

Ch 11: File IOs and Exception in Python

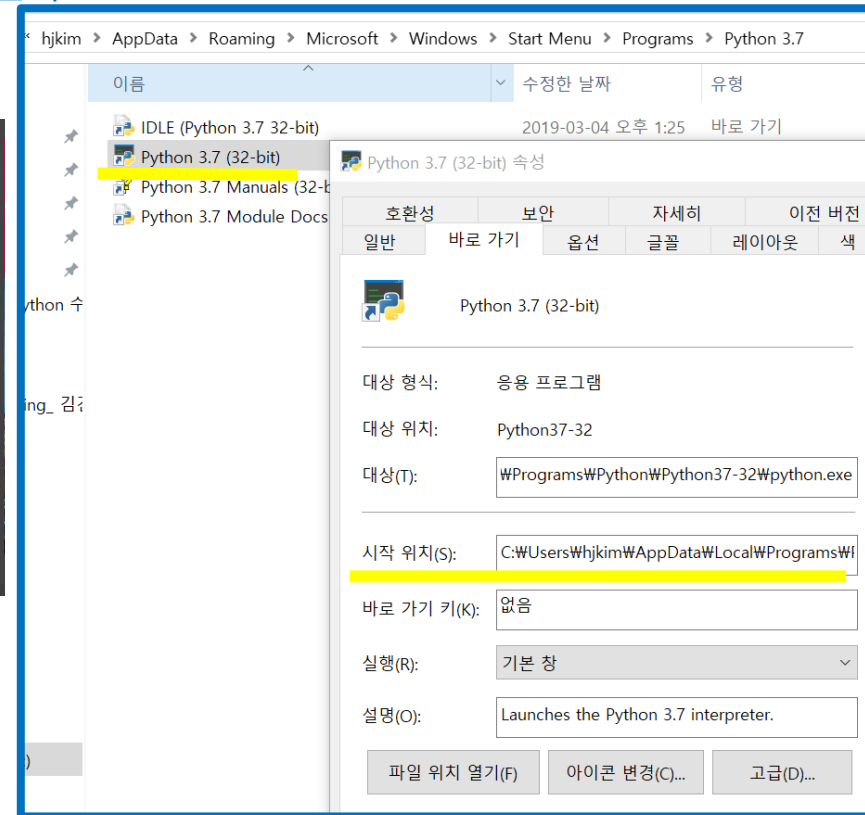
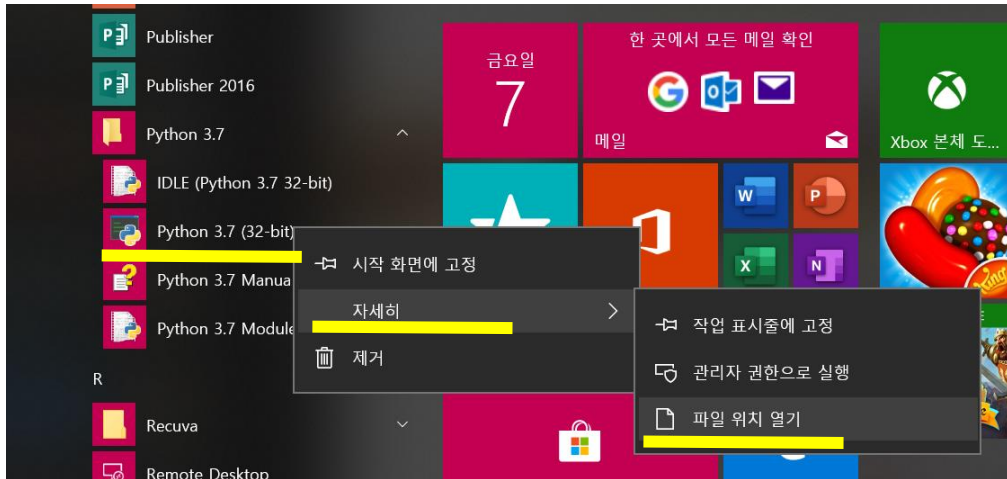
- File IO
 - Text File Manipulation
 - Big File with Numbers
 - CSV File
- Exception Handling

File의 생성위치

- Default: Python shell이 설치된 directory에 foo.txt가 만들어진다

```
>>> f = open("foo.txt", "w")
```

```
>>> f.close()
```



- 바탕화면에 파일을 생성하려면 바탕화면을 Python의 working directory로 설정

```
>>> import os
```

```
>>> os.chdir("C:\\Users\\hjkim\\Desktop")
```

```
>>> f = open("foo.txt", "w")
```

```
>>> f.close()
```

바탕화면까지의 Path를 알고 싶으면 바탕화면에 있는 파일을 mouse의 오른쪽 버튼으로 click 한후에 "속성" 을 선택

- 특정 directory 속에 file 을 생성하려면 open() 에서 file생성위치의 absolute path를 명시

```
>>> myfile = open("C:\\Users\\hjkim\\Desktop\\foo.txt", "w+")
```

```
>>> myfile.close()
```

