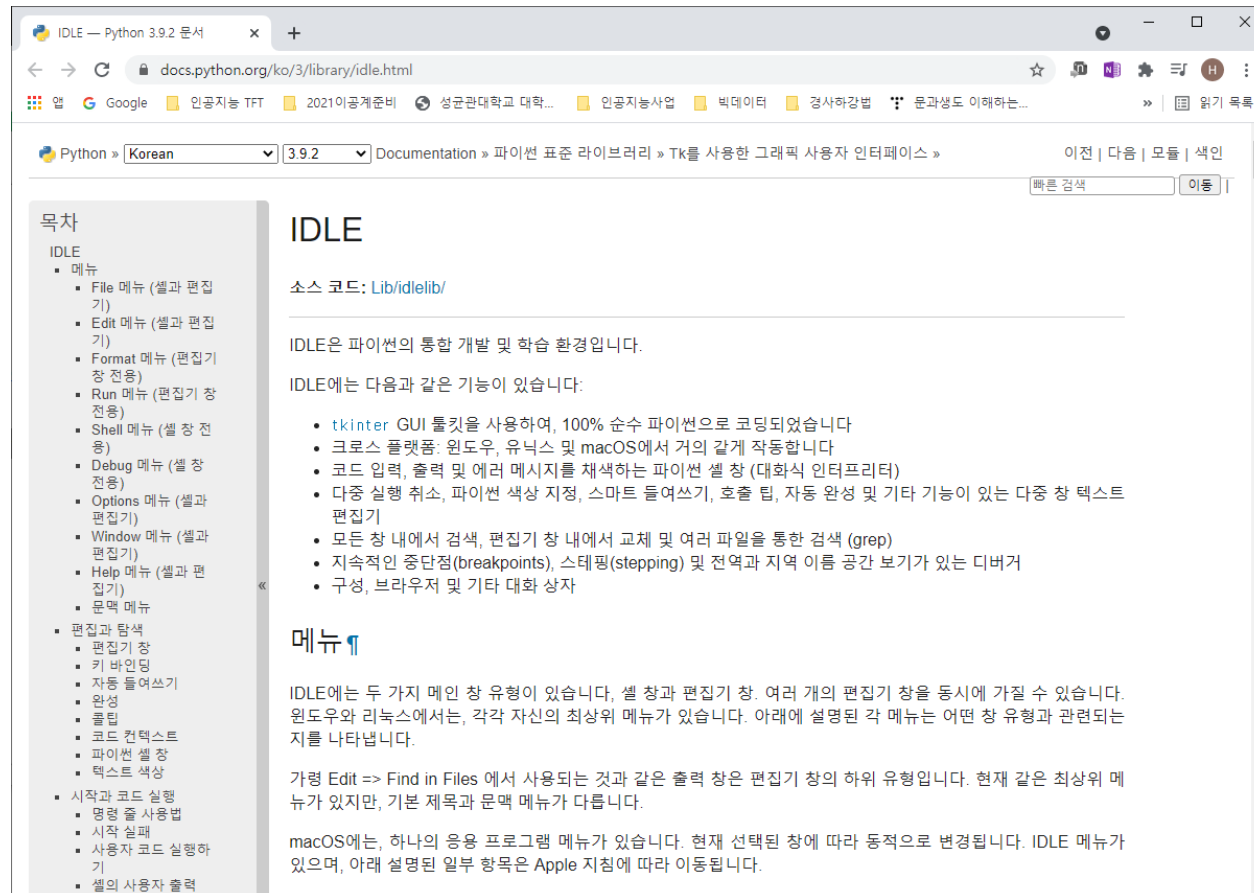
C:\Users\강 파이썬>path
PATH=D:\ProgramData\Anaconda3;D:\ProgramData\Anaconda3\Library\mingw-w64\bin;D:\ProgramData\Anaconda3\Library\usr\bin;D:\ProgramData\Anaconda3\Library\bin;D:\ProgramData\Anaconda3\Scripts;C:\Program Files\Common Files\Oracle\Java\javapath;C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files\NVIDIA Corporation\NVIDIA NGX;C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common;C:\Program Files\NVIDIA Corporation\NVIDIA NoDLIS;C:\Program Files\Java\jdk-15\bin;C:\WAPM_Setup\Server\Apache\bin;C:\WAPM_Setup\Server\MySQL5\bin;C:\WAPM_Setup\Server\PHP5;C:\Microsoft US Code\bin;C:\Program Files\Git\cmd;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;C:\Program Files (x86)\Papyrus\Papyrus-PlugIn-web;C:\Program Files (x86)\Papyrus\Papyrus-PlugIn-web\Addins;D:\Python21\Python39\Scripts\;D:\Python21\Python39\;D:\Python\Python38-32\Scripts\;D:\Python\Python38-32\;C:\Users\강 파이썬\AppData\Local\Microsoft\WindowsApps;D:\PyCharm 2019.3.4\bin;C:\Users\강 파이썬\AppData\Local\GitHubDesktop\bin;C:\Users\강 파이썬\AppData\Local\Microsoft\WindowsApps;D:\PyCharm Community Edition 2020.3.3\bin;D:\2021_0310_Microsoft_US_Code\bin

