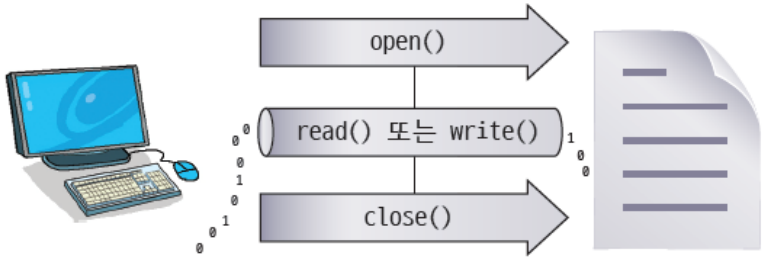
10장 파일 입출력과 예외처리

파일 입출력. p339

- 변수는 프로그램이 실행되는 동안 데이터를 저장하는 좋은 방법이지만 프로그램이 종료된 후에도 데이터를 유지하려면 파일에 저장해야 한다.

파일 열고 닫기. p339

- 파일에 저장된 데이터를 읽는 방법
  - open() 함수 이용
  - open()은 파일 이름을 받아서 파일 객체를 생성한 후에 반환한다. 파일이 열리면 파일에서 데이터를 읽거나 쓸 수 있다. 파일과 관련된 작업이 모두 종료되면 close()를 호출하여서 파일을 닫아야 한다.

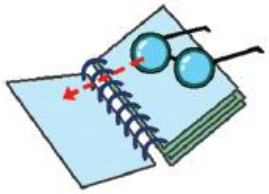



```
f = open("input.txt", "r")
```

파일객체 파일의 이름 파일 모드

파일 열고 닫기. p340

- 파일 모드의 종류



"r"

파일의 처음 부터 읽는다.

"w"

파일의 처음 부터 쓴다.
만약 파일이 존재하면 기존의 내용이 지워진다.

"a"

파일의 끝에 쓴다.
파일이 없으면 생성 된다.

- 파일이 현재 작업 디렉토리에 없으면 경로를 써준다.

```
f = open(“d:\\input.txt”, “r”)
```

파일에 쓰기. p340

- “w” 모드로 열면 write() 메소드를 이용하여 파일에 텍스트를 쓸 수 있다.

```
>>> f = open("test.txt", "w")
>>> f.write("파이썬은 강력합니다.\n")
12

>>> f.close()
```

test.txt

```
파이썬은 강력합니다.\n
```

파일에서 읽기. p341

- read() 사용

```
>>> f = open("test.txt", "r")
>>> s = f.read()
>>> s
'파이썬은 강력합니다.\n'
>>> f.close()
```

test.txt

파이썬은 강력합니다.\n

- read()가 반환하는 것은 문자열. 따라서 문자열이 제공하는 여러 가지 연산을 사용할 수 있다.

```
>>> f = open("test.txt", "r")
>>> s = f.read()
>>> s[:3]
'파이썬'
>>> f.close()
```

파일에 추가하기. p341

- 파일 모드 'a' 를 사용

```
>>> f = open("test.txt", "a")
>>> f.write("파이썬은 강력합니다.\n")
12

>>> f.close()
>>> f = open("test.txt", "r")
>>> f.read()
'파이썬은 강력합니다.\n파이썬은 간결합니다.\n'
>>> f.close()
```

test.txt

```
파이썬은 강력합니다.\n
파이썬은 간결합니다.\n`
```

with 문으로 파일 열기. p341

- with 문은 with 문 내의 블록이 종료될 때 파일이 자동으로 닫친다. close()를 명시적으로 호출할 필요는 없다.

```
with open("test.txt", "w") as f:
        f.write("파이썬은 간결합니다.\n")

        ....                    # f.close()를 호출하지 않아도 된다.
```

파일을 연 후에는 무엇을 하는가. p342

- 파일을 열어서 문자열을 쓰거나 읽을 수 있다. 따라서 문자열 관련 많은 작업들을 할 수가 있게 된다.