Open Mode of File

- 바탕화면이 Python의 working directory로 설정되었다고 가정하자

```
>>> f = open("foo.txt", "w")
>>> f.close()
```

파일 객체 = open(파일 이름, 파일 열기 모드)

파일열기모드	설명
r	읽기모드 - 파일을 읽기만 할 때 사용
w	쓰기모드 - 파일에 내용을 쓸 때 사용
a	추가모드 - 파일의 마지막에 새로운 내용을 추가

W+: read & write를 위해

**r+는 read & write 이지만
(파일이 이미 존재할 때만 사용 가능)**

Example

```
>>> myfile = open("foo.txt", "w")
>>> myfile.write("Python is good.\n Yes Great!\n ")
>>> myfile.close()
```

➔foo.txt라는 파일이 생성되고, 아래 두줄이 file에 write

“Python is good”

“Yes, Great!”

Keyboard Symbol Names

- ~ : Tilde
- @ : At symbol, Ampersat
- # : Sharp, Hash, Octothorpe
- & : Ampersand
- (: Open Parenthesis
- { : Open Brace, Curly Bracket
- [: Open Bracket
- | : Pipe, Vertical bar
- ' : Apostrophe, Single Quote
- " : Quote, Double Quote
- / : Slash, Forward Slash
- ⧻ or ₩ : Reverse Slash, Back Slash
- * : Asterisk
- - : Hyphen

₩ : Won Currency Symbol
¥ : Yen Currency Symbol

파일을 쓰기 모드로 열어 출력값 적기

foo.txt는 바탕화면에

```
# writedata.py
f = open("foo.txt", "w")
for i in range(1, 11):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```

foo.txt

```
1번째 줄입니다.\n
2번째 줄입니다.\n
3번째 줄입니다.\n
...
10번째 줄입니다.\n
```

파일에 새로운 내용 추가하기

```
# adddata.py
f = open("foo.txt", "a")
for i in range(11, 20):
    data = "%d번째 줄입니다.\n" % i
    f.write(data)
f.close()
```

foo.txt

```
1번째 줄입니다.\n
2번째 줄입니다.\n
3번째 줄입니다.\n
...
10번째 줄입니다.\n
11번째 줄입니다.\n
...
19 번째 줄입니다.\n
```

readline() 함수 사용하기

```
# readline.py
f = open("foo.txt", "r")
line = f.readline()
print(line)
f.close()
```

만약 모든 라인을 읽어서 화면에 출력하고 싶다

```
# readline_all.py
f = open("foo.txt", "r")
while True:
    line = f.readline()
    if not line: break
    print(line)
f.close()
```

foo.txt는 바탕화면에

foo.txt

```
1번째줄입니다.\n
2번째줄입니다.\n
3번째줄입니다.\n
...
10번째줄입니다.\n
11번째줄입니다.\n
...
19 번째줄입니다.\n
```

File Pointer & 3 Marks in File

foo.txt는 바탕화면에

- foo.txt가 바탕화면에 있다면

File pointer



“Python is good”
“Yes, Great!”

file의 내부에는
시작부분에 `beginning_of_file` mark,
각 line뒤에는 `end_of_line` mark,
file의 맨뒤에는 `end_of_file` mark
숨어 있다

- 먼저 file을 열어야 한다

```
myfile = open("C :/Users/hjk/Desktop/foo.txt", "r" )
```

처음에는 myfile (**file pointer**)가 **beginning_of_file**에 위치.

`readline()`을 할때마다 file pointer가 한줄씩 전진

```
myfile.readline()      # shows you “Python is good”
```

```
myfile.readline()      # shows you “Yes, Great!”
```

```
myfile.readline()      # shows you “”
```

** `end_of_file` mark에 file pointer가 도착한후에 또 처음부터 데이터를 읽으려면
`myfile.seek(0.0)` 하면 file pointer가 `beginning_of_file` 위치로 돌아간다

** `myfile.close()`를 하고 다시 `open(...)`으로 file을 열면 file의 맨처음으로 간다

readlines() 함수 이용하기

foo.txt는 바탕화면에

두 번째 방법은 readlines() 함수를 이용하는 방법이다. 다음의 예를 보자.

```
f = open("foo.txt", "r")
lines = f.readlines()
for line in lines:
    print(line)
f.close()
```

새파일.txt

```
1번째줄입니다.\n
2번째줄입니다.\n
3번째줄입니다.\n
...
10번째줄입니다.\n
```

readlines() 함수는 파일의 모든 라인을 읽어서 각각의 줄을 요소로 갖는 리스트로 리턴한다. 따라서 위의 예에서 lines는 ["1 번째 줄입니다.\n", "2 번째 줄입니다.\n", ..., "10 번째 줄입니다.\n"] 라는 리스트가 된다. f.readlines()에서 f.readline()과는 달리 s가 하나 더 붙어 있음에 유의하자.

read() 함수 이용하기

세 번째 방법은 read() 함수를 이용하는 방법이다. 다음의 예를 보자.