C:\Users\강 파이썬>
  
```

IDLE 둘러보기

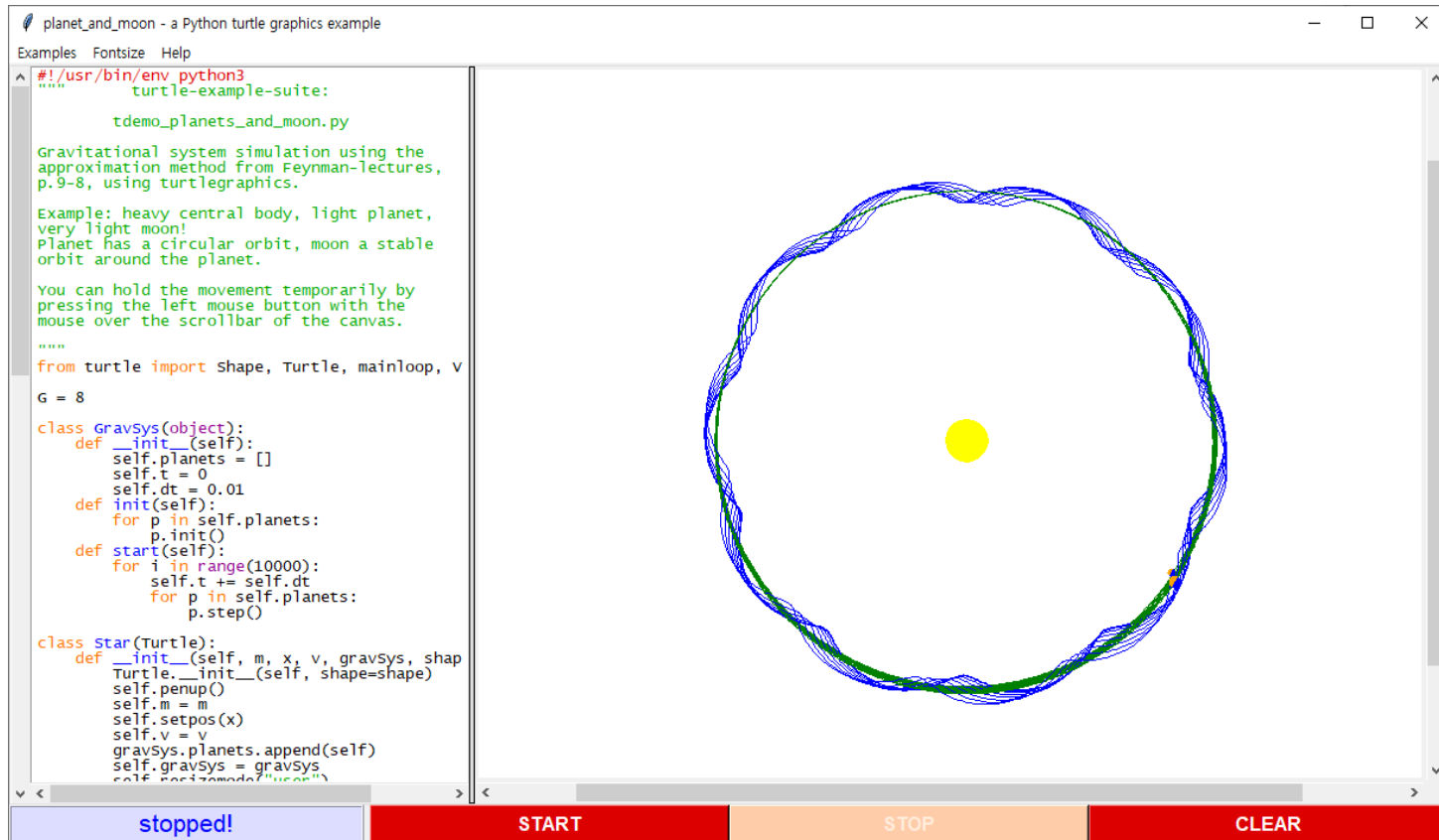
IDLE 문서

- 'IDLE 문서'로 검색
 - <https://docs.python.org/ko/3/library/idle.html>



그래픽 모듈 '거북이'

- Help | Turtle Demo



Quiz

- 우리의 강의자료가 저장된 개발자의 홈 페이지라 부르는 사이트는?
- 파이썬을 위한 온라인 개발환경은 무엇이 있는가?
- 표준 파이썬 개발환경 사이트는?
- 표준 파이썬 개발환경 IDE 이름은?
- 표준 파이썬 개발환경 IDE에 포함된 그래픽 도구의 이름은?

- 다음 중 파이썬에 대한 설명 중 잘못된 것은?
 - 표준 파이썬의 홈페이지는 www.python.org이다.
 - 표준 파이썬의 IDE는 IDLE이다.
 - 파이참은 전문 편집기 부류이다.
 - 표준 파이썬에는 turtle이라는 그래픽 모듈이 있다.

파이썬의 표준 개발환경

파이썬 설치 후 메뉴

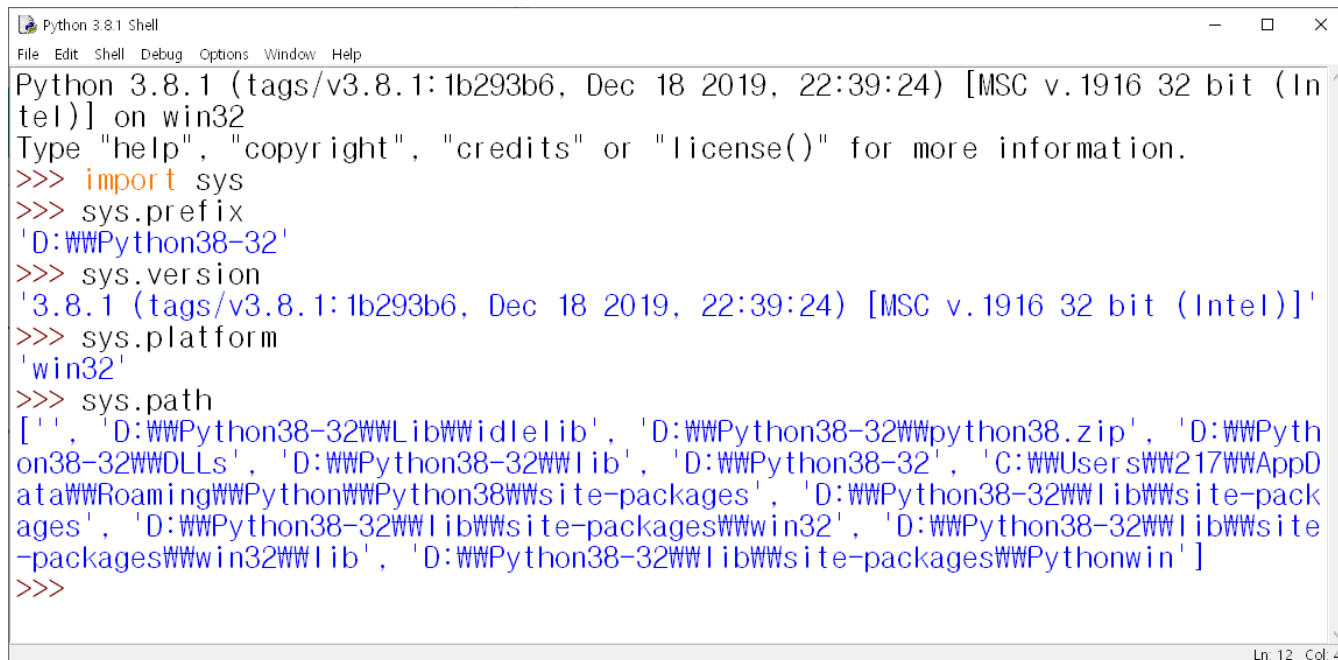
- **IDLE**
 - IDE 모습의 쉘
- **Python shell**
 - 도스 창 모습의 쉘
- **파이썬 매뉴얼**
 - 매뉴얼
- **Module docs**
 - 설치 모듈 문서
- **모듈**
 - 라이브러리 파일
 - 내부에는 함수, 클래스 등의 파이썬 소스



IDLE 셸

• 주요 시스템 환경 조사

- sys.prefix: 현 파이썬 설치 루트 폴더
- sys.version, sys.platform: 버전과 플랫폼
- sys.path: 현 파이썬 실행의 경로 목록
 - **sys의 path는 실행파일이나 패키지를 찾는 순서**
 - 이 경로에 없으면 오류 발생

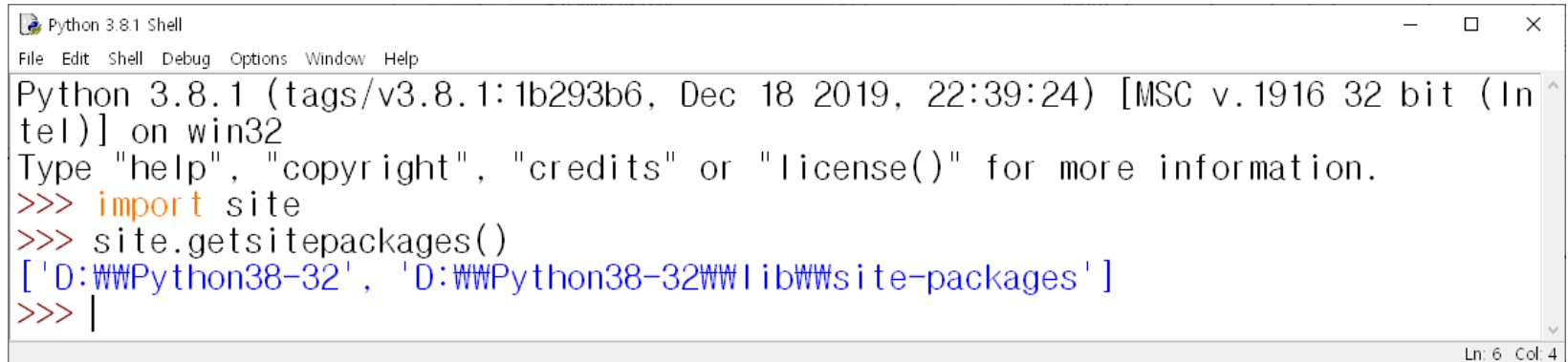