```
f = open("foo.txt", "r")
data = f.read()
print(data)
f.close()
```

f.read()는 파일의 내용 전체를 문자열로 리턴한다. 따라서 위 예의 data는 파일의 전체 내용

open() with “with” statement

```
f = open("foo.txt", 'w')  
f.write("Life is too short, you need python")  
f.close()
```

```
with open("foo.txt", "w") as f:  
    f.write("Life is too short, you need python")
```

위와 같이 with문을 이용하면 with 블록을 벗어나는 순간 열린 파일 객체 f가 자동으로 close되어 편리하다. (※ with구문은 파이썬 2.5부터 지원됨)

Reading from a File: txt file

pi_digits.txt이 바탕화면에 있다면

```
pi_digits.txt 3.1415926535
               8979323846
               2643383279
```

file_reader.py

```
with open('pi_digits.txt') as file_object:
    contents = file_object.read()
    print(contents)
```



```
3.1415926535
8979323846
2643383279
```



```
with open('pi_digits.txt') as file_object:
    contents = file_object.read()
    print(contents.rstrip())
```



\n 제거



```
3.1415926535
8979323846
2643383279
```

String의 공백제거: rstrip(), lstrip(), strip()

```
>>> str = "this is string example....wow!!!_"
```

```
>>> print(str.rstrip())
```

```
this is string example....wow!!!
```

```
>>> str = "88888888this is string example....wow!!!88888888"
```

```
>>> print(str.rstrip('8'))
```

```
88888888this is string example....wow!!!
```

```
>>> str = "HJ Kim \n \n_"
```

```
>>> print(str.rstrip())
```

```
HJ Kim
```

- Python on Linux OS

지금 coding하는 program이
text_files directory에 있는 경우

```
with open('text_files/filename.txt') as file_object:
```

```
file_path = '/home/ehmatthes/other_files/text_files/filename.txt'  
with open(file_path) as file_object:
```

- Python on Window OS

```
with open('text_files\filename.txt') as file_object:
```

```
file_path = 'C:\Users\ehmatthes\other_files\text_files\filename.txt'  
with open(file_path) as file_object:
```

Reading from a File: Reading Line by Line

pi_digits.txt

3.1415926535
8979323846
2643383279

pi_digits.txt이 바탕화면에 있다면

file_reader.py

```
❶ filename = 'pi_digits.txt'

❷ with open(filename) as file_object:
❸     for line in file_object:
        print(line)
```

3.1415926535

8979323846

2643383279

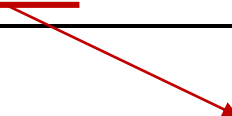
```
filename = 'pi_digits.txt'

with open(filename) as file_object:
    for line in file_object:
        print(line.rstrip())
```

3.1415926535

8979323846

2643383279

 \n 제거

Reading from a File: Making a List of Lines from a File

pi_digits.txt이 바탕화면에 있다면

```
filename = 'pi_digits.txt'
```

```
with open(filename) as file_object:
```

❶

```
    lines = file_object.readlines()
```

❷

```
for line in lines:  
    print(line.rstrip())
```



```
3.1415926535  
8979323846  
2643383279
```

```
filename = 'pi_digits.txt'
```

```
with open(filename) as file_object:
```

```
    lines = file_object.readlines()
```



```
3.1415926535 8979323846 2643383279  
36
```

❶

```
pi_string = ''
```

❷

```
for line in lines:  
    pi_string += line.rstrip()
```

❸


```
print(pi_string)  
print(len(pi_string))
```

Writing to an Empty File

programming.txt이 바탕화면에 있다면

```
write_message.py  filename = 'programming.txt'

❶ with open(filename, 'w') as file_object:
❷     file_object.write("I love programming.")
```




programming.txt

I love programming.

```
filename = 'programming.txt'

with open(filename, 'w') as file_object:
    file_object.write("I love programming.")
    file_object.write("I love creating new games.")
```




programming.txt

I love programming.I love creating new games.

```
filename = 'programming.txt'

with open(filename, 'w') as file_object:
    file_object.write("I love programming.\n")
    file_object.write("I love creating new games.\n")
```



programming.txt

I love programming.
I love creating new games.

Writing to an Empty File: Appending a File

programming.txt

```
I love programming.  
I love creating new games.
```

write_
message.py

```
filename = 'programming.txt'  
  
❶ with open(filename, 'a') as file_object:  
❷     file_object.write("I also love finding meaning in large datasets.\n")  
     file_object.write("I love creating apps that can run in a browser.\n")
```

programming.txt