```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import sys
>>> sys.prefix
'D:\\Python38-32'
>>> sys.version
'3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)]'
>>> sys.platform
'win32'
>>> sys.path
['', 'D:\\Python38-32\\Lib\\idlelib', 'D:\\Python38-32\\python38.zip', 'D:\\Python38-32\\DLLs', 'D:\\Python38-32\\Lib', 'D:\\Python38-32', 'C:\\Users\\217\\AppData\\Roaming\\Python\\Python38\\site-packages', 'D:\\Python38-32\\Lib\\site-packages', 'D:\\Python38-32\\Lib\\site-packages\\win32', 'D:\\Python38-32\\Lib\\site-packages\\win32\\Lib', 'D:\\Python38-32\\Lib\\site-packages\\Pythonwin']
>>>
```

패키지 설치 폴더

- `site.getsitepackages()`

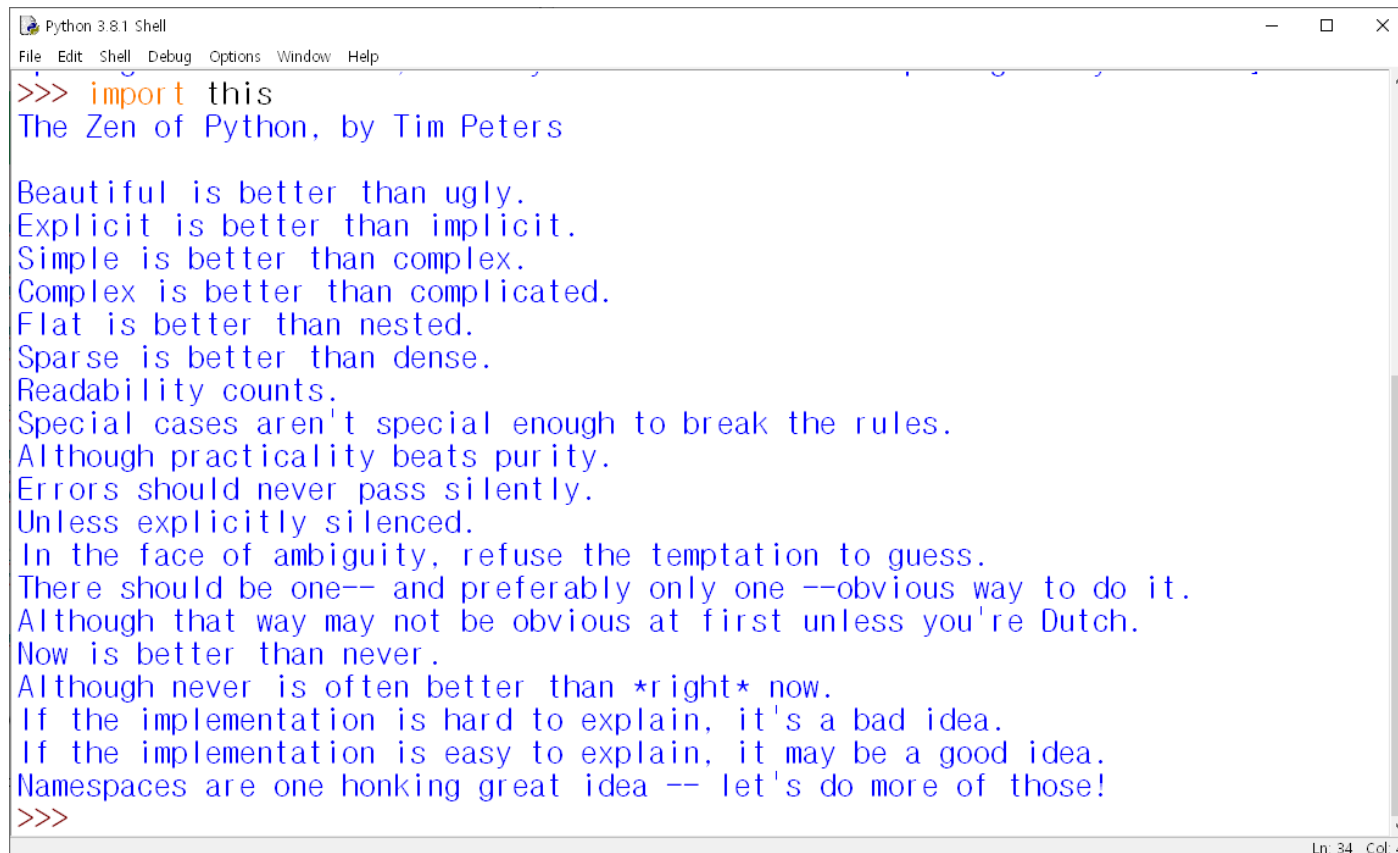
```
>>> import site
>>> site.getsitepackages()
['D:\WWPython38-32', 'D:\WWPython38-32\lib\site-packages']
>>>
```



```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import site
>>> site.getsitepackages()
['D:\WWPython38-32', 'D:\WWPython38-32\lib\site-packages']
>>> |
```

파이썬의 철학

- import this

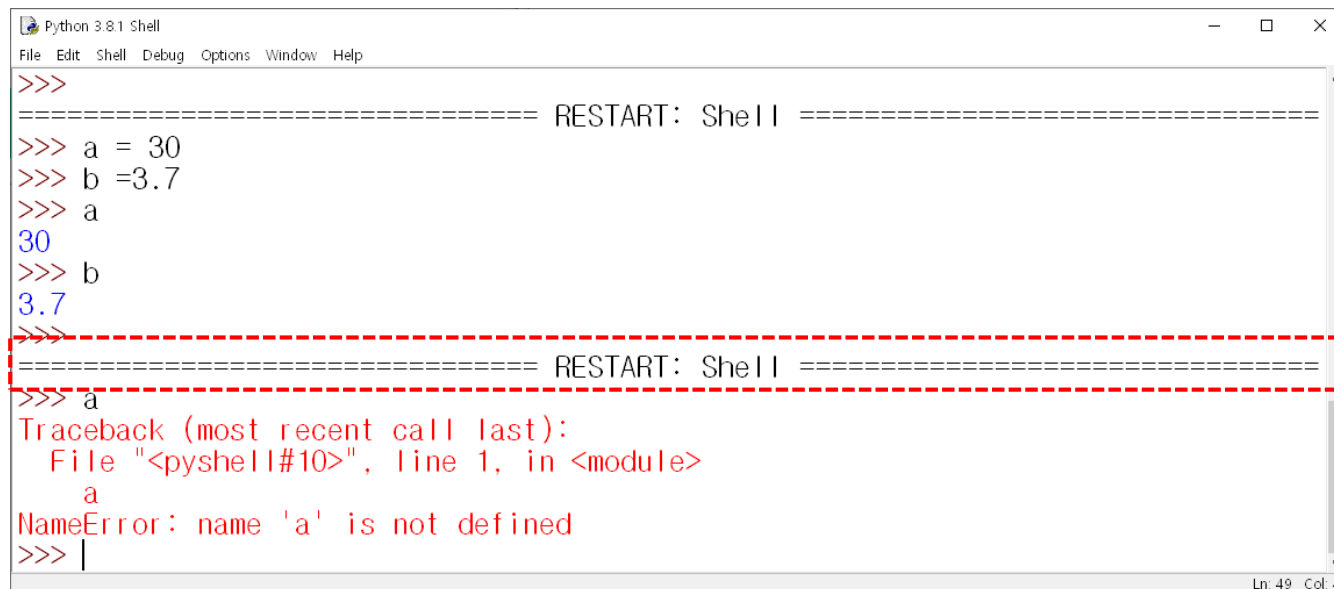


```
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```


파이썬 재실행

- 셸은 위에서 실행된 내용을 저장
- 재실행
 - 메뉴 Shell | Restart shell, ctrl + F6
 - 모든 정보가 사라지고 다시 시작



The screenshot shows a Python 3.8.1 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The command prompt shows the following sequence of commands and outputs:

```
>>>
===== RESTART: Shell =====
>>> a = 30
>>> b = 3.7
>>> a
30
>>> b
3.7
>>>
===== RESTART: Shell =====
>>> a
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    a
NameError: name 'a' is not defined
>>> |
```

A red dashed rectangle highlights the restart line and the subsequent error message. The status bar at the bottom right indicates "Ln: 49 Col: 4".

모듈 importlib

• import 구현 모듈

- 함수 reload(module)
 - **Reload the module and return it.**
- import importlib as imp
- imp.reload(this)

```
>>> import importlib as imp
...
>>> imp.reload(this)
...
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
```

• 모듈명.__all__

- 모든 메소드 항목 리스트

```
>>> imp.__all__
['__import__', 'import_module', 'invalidate_caches', 'reload']
```

주요 기능

- 코드 히스토리

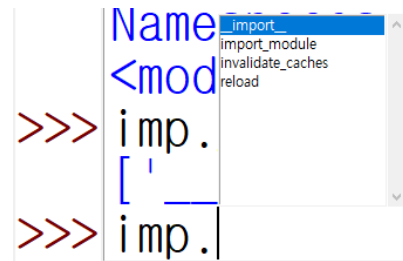
- 이전 코드
 - alt + P
- 이후 코드
 - alt + N

- 코드 이후 보이기

- 코딩 시 ctrl + space

- 도움말

- help(print)



```
>>> help(print)
Help on built-in function print in module builtins:

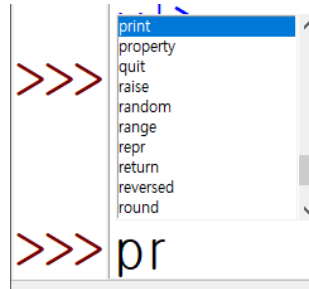
print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

코딩 자동 완성 기능

- Auto completion

- pr + [tab]
 - pr 관련 함수 보이기



- 함수의 괄호를 누르면 함수의 도움말이 표시(call tips)
 - 인자에 대한 도움말

```

Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1928 64 bit
D64]] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print(
    print(value, ..., sep=' ', end='\\n', file=sys.stdout, flush=False)
  
```

표준 라이브러리와
모듈, 패키지

표준 라이브러리

- 구글링에서 '파이썬 표준 라이브러리' 로 검색

- <https://docs.python.org/ko/3/library/>

- 내장 함수

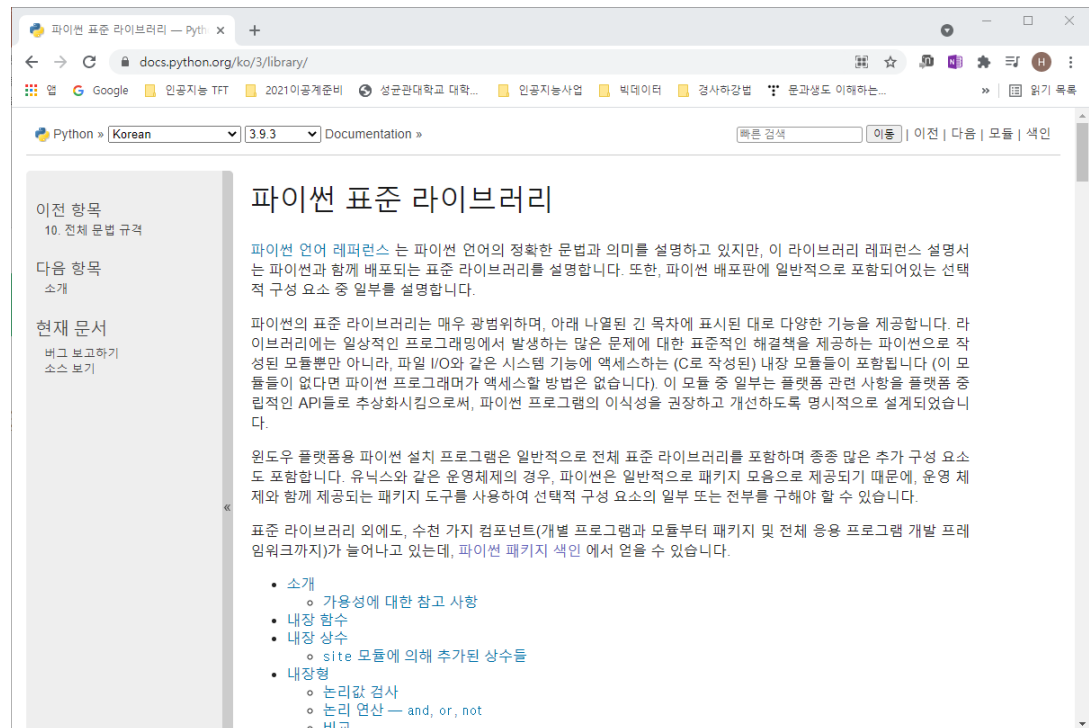
- 그대로 사용

- `print()` 등

- 표준 모듈

- 표준 설치 시 설치되는 확장 라이브러리

- `import` 문 필요



내장 함수

Built-in function

Python 3.9.4 문서

docs.python.org/ko/3/library/functions.html

Python » Korean » 3.9.4 » Documentation » 파이썬 표준 라이브러리 »

이전 항목
소개

다음 항목
내장 상수

현재 문서
버그 보고하기
소스 보기

내장 함수

파이썬 인터프리터에는 항상 사용할 수 있는 많은 함수와 형이 내장되어 있습니다. 여기에서 알파벳 순으로 나열합니다.

내장 함수			
abs()	delattr()	hash()	memoryview()
all()	dict()	help()	min()
any()	dir()	hex()	next()
ascii()	divmod()	id()	object()
bin()	enumerate()	input()	oct()
bool()	eval()	int()	open()
breakpoint()	exec()	isinstance()	ord()
bytearray()	filter()	issubclass()	pow()
bytes()	float()	iter()	print()
callable()	format()	len()	property()
chr()	frozenset()	list()	range()
classmethod()	getattr()	locals()	repr()
compile()	globals()	map()	reversed()
complex()	hasattr()	max()	round()
			set()
			setattr()
			slice()
			sorted()
			staticmethod()
			str()
			sum()
			super()
			tuple()
			type()
			vars()
			zip()
			__import__()

abs(x)
숫자의 절댓값을 돌려줍니다. 인자는 정수, 실수 또는 `__abs__()`를 구현하는 객체입니다. 인자가 복소수면 그 크기가 반환됩니다.

all(iterable)
`iterable`의 모든 요소가 참이면 (또는 `iterable`이 비어있으면) `True`를 돌려줍니다. 다음과 동등합니다:

```
def all(iterable):
    for element in iterable:
        if not element:
            return False
    return True
```

any(iterable)
`iterable`의 요소 중 어느 하나라도 참이면 `True`를 돌려줍니다. `iterable`이 비어 있으면 `False`를 돌려줍니다. 다음과 동등합니다.

Python » Korean » 3.10.5 » 3.10.5 Documentation » 파이썬 표준 라이브러리 » 내장 함수

이전 항목
소개

다음 항목
내장 상수

이 페이지
버그 보고하기
소스 보기

내장 함수

파이썬 인터프리터에는 항상 사용할 수 있는 많은 함수와 형이 내장되어 있습니다. 여기에서 알파벳 순으로 나열합니다.

내장 함수

A	E	L	R
abs()	enumerate()	len()	range()
alter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
any()			round()
anext()	F	M	S
ascii()	filter()	map()	set()
	float()	max()	setattr()
	format()	memoryview()	slice()
	frozenset()	min()	sorted()
B			staticmethod()
bin()			str()
bool()			sum()
breakpoint()	G	N	super()
bytearray()	getattr()	next()	
bytes()	globals()	ord()	
C	H	O	T
callable()	hasattr()	object()	tuple()
chr()	hash()	oct()	type()
classmethod()	help()	open()	
compile()	hex()	ord()	
complex()			
	I	P	V
	id()	pow()	vars()
	input()	print()	
	int()	property()	
	isinstance()		
	issubclass()		
	iter()		
D			Z
delattr()			zip()
dict()			
dir()			
divmod()			
			_
			__import__()

abs(x)
숫자의 절댓값을 돌려줍니다. 인자는 정수, 실수 또는 `__abs__()`를 구현하는 객체입니다. 인자가 복소수면 그 크기가 반환됩니다.

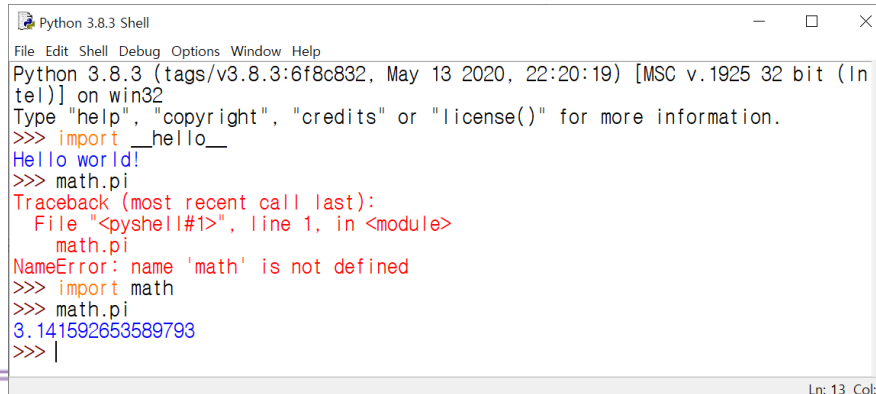
alter(async_iterable)
Return an **asynchronous iterator** for an **asynchronous iterable**. Equivalent to calling `x.__aiter__()`.

Note: Unlike `iter()`, `alter()` has no 2-argument variant.

버전 3.10에 추가.

모듈 module

- **print() 와 같은 함수**
 - 내장 함수(Built in function)
- **모듈**
 - 데이터, 함수, 클래스 등이 담겨져 있는 파일
- **표준 모듈 또는 외부의 라이브러리(모듈)을 사용하기 위해서는**
 - import 하는 작업이 필요
 - **import 모듈명**
 - **import 모듈명1, 모듈명2, ...**
 - 만약 내가 원하는 이름: import ~ as ~ 를 사용
 - **import 모듈명 as 원하는 이름**
 - **import 모듈명1 as 원하는 이름, 모듈명2 as 원하는 이름, ...**