```
I love programming.  
I love creating new games.  
I also love finding meaning in large datasets.  
I love creating apps that can run in a browser.
```

Ch 11: File IOs and Exception in Python

- File IO
 - Text File Manipulation
 - Big File with Numbers
 - CSV File
- Exception Handling

π Computation in Python

Remember?

$$\sum_{n=1}^{\infty} \frac{1}{n^2} = \lim_{n \rightarrow \infty} \left(\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \cdots + \frac{1}{n^2} \right) = \frac{\pi^2}{6}$$

```
def pi_series_iter(n) :  
    result = 0  
    for i in range(1, n+1) :  
        result = result + 1/(i**2)  
    return result
```

```
def pi_approx_iter(n) :  
    x = pi_series_iter(n)  
    return (6*x)**(.5)
```

```
def pi_series_r(i) :  
    assert(i >= 0)  
    # base case  
    if i == 0:  
        return 0  
    # recursive case  
    else:  
        return pi_series_r(i-1) + 1 / i**2
```

```
def pi_approx_r(n) :  
    x = pi_series_r(n)  
    return (6*x)**(.5)
```

```
pi_approx_iter(10) == 3.04936163598207)  
pi_approx_iter(100) == 3.1320765318091053)  
pi_approx_iter(1000) == 3.1406380562059946)  
pi_approx_iter(10000) == 3.1414971639472147)
```

Floating Point Number (실수)의 표현

- Fixed Point Representation

고정소수점 방식

소수점 위치를 고정시킨 수 표현 방식

- 소수점 이하 자리수를 일정하게 유지시키면서 실수를 표현
- 소수점(point) 이 항상 같은 위치에 있음

고정소수점 표현

$$23.456_{(10)} = 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2} + 6 \times 10^{-3}$$

$$10.101_{(2)} = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} = 2.625_{(10)}$$

3자리 유효숫자 10진수 부동소수점 표현

$$0.123 \times 10^3$$

```
>>> a = 0.1234567890123456789
>>> a
0.12345678901234568
>>>
>>> b = 1.1234567890123456789
>>> b
1.1234567890123457
>>>
>>> c = 11.1234567890123456789
>>> c
11.123456789012346
```

- Floating Point Representation

부동소수점 방식 (지수표현 방식)

0.12 의 소수점을 가진 실수의 표현은 아래의 다양한 부동소수점 방식이 가능

$$1) 0.12 = 0.12 \times 10^0$$

$$2) 0.12 = 12 \times 10^{-2}$$

$$3) 0.12 = 0.012 \times 10^1$$

```
>>> a = 4.24E10
>>> a = 4.24e-10
```

$$4.24 \times 10^{10}$$

$$4.24 \times 10^{-10}$$

여기서 E 와 e는 혼용가능

Floating-point number는 정수와 실수에 있는 숫자 17개로 제한하는 고정소수점 방식으로 표현하고,

더 정확한것을 원하면 부동소수점방식으로 표현하면 된다.

“decimal” Module

- `getcontext().prec` 을 setting 하면
- Decimal type의 arithmetic은 `getcontext().prec` 에 따른다
- Floating number의 default precision을 control 할수 있다


```
>>> a = 0.1234567890123456789
>>> a
0.12345678901234568
```

```
>>> from decimal import *
>>> getcontext().prec = 6
>>> Decimal(1) / Decimal(7)
Decimal('0.142857')
>>> getcontext().prec = 28
>>> Decimal(1) / Decimal(7)
Decimal('0.1428571428571428571428571428571429')
```

Big file 생성해보기! pi_million_digits.txt

- 첫번째 줄은 3. + 100개 숫자 + end of line.
- 두번째 줄부터는 bb + 100개 숫자 + end of line.
- 총 1만줄

```
3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912983367336244065664308602139494639522473719070217986094370277053921717629317675238467481816589813878620312658590139765597542908601520453809265394054247489868871508114253606189542561834516239602003856400853609892792884261564003386415421850202280061800331135332829076809953120034900160907515892304237784395768008843265796835590469628970986832417691234893746185763020779404936463935252511061395279817833193472808119172981388781242209954996910193473833914176438837089727750144485233004
```

 pi_million_digits - 메모장

소숫점아래 100개 digits

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818
```

1만
lines

```
16589813878620312658590139765597542908601520453809265394054247489868871508114253606189542561834516239602003856400853609892792884261564003386415421850202280061800331135332829076809953120034900160907515892304237784395768008843265796835590469628970986832417691234893746185763020779404936463935252511061395279817833193472808119172981388781242209954996910193473833914176438837089727750144485233004
```

Chudnovsky algorithm

From Wikipedia, the free encyclopedia

$$\frac{1}{\pi} = 12 \sum_{k=0}^{\infty} \frac{(-1)^k (6k)! (545140134k + 13591409)}{(3k)! (k!)^3 (640320)^{3k+3/2}}$$