```
Python 3.8.3 Shell
File Edit Shell Debug Options Window Help
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import __hello__
Hello world!
>>> math.pi
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    math.pi
NameError: name 'math' is not defined
>>> import math
>>> math.pi
3.141592653589793
>>> |
```


Quiz

- 내장함수와 모듈의 사용 방법의 차이는?
- 대표적인 내장함수의 종류는?
- 대표적인 모듈의 종류는?
- 대표적인 표준 모듈의 종류는?
- 모듈 random을 이름 rd로 사용하려는 문장은?
- 모듈의 random에서 사용하는 대표적인 함수는?

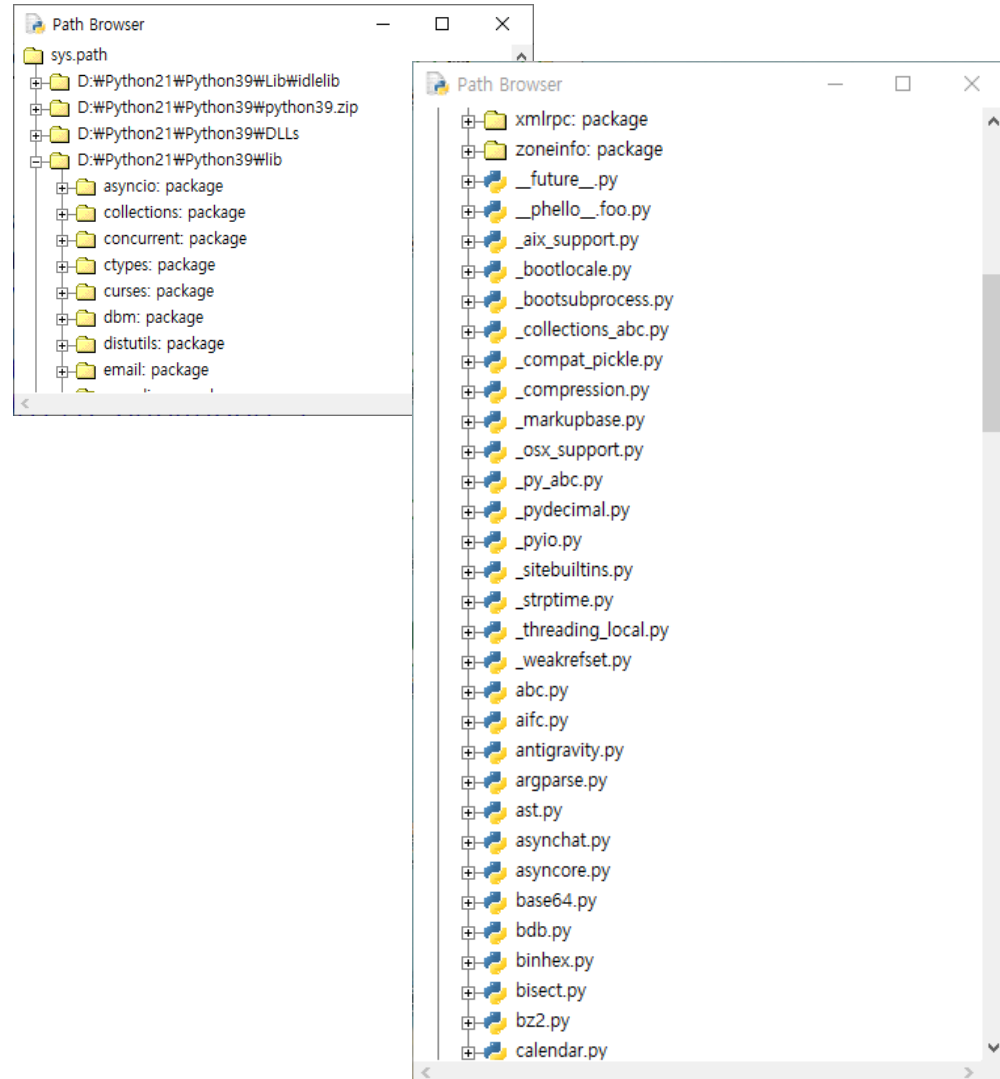
- 다음 파이썬의 설명 중 잘못된 것은?
 - 내장 함수는 바로 함수를 사용할 수 있다.
 - 모든 모듈과 패키지는 pip 등으로 설치해야 한다.
 - print()가 대표적인 내장 함수이다.
 - 모듈 random을 사용하려면 import 해야 한다.

IDLE에서 모듈, 패키지 보기

IDLE 메뉴 1

- **File | Path Browser**

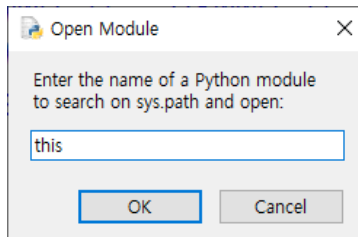
- sys.path 확인



- 하부 파일도 조회

IDLE 메뉴 2

- File | Module Browser



```

this.py - D:\Python21\Python39\lib\this.py (3.9.2)
File Edit Format Run Options Window Help
s = """Gur Mra bs Clguba, ol Gvz Crgref

Ornhgvshy vf orggre guna htyl.
Rkcyvpg vf orggre guna vzcypvg.
Fvzcyr vf orggre guna pbzcyrk.
Pbzcyrk vf orggre guna pbzcyvpngrq.
Syng vf orggre guna arfgrq.
Fonefr vf orggre guna qrafr.
Ernqnovyvg! pbhagf.
Forpvnv pnfrf nera'g forpvnv rabhtu gb oernx gur ehryf.
Nygubhtu cenpgvpnyvgl orngf chevgl.
Reebef fubhyq arire onff fvyragyl.
Hayrff rkcyvpgyl fvyraprq.
Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff.
Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg.
Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu.
Abj vf orggre guna arire.
Nygubhtu arire vf bsgra orggre guna *evtug* abj.
Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn.
Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn.
Anzrfonprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!"""

d = {}
for c in (65, 97):
    for i in range(26):
        d[chr(i+c)] = chr((i+13) % 26 + c)
print("".join([d.get(c, c) for c in s]))

```

IDLE에서 모듈 확인

- 모듈 random: random.py**
 - Menu alt+c