The **Chudnovsky algorithm** is a fast method for calculating the digits of π . It was published by the [Chudnovsky brothers](#) in 1989,^[1]

```
from decimal import Decimal as Dec, getcontext as gc
def PI(maxK=70, prec=1008, disp=1007): # parameter defaults chosen to
                                         # gain 1000+ digits within a few seconds
    gc().prec = prec
    K, M, L, X, S = 6, 1, 13591409, 1, 13591409
    for k in range(1, maxK+1):
        M = (K**3 - 16*K) * M // k**3
        L += 545140134
        X += -262537412640768000
        S += Dec(M * L) / X
        K += 12
    pi = 426880 + Dec(10005).sqrt() / S
    pi = Dec(str(pi)[:disp]) # drop few digits of precision for accuracy
    return pi
```

소수점 아래 prec 범위까지
계산을 하도록 setting

Dec(n)은 n을 Decimal로 표현

PI()의 display parameter
범위로 trimming하고

```
digits = 1000000
Pi = str(PI(70, digits + 2, digits + 1))
f = open("pi_million_digits.txt", "w")
f.write(Pi[0:100])
f.write("\n")
lines = ((digits + 1) - 100) // 98
if (((digits + 1) - 100) % 98 != 0):
    lines += 1
for i in range(lines):
    start = 100 + (i * 98)
    f.write(" ")
    f.write(Pi[start:start + 98])
    f.write("\n")
```

PI()의 Dec type의 value를
string으로 변환하여 txt file에
저장하는 code

Reading from a File: Large Files

- 첫번째 줄은 3. + 100개 숫자 + end of line.
- 두번째 줄부터는 bb + 100개 숫자 + end of line.
- 총 1만줄

pi_million_digits.txt - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
3.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306
019898 3809525720106548586327886593615338182796823030195203530185296899577362259941389124972177528347913151 55748572424
47802759009 9465764078951269468398352595709825822620522489407726719478268482601476990902640136394437455305068203 49625
40907186494231961 5679452080951465502252316038819301420937621378559566389377870830390697920773467221825625996615014215
16201052652272111660396 665573092547110557853763466820653109896526918620564769312570586356620185581007293606598764861179
26312986080998886874132604721 569516239658645730216315981931951673538129741677294786724229246543668009806769282382806899
61190625454337213153595845068772460 290161876679524061634252257719542916299193064553779914037340432875262888963995879475
57811196358330059408730681216028764962867 446047746491599505497374256269010490377819868359381465741268049256487985561455
82027342092222453398562647669149055628425039127 57710284027998066365825488926488025456610172967026640765590429099456815C
32326092752496035799646925650493681836090032380929345 95889706953653494060340216654437558900456328822505452556405644824E
49887275846101264836999892256959688159205600101655256375678 56672279661988578279484885583439751874454551296563443480396E
55690347119729964090894180595343932512362355081349490043642785271 38315912568989295196427287573946914272534366941532361C
93860381017121585527266830082383404656475880405138080163363887421637140 64354955618689641122821407533026551004241048967E
21025941737562389942075713627516745731891894562835257044133543758575342698699 472547031656613991999682628247270641336222
48639688369032655643642166442576079147108699849157337496488352927693282207629472823 815374099615455987982598910937171262
16975774183023986006591481616404944965011732131389574706208847480236537103115089842799275 442685327797431139514357417221
38363185745698147196210841080961884605456039038455343729141446513474940784884423772175154334260 30669883176833100113310E
9025100158882721647450068207041937615845471231834600726293395505482395571372568402322682130124767945 226448209102356477E
064 7842645676338818807565612168960504161139039063960162022153684941092605387688714837989559999112099164 646441191856E
843271922 3223810158744450528665238022532843891375273845892384422535472653098171578447834215822327020690287232 330053E
721268039226911 0277722610254414922157650450812067717357120271802429681062037765788371669091094180744878140490755178 5
```

```
filename = 'pi_million_digits.txt'

with open(filename) as file_object:
    lines = file_object.readlines()

pi_string = ''
for line in lines:
    pi_string += line.strip()

print(pi_string[:52] + "...")
print(len(pi_string))
```

3.14159265358979323846264338327950288419716939937510...
1000002

"3."을 제외하고 1million 숫자 → 1,000,002개 숫자

Ch 11: File IOs and Exception in Python

- File IO
 - Text File Manipulation
 - Big File with Numbers
 - CSV File
- Exception Handling

CSV File

CSV stands for "comma-separated values". Its data fields are most often separated, or **delimited**, by a **comma**. For example, let's say you had a spreadsheet containing the following

Name	Class	Dorm	Room	GPA
Sally Whittaker	2018	McCarren House	312	3.75
Belinda Jameson	2017	Cushing House	148	3.52
Jeff Smith	2018	Prescott House	17-D	3.20
Sandy Allen	2019	Oliver House	108	3.48

This data could be represented in a CSV-formatted file as follows:

```
Sally Whittaker,2018,McCarren House,312,3.75
Belinda Jameson,2017,Cushing House,148,3.52
Jeff Smith,2018,Prescott House,17-D,3.20
Sandy Allen,2019,Oliver House,108,3.48
```

통합 문서1.xlsx

Excel 통합 문서 (*.xlsx)

Excel 통합 문서 (*.xlsx)

Excel 매크로 사용 통합 문서 (*.xlsm)

Excel 바이너리 통합 문서 (*.xlsb)

Excel 97 - 2003 통합 문서 (*.xls)

XML 데이터 (*.xml)

웹 보관 파일 (*.mht;*.mhtml)

웹 페이지 (*.htm;*.html)

Excel 서식 파일 (*.xltx)

Excel 매크로 사용 서식 파일 (*.xltm)

Excel 97 - 2003 서식 파일 (*.xlt)

텍스트 (탭으로 분리) (*.txt)

유니코드 텍스트 (*.txt)

XML 스프레드시트 2003 (*.xml)

Microsoft Excel 5.0/95 통합 문서 (*.xls)

CSV (쉼표로 분리) (*.csv)

CSV File을 Open하고 작업하는 예제

data.csv에 3 line으로

A, B, C

B, A, E

C, A, F

가 들어 있고

A의 친구는 B, C

B의 친구는 A, E

C의 친구는 A, F

를 dictionary로 표현하고 싶다면

```
Friend_Dict = { }
```

```
with open('data.csv') as csvFile:
```

```
    for line in csvFile:
```

```
        rowArray = line.strip().split(',')
```

```
        Friend_Dict[rowArray[0]] = rowArray[1:]
```

➔ csvFile = open("data.csv", "r+") 와 같다고 보면됨

➔ file안에 있는 모든 line에 대해서

➔ rowArray 는 ["A","B","C"] 방식으로 값을 가짐

strip() removes newline \n 제거

➔ { "A": ["B", "C"] } 스타일의 dictionary 생성

Adds dictionary as Key : First word in the row and
Value : Row without first word

위에서 for loop 문장이 끝나면

Friend_Dict ➔ { "A": ["B", "C"], "B": ["A", "E"], "C": ["A", "F"] }

csv.reader() & csv.writer() in "csv" Module!

**** csv file을 read할때에 one line을 list로 받아서 처리**
**** list를 one line의 csv 형태로 csv file에 write 할수 있음**

csv.reader(csvfile): csvfile에서 read를 위한 file pointer 준비

```
import csv
Friend_Dict = {}
with open('friends.csv', 'r+') as csvdataFile
    friend_reader = csv.reader(csvdataFile)
    for row in friend_reader:
        rowArray = row
        Friend_Dict[rowArray[0]] = rowArray[1:]
```

- friend_reader는 file pointer
- 한줄씩 list로 가져와서
- dictionary 속으로 집어넣기

csv.writer(csvfile): csvfile를 write를 위한 file pointer 준비

```
with open('result.txt ' , 'w+' ) as csvresultFile:
```

```
    foaf_writer = csv.writer(csvresultFile) → foaf_writer는 file pointer
```

```
    .....
```

```
    foafList = ["bob", "kim", "mike"]
```

```
    foaf_writer.writerow(foafList) → result.txt에 1줄로 "bob", "kim", "mike"를 write를 함.
```

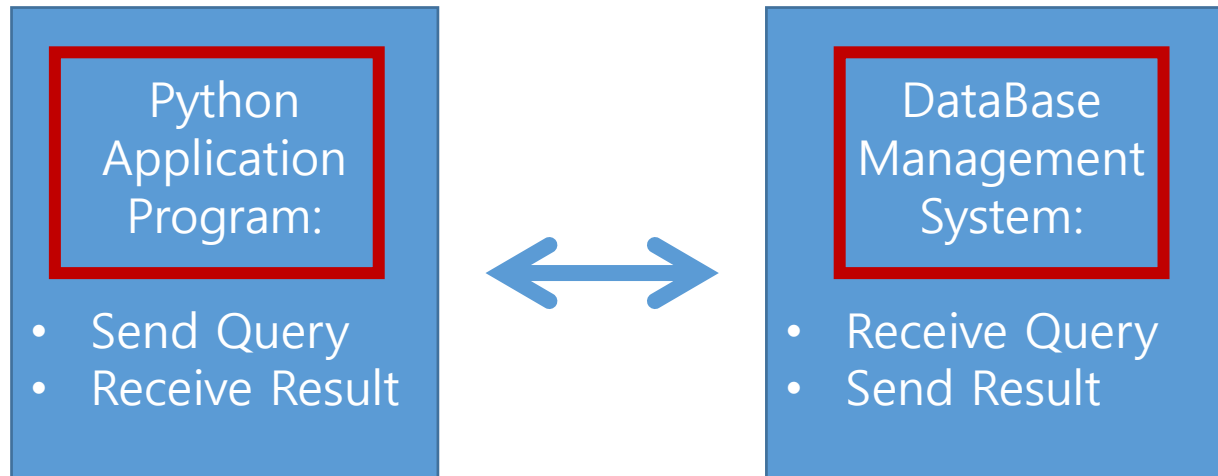
Ch 11: File IOs and Exception in Python

- File IO
 - Text File Manipulation
 - Big File with Numbers
 - CSV File
- Exception Handling

Exception Handling in Python

Error = Exception

Interpreter 방식의 언어에서
Run-time에 flexible한
상황전개를 위해서....



Exceptions

division.py

```
print(5/0)
```

Of course Python can't do this, so we get a traceback:

```
Traceback (most recent call last):  
  File "division.py", line 1, in <module>  
    print(5/0)
```

❶ ZeroDivisionError: division by zero

```
>>> f = open("나없는파일", 'r')
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
FileNotFoundError: [Errno 2] No such file or directory: '나없는파일'
```

```
>>> a = [1,2,3]
```

```
>>> a[4]
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
IndexError: list index out of range
```

Exception Handling

- Full list of [standard built-in exceptions](https://docs.python.org/3/library/exceptions.html) (users may create their own)

<https://docs.python.org/3/library/exceptions.html>

$$ax^2 + bx + c = 0$$

Quadratic Formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- In the quadratic equation example, remember [ZeroDivisonError](#), [ValueError](#)
- Other types of exceptions may arise
 - not entering the right number of parameters (“[unpack tuple of wrong size](#)”)
 - entering an identifier instead of a number ([NameError](#))
 - entering an invalid Python expression ([TypeError](#))
 - Refer to sample code *quadratic6.py*

Exception Handling: Intuition

[1/3]

```
x = 5 + "ham"
```

Traceback (most recent call last):

File "<pyshell#0>", line 1, in <module>

x = 5 + 'ham'

TypeError: unsupported operand type(s) for +: 'int' and 'str'

```
try:
```

```
    x = 5 + "ham"
```

```
except:
```

```
    print("darn it")
```

TRY

- 'try' TO EXECUTE the code below...
- May be used anywhere that keyboarduser input is required

EXCEPT

- CATCHES all ERRORS or can just catch a specific error
- May be used anywhere that keyboarduser input is required

darn = damn (비난하다)

Exception Handling: Intuition

[2/3]



PASS

- says to IGNORE and move on
- may be used in For, While, Try/Except instances

```
>>> try:  
    x = 5 + 'ham'  
except:  
    pass
```

```
>>> def doesNothing():  
    pass  
  
>>> doesNothing()
```

오류 회피하기

프로그래밍을 하다 보면 특정 오류가 발생할 경우 그냥 통과시켜야 할 때가 있을 수 있다. 다음의 예를 보자.

```
try:  
    f = open("나없는파일", 'r')  
except FileNotFoundError:  
    pass
```

try문 내에서 FileNotFoundError가 발생할 경우 pass를 사용하여 오류를 그냥 회피하도록 한 예제

RAISE
- FORCE AN ERROR
to occur
`raise TypeError("hahaha")`

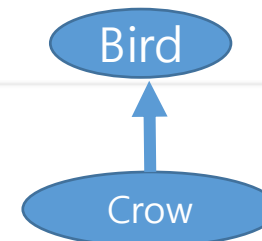
```
>>> raise TypeError("hahahaha")  
  
Traceback (most recent call last):  
  File "<pyshell#0>", line 1, in <module>  
    raise TypeError("hahahaha")  
TypeError: hahahaha  
>>>
```

오류 일부러 발생시키기

이상하게 들리겠지만 프로그래밍을 하다 보면 종종 오류를 일부러 발생시켜야 할 경우도 생긴다. 파이썬은 `raise`라는 명령어를 이용해 오류를 강제로 발생시킬 수 있다.

예를 들어 `Bird`라는 클래스를 상속받는 자식 클래스는 반드시 `fly`라는 함수를 구현하도록 만들고 싶은 경우(강제로 그렇게 하고 싶은 경우)가 있을 수 있다. 다음 예를 보자.

```
class Bird:  
    def fly(self):  
        raise NotImplementedError
```



"finally" Clause

"finally clause"는 exception 발생여부와 관계없이 꼭 수행

```
f = open('foo.txt', 'w')
try:
    # 무언가를 수행한다.
finally:
    f.close()
```

foo.txt라는 파일을 쓰기 모드로 연 후에 try문이 수행된 후 예외 발생 여부에 상관없이 finally절에서 f.close()로 열린 파일을 닫을 수 있다.

```
try:
    x = 5 + "ham"

except ZeroDivisionError:
    print("darn it")

finally:
    print("Let's go further!")
```

```
try:
    x = 5 + "ham"

except TypeError:
    print("darn it")

finally:
    print("Let's go further!")
```

Multiple Error Handling

try문 내에서 여러개의 오류를 처리하기 위해서는 다음과 같은 구문을 이용한다.