```

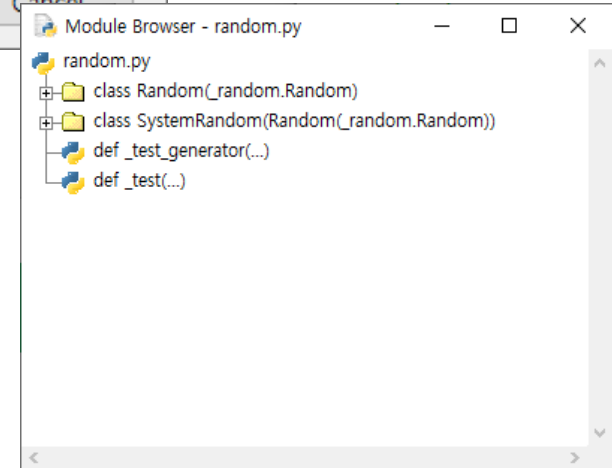
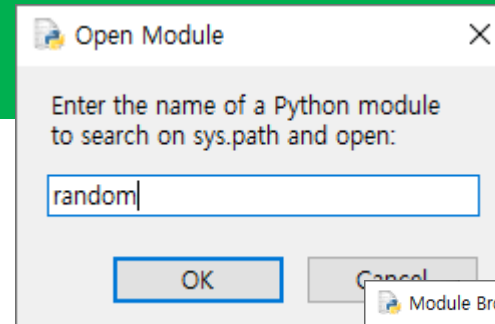
print('avg %g, stddev %g, min %g, max %g\n' % (xbar, sigma, low
def _test(N=2000):
    _test_generator(N, random, ())
    _test_generator(N, normalvariate, (0.0, 1.0))
    _test_generator(N, lognormvariate, (0.0, 1.0))
    _test_generator(N, vonmisesvariate, (0.0, 1.0))
    _test_generator(N, gammavariate, (0.01, 1.0))
    _test_generator(N, gammavariate, (0.1, 1.0))
    _test_generator(N, gammavariate, (0.1, 2.0))
    _test_generator(N, gammavariate, (0.5, 1.0))
    _test_generator(N, gammavariate, (0.9, 1.0))
    _test_generator(N, gammavariate, (1.0, 1.0))
    _test_generator(N, gammavariate, (2.0, 1.0))
    _test_generator(N, gammavariate, (20.0, 1.0))
    _test_generator(N, gammavariate, (200.0, 1.0))
    _test_generator(N, gauss, (0.0, 1.0))
    _test_generator(N, betavariate, (3.0, 3.0))
    _test_generator(N, triangular, (0.0, 1.0, 1.0 / 3.0))

## -----
## ----- fork support -----

if hasattr(_os, "fork"):
    _os.register_at_fork(after_in_child=_inst.seed)

if __name__ == '__main__':
    _test()

```



모듈 직접 실행

- cmd 창 열어
 - 명령 python 파일.py
 - > python random.py
- REPL 셸로 모듈 import

관리자: C:\Windows\System32\cmd.exe - python

```
D:\Python21\Python39\Lib>python
Python 3.9.2 (tags/v3.9.2:1a79785, Feb 19 2021, 13:44:55) [MSC v.1916 64-bit (AMD64)]
Type "help", "copyright", "credits" or "license()" for more
>>> import random
>>> random.randint(0, 3)
2
>>> random.randint(0, 3)
0
>>> random.randint(0, 3)
2
>>> random.randint(0, 3)
2
>>> random.randint(0, 3)
2
>>> random.randint(0, 3)
1
>>> random.randint(0, 3)
0
>>> random.randint(0, 3)
0
>>> random.randint(0, 3)
3
>>>
```

```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.
지정된 경로를 찾을 수 없습니다.

D:\Python21\Python39\Lib>python random.py
0.000 sec, 2000 times random
avg 0.511218, stddev 0.285653, min 0.000499013, max 0.999594

0.001 sec, 2000 times normalvariate
avg 0.00857388, stddev 0.990751, min -3.37108, max 3.0941

0.001 sec, 2000 times lognormvariate
avg 1.6651, stddev 2.29108, min 0.0336902, max 39.8299

0.001 sec, 2000 times vonmisesvariate
avg 3.12439, stddev 2.29158, min 0.00117169, max 6.28036

0.001 sec, 2000 times gammavariate
avg 0.0134559, stddev 0.148196, min 0, max 4.33881

0.001 sec, 2000 times gammavariate
avg 0.09715, stddev 0.289514, min 3.08163e-29, max 3.16598

0.001 sec, 2000 times gammavariate
avg 0.180767, stddev 0.574914, min 3.2058e-51, max 11.4787

0.001 sec, 2000 times gammavariate
avg 0.492577, stddev 0.738238, min 1.03248e-06, max 6.78817

0.001 sec, 2000 times gammavariate
avg 0.883912, stddev 0.933105, min 0.000100033, max 6.91854

0.001 sec, 2000 times gammavariate
avg 0.964436, stddev 0.987662, min 0.000228203, max 10.984

0.002 sec, 2000 times gammavariate
avg 2.02169, stddev 1.40661, min 0.0430796, max 9.44286

0.003 sec, 2000 times gammavariate
avg 19.9942, stddev 4.54677, min 9.87121, max 38.0844

0.002 sec, 2000 times gammavariate
avg 199.567, stddev 14.4779, min 156.133, max 262.905

0.001 sec, 2000 times gauss
avg -0.0288543, stddev 1.05695, min -3.77795, max 3.39664

0.004 sec, 2000 times betavariate
avg 0.502721, stddev 0.193058, min 0.0409878, max 0.987212

0.001 sec, 2000 times triangular
avg 0.441244, stddev 0.20941, min 0.00692722, max 0.990435

D:\Python21\Python39\Lib>
```

모듈 소스 코드 확인

- 모듈 inspect
- inspect.getsource()
 - 모듈의 소스 코드를 반환
 - random 라이브러리의 소스 코드를 출력하는 예시 코드
 - import inspect
 - import random
 - print(inspect.getsource(random))
- inspect.getfile()
 - 라이브러리가 구현된 py 파일의 경로를 직접 찾아서 코드를 확인

```
>>> import inspect
>>> import random
>>> print(inspect.getsource(random))
Squeezed text (941 lines).
>>> print(inspect.getfile(random))
D:\Python3102W\lib\random.py
```

- 모든 모듈이 파이썬 소스가 있는 것이 아님
 - 모듈 math는 없음

패키지 package

- 패키지는 연관된 모듈들의 집합
 - urllib 패키지
 - URL과 관련된 request, response, parse 등 여러 가지의 모듈들이 포함
- 일반적으로 사용 시
 - 모듈과 패키지를 구분하지 않은 경우 많음
 - 파이썬 경우
 - 라이브러리 >= 패키지 >= 모듈

패키지 확인

• 패키지 urllib

The first screenshot shows a Python 3.8.3 Shell window with the following code and output:

```
>>> import urllib
>>> dir(urllib)
['_builtins_', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__package__', '__path__', '__spec__', 'parse']
>>> dir(urllib.request)
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    dir(urllib.request)
AttributeError: module 'urllib' has no attribute 'request'
>>> dir(urllib.parse)
['_DefragResult', '_DefragResultBytes', 'MAX_CACHE_SIZE', 'ParseResult', 'ParseResultBytes', 'Quoter', 'ResultBase', 'SplitResult', 'SplitResultBytes', '_ALWAYS_SAFE_BYTES', '_DefragResultBase', '_NetlocResultMixinBase', '_NetlocResultMixinBytes', '_NetlocResultMixinStr', '_ParseResultBase', '_ResultMixinBase', '_ResultMixinStr', '_SplitResultBase', '_all_', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '__uri__', '_checknetloc', '_coerce_args', '_decode_args', '_encode_result', '_hexdig', '_hextobyte', '_hostprog', '_implicit_encoding', '_implicit_errors', '_noop', '_parse_cache', '_portprog', '_safe_quoters', '_splitattr', '_splitnetloc', '_splitport', '_splitparams', '_splitpasswd', '_splitport', '_splitquery', '_splittag', '_splittype', '_splituser', '_splitvalue', '_typeprog', '_clear_cache', '_collections', '_namedtuple', '_non_hierarchical', '_parse_qs', '_parse_qsl', '_quote', '_quote_from_bytes', '_quote_plus', '_requote_chars', '_splitattr', '_splithost', '_splitport', '_splitpasswd', '_splitport', '_splitquery', '_splittag', '_splittype', '_splituser', '_splitvalue', '_sys', '_unescape', '_unquote', '_unquote_plus', '_unquote_to_bytes', '_unwrap', '_uridefrag', '_encode', '_urljoin', '_urlparse', '_urlsplit', '_urlunparse', '_urlunsplit', '_use_fragment', '_uses_netloc', '_uses_params', '_uses_query', '_uses_relative', '_warn']
>>> |
```

The second screenshot shows a Module Browser window for the 'request.py' module. It displays the module's structure and documentation. The documentation includes the following text:

urlopen(url, data=None) — Basic usage is the same as original urllib. pass the url and optionally data to post to an HTTP URL, and get a file-like object back. One difference is that you can also pass a Request instance instead of URL. Raises a URLError (subclass of OSError); for HTTP errors, raises an HTTPError, which can also be treated as a valid response.

build_opener — Function that creates a new OpenerDirector instance. Will install the default handlers. Accepts one or more Handlers as arguments, either instances or Handler classes that it will instantiate. If one of the argument is a subclass of the default handler, the argument will be installed instead of the default.

install_opener — Installs a new opener as the default opener.

objects of interest:

OpenerDirector — Sets up the User Agent as the Python-urllib client and manages the Handler classes, while dealing with requests and responses.

Request — An object that encapsulates the state of a request. The state can be as simple as the URL. It can also include extra HTTP headers, e.g. a User-Agent.

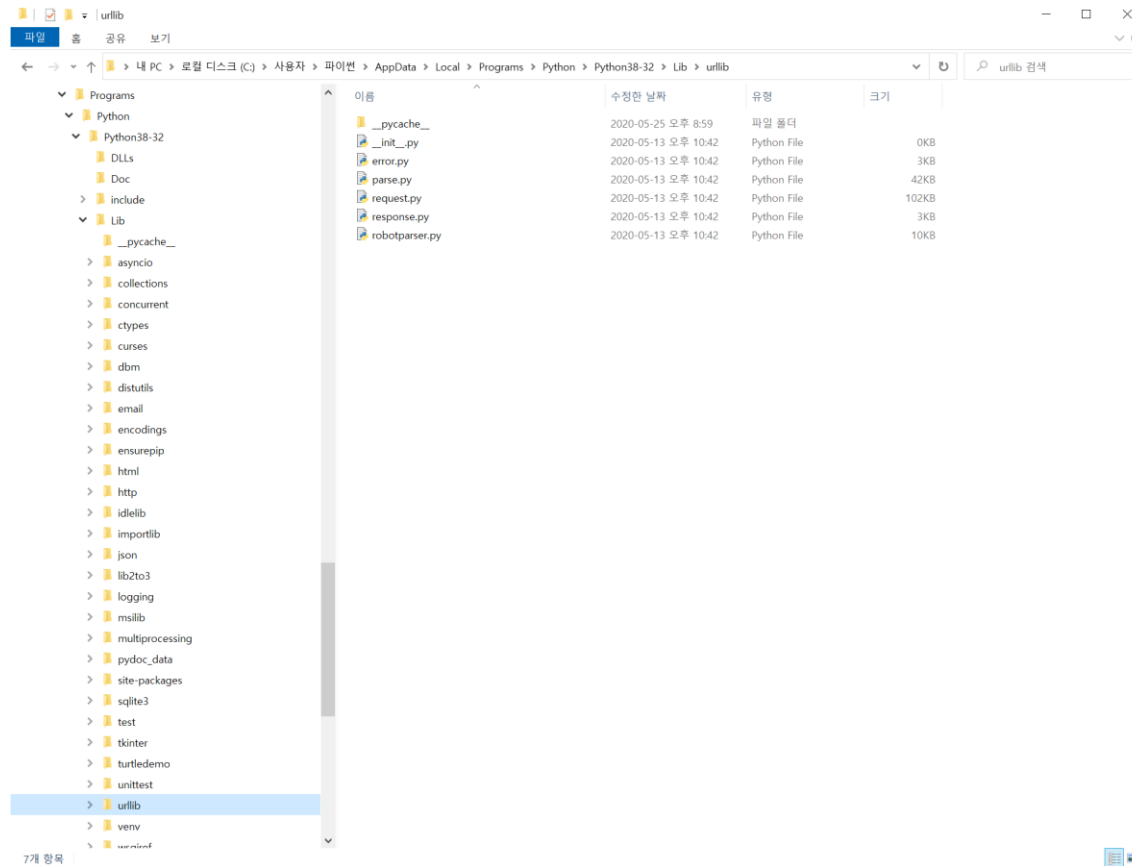
• 메뉴 File | Module Browser

– alt + c

• urllib.request

패키지 폴더 확인

- C:\Users\파이썬\AppData\Local\Programs\Python\Python38-32\Lib\urllib



path

- cmd 창에서 확인

- 표준 파이썬의 path
 - D:\Python21\Python39\Scripts;
 - D:\Python21\Python39;
- 아나콘다를 위한 path
 - D:\ProgramData\20210223\Anaconda3;
 - D:\ProgramData\20210223\Anaconda3\Library\mingw-w64\bin;
 - D:\ProgramData\20210223\Anaconda3\Library\usr\bin;
 - D:\ProgramData\20210223\Anaconda3\Library\bin;
 - D:\ProgramData\20210223\Anaconda3\Scripts;

폴더 확인

- **표준 라이브러리 폴더**
 - 설치 폴더 하부 lib
 - `D:\Python21\Python39\Lib`
- **패키지**
 - 하부 폴더
- **모듈**
 - *.py 파일

표준 라이브러리

- 내장 함수
- 표준 모듈
 - 내장 모듈
 - `math`
 - 소스 없이(C로 작성) 기본으로 제공
 - 표준 (파이썬) 모듈
 - `random, os`
 - 소스로 제공
 - `Python39\Lib` 폴더

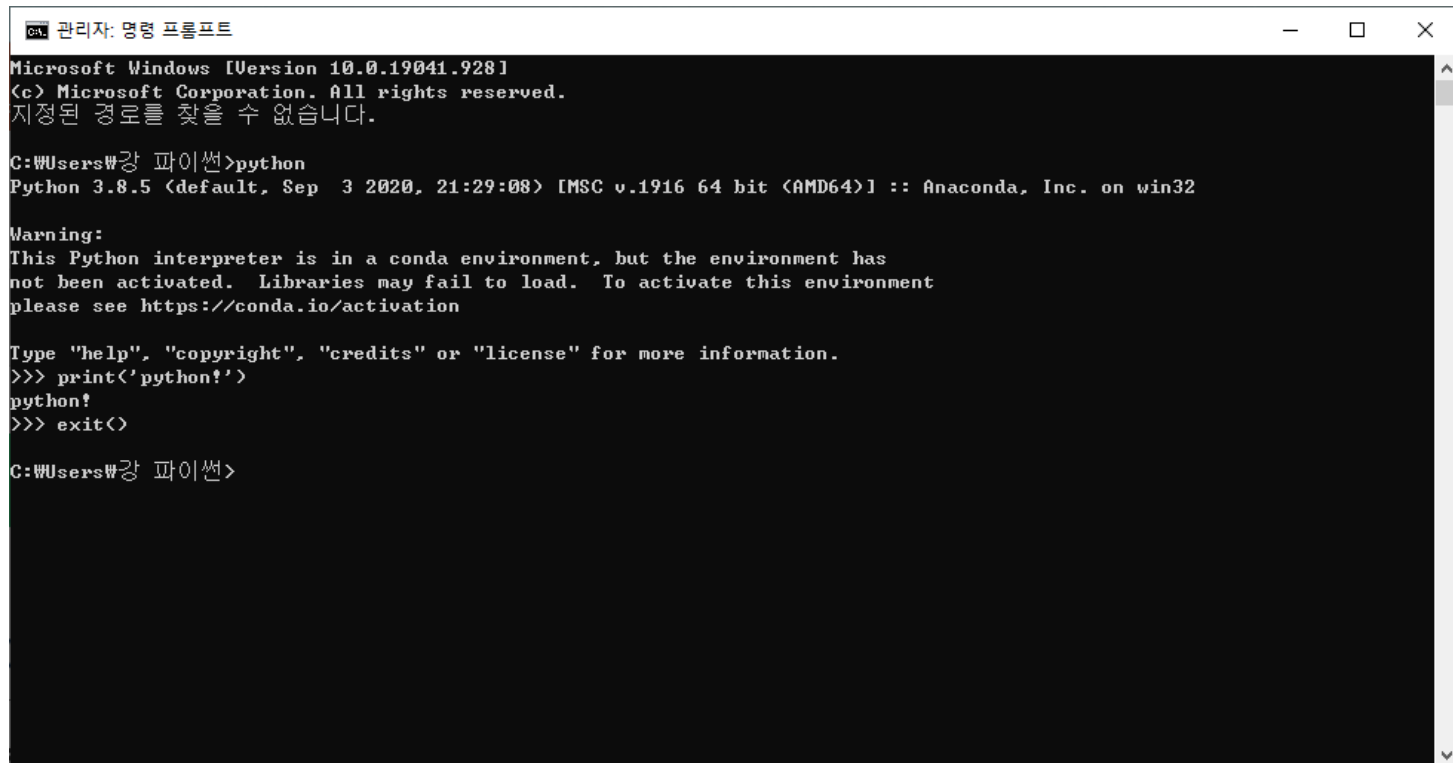
모듈 os

- 함수
 - `os.getcwd()`
 - `os.getpid()`
 - `os.listdir('.')`
- 전체 목록 확인
 - `dir(os)`

파이썬 인터프리터를 웹로 활용

파이썬 셸 실행과 종료

- 파이썬 셸 열기
 - 명령어 `python`
- 셸 나가기
 - `ctrl+D`, `exit()`