```
try:
    ...
except 발생 오류1:
    ...
except 발생 오류2:
    ...
```

```
try:
    a = [1,2]
    print(a[3])
    4/0
except ZeroDivisionError:
    print("0으로 나눌 수 없습니다.")
except IndexError:
    print("인덱싱 할 수 없습니다.")
```

a는 2개의 요소값을 가지고 있기 때문에 `a[3]` 는 `IndexError` 를 발생시키므로 "인덱싱 할 수 없습니다."라는 문자열이 출력될 것이다. 인덱싱 오류가 먼저 발생했으므로 `4/0` 으로 발생하는 `ZeroDivisionError` 는 발생하지 않았다.

Exception Handling: Formal Syntax

- To explicitly filter out all error types

```
try:  <body>
except :
    <exception handling>
```

- To cope with multiple errors

```
try:  <body>
except <error_1> :
    <exception handling_1>
...
except <error_n> :
    <exception handling_n>
```

- To explicitly filter out error types and store the error as a variable

```
try:
    <body>
except <error_1> as <variable_1> :
    <exception handling_1>
...
except <error_n> as <variable_n> :
    <exception handling_n>
```

Exception Handling: Code Example

```
# quadratic5.py
#     A program that computes the real roots of a quadratic equation.
#     Illustrates exception handling to avoid crash on bad inputs

import math

def quadratic5():
    print("This program finds the real solutions to a quadratic\n")

    try:
        a, b, c = eval(input("Please enter the coefficients (a, b, c): "))
        discRoot = math.sqrt(b * b - 4 * a * c)
        root1 = (-b + discRoot) / (2 * a)
        root2 = (-b - discRoot) / (2 * a)
        print("\n The solutions are:", root1, root2)
    except ValueError:
        print("\n No real_number roots")
```

$$ax^2 + bx + c = 0$$

Quadratic Formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Using Exceptions to Prevent Crashed Situation [1/2]

division.py

```
print("Give me two numbers, and I'll divide them.")
print("Enter 'q' to quit.")
```

```
while True:
```

- ❶

```
    first_number = input("\nFirst number: ")
    if first_number == 'q':
        break
```
- ❷

```
    second_number = input("Second number: ")
    if second_number == 'q':
        break
```
- ❸

```
    answer = int(first_number) / int(second_number)
    print(answer)
```

```
Give me two numbers, and I'll divide them.
Enter 'q' to quit.
```



Crash!

```
First number: 5
Second number: 0
Traceback (most recent call last):
  File "division.py", line 9, in <module>
    answer = int(first_number) / int(second_number)
ZeroDivisionError: division by zero
```

Using Exceptions to Prevent Crashed Situation [2/2]

division.py

```
print("Give me two numbers, and I'll divide them.")
print("Enter 'q' to quit.")

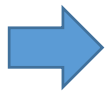
while True:
    first_number = input("\nFirst number: ")
    if first_number == 'q':
        break
    second_number = input("Second number: ")
    ❶ try:
        ❷ answer = int(first_number) / int(second_number)
        ❸ except ZeroDivisionError:
            print("You can't divide by 0!")
        else:
            print(answer)
```

Give me two numbers, and I'll divide them.
Enter 'q' to quit.

First number: 5
Second number: 0
You can't divide by 0!

First number: 5
Second number: 2
2.5

First number: q



Prevent Crash!

Handling the FileNotFoundError Exception

alice.py

```
filename = 'alice.txt'

with open(filename) as f_obj:
    contents = f_obj.read()
```



If alice.txt does not exist, Crash!

```
Traceback (most recent call last):
  File "alice.py", line 3, in <module>
    with open(filename) as f_obj:
FileNotFoundError: [Errno 2] No such file or directory: 'alice.txt'
```

alice.py

```
filename = 'alice.txt'

try:
    with open(filename) as f_obj:
        contents = f_obj.read()
except FileNotFoundError:
    msg = "Sorry, the file " + filename + " does not exist."
    print(msg)
```



Sorry, the file alice.txt does not exist.

Prevent Crash!

Working with Multiple Files and Exception Handling Routine

```
def count_words(filename):  
    """Count the approximate number of words in a file."""  
    try:  
        with open(filename) as f_obj:  
            contents = f_obj.read()  
    except FileNotFoundError:  
        msg = "Sorry, the file " + filename + " does not exist."  
        print(msg)  
    else:  
        # Count approximate number of words in the file.  
        words = contents.split()  
        num_words = len(words)  
        print("The file " + filename + " has about " + str(num_words) +  
              " words.")  
  
filename = 'alice.txt'  
count_words(filename)
```

word_count.py

```
filenames = ['alice.txt', 'siddhartha.txt', 'moby_dick.txt', 'little_women.txt']  
for filename in filenames:  
    count_words(filename)
```

The file alice.txt has about 29461 words.

Sorry, the file siddhartha.txt does not exist.

Prevent Crash!

The file moby_dick.txt has about 215136 words.

The file little_women.txt has about 189079 words.