```
관리자: 명령 프롬프트
Microsoft Windows [Version 10.0.19041.928]
(c) Microsoft Corporation. All rights reserved.
지정된 경로를 찾을 수 없습니다.

C:\Users\강 파이썬>python
Python 3.8.5 (default, Sep 3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] :: Anaconda, Inc. on win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated. Libraries may fail to load. To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> print('python!')
python!
>>> exit()

C:\Users\강 파이썬>
```

파일에 저장, hello.py

- 직접 파이썬 소스인 파일 실행
 - python hello.py

The image shows a Windows PowerShell window and a Notepad window. The PowerShell window has a dark blue background and shows the following commands and output:

```
>>> exit()
D:\>notepad hello.py
D:\>python hello.py
0
1
2
3
4
5
6
7
8
9
D:\>
```

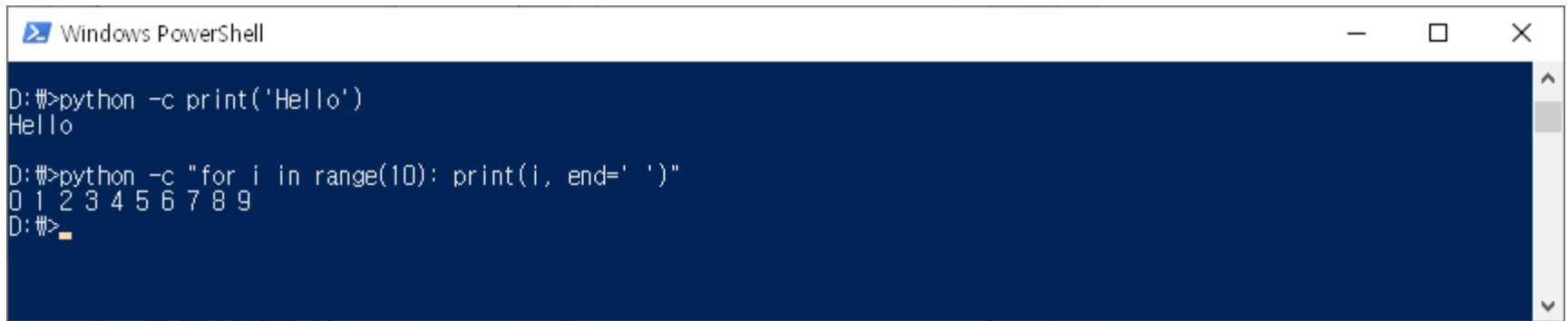
The Notepad window, titled "hello.py - Windows 메모장", shows the following code:

```
for i in range(10):
    print(i)
```

The status bar at the bottom of the Notepad window indicates "Ln 2, Col 13", "100%", "Windows (CRLF)", and "UTF-8".

한 줄 실행

- 옵션 -c



```
Windows PowerShell
D:\>python -c print('Hello')
Hello
D:\>python -c "for i in range(10): print(i, end=' ')"
0 1 2 3 4 5 6 7 8 9
D:\>
```

- `python -c "i = 10; print('짝수') if i%2 == 0 else print('홀수')"`

표준 파이썬 매뉴얼

파이썬 매뉴얼

Python 3.8.2 documentation

handles multiple arguments, floating point quantities, and strings. Strings are printed without quotes, and a space is inserted between items, so you can format things nicely, like this:

```
>>> i = 256*256
>>> print('The value of i is', i)
The value of i is 65536
```

The keyword argument `end` can be used to avoid the newline after the output, or end the output with a different string:

```
>>> a, b = 0, 1
>>> while a < 1000:
...     print(a, end=',')
...     a, b = b, a+b
...
0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,
```

Footnotes

- [1] Since `**` has higher precedence than `-`, `-3**2` will be interpreted as `-(3**2)` and thus result in `-9`. To avoid this and get `9`, you can use `(-3)**2`.
- [2] Unlike other languages, special characters such as `\n` have the same meaning with both single (`'...'`) and double (`"..."`) quotes. The only difference between the two is that within single quotes you don't need to escape `"` (but you have to escape `'`) and vice versa.

Python » 3.8.2 Documentation » The Python Tutorial »

previous | next | modules | index

© Copyright 2001-2020, Python Software Foundation.
The Python Software Foundation is a non-profit corporation. [Please donate.](#)

Last updated on Feb 25, 2020. [Found a bug?](#)
Created using [Sphinx](#) 2.4.3.

표준 패키지 활용

Palnet_and_moon.py 실행 해보기

- **from import 구문**

- from 모듈(패키지명)명 import 함수명 as 다른사용이름

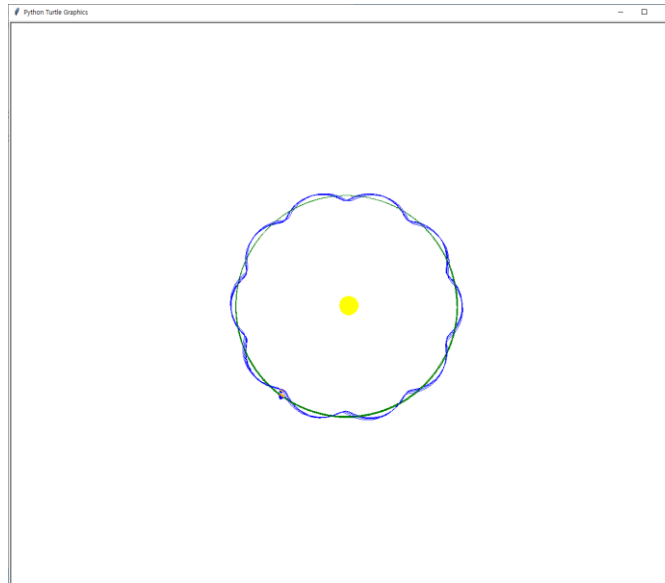
- **패키지 `turtledemo`로 여러 모듈을 제공**

- `>>> from turtledemo import planet_and_moon as pm`

- `>>> dir(pm)`

- `['G', 'GravSys', 'Shape', 'Star', 'Turtle', 'Vec', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'main', 'mainloop']`

- `>>> pm.main()`



Quiz

- 문장 import를 예를 들어 설명하시오.
- 문장 from import를 예를 들어 설명하시오.
- 파이썬에서 모듈의 집합을 의미하는 용어는?
- IDLE 쉘에서 모듈 소스를 바로 볼 수 있는 메뉴는?
- 0부터 9까지 한 줄에 출력하는 파이썬 코드는?

- 다음 표준 파이썬의 설명 중 잘못된 것은?
 - 모듈 random에는 randint()라는 함수가 있다.
 - 파이썬 모듈의 집합을 패키지라 한다.
 - 표준 모듈이 설치된 폴더는 \Python39\Lib 폴더이다.
 - 명령어 python으로 파이썬 셸을 실행시킬 수 있다.