test.txt

```
파이썬은 강력합니다.\n
파이썬은 간결합니다.\n
파이썬은 배우기 쉽습니다.\n
```



```
[ "파이썬은 강력합니다.",
"파이썬은 간결합니다.",
"파이썬은 배우기 쉽습니다." ]
```

```
f = open("test.txt", "r")
s = f.read()
myList = s.split("\n")
print(myList)
f.close()
```

```
['파이썬은 강력합니다.', '파이썬은 간결합니다.', '파이썬은 배우기 쉽습니다.', '']
```

문자 인코딩. p343

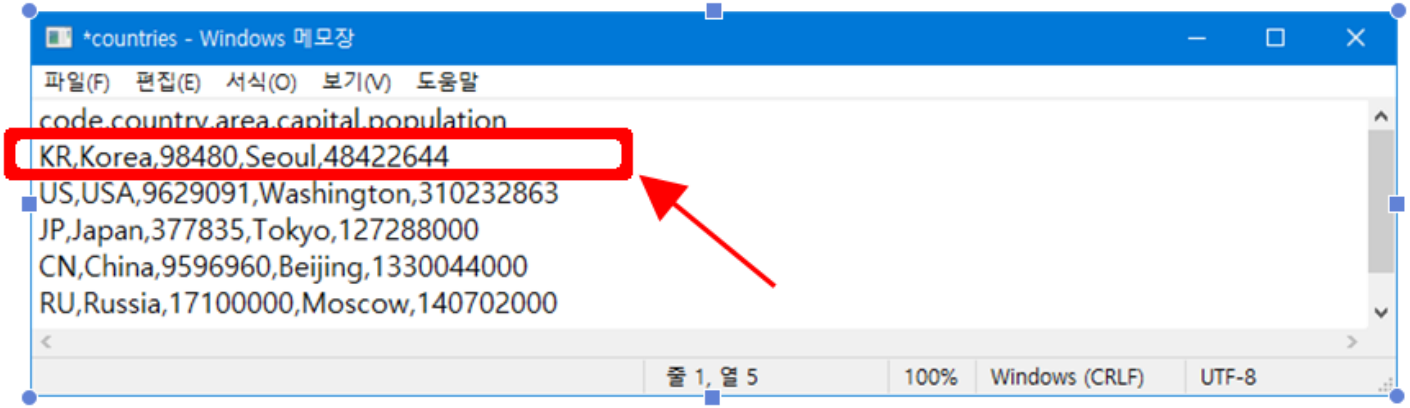
- 텍스트 파일을 처리할 때 문자 인코딩이 중요한 이유는 인코딩에 따라서 동일한 파일이라도 파일을 이루는 바이트가 달라지기 때문이다. 파이썬에서는 운영체제로부터 문자 인코딩 설정을 가져온다.
- 과거의 텍스트 파일은 거의 ANSI 코드이다. ANSI 코딩은 유니코드가 나오기 전에 많이 사용하였던 완성형 코드이다.
- 사용자가 UTF-8 기반의 파일을 열 때는 특별히 다음과 같이 인코딩을 지정하여야 한다.

```
infile = open("input.txt", "r", encoding="utf-8")
```

Lab: 행맨. p344

CSV 파일. p346

- 테이블 형식의 데이터를 저장하고 이동하는 데 사용되는 구조화된 텍스트 파일 형식
- Microsoft Excel와 같은 스프레드 시트에 적합한 형식



CSV 데이터 처리하기. p346

- CSV 파일은 판다스(13장)를 이용해서 읽는 것이 최선
- 여기서는 순수 파이썬을 이용한 처리 방법을 소개
  - 파이썬 모듈 csv는 CSV reader와 CSV writer를 제공한다.
  - 두 객체 모두 파일 핸들을 첫 번째 매개 변수로 사용한다. 필요한 경우 delimiter 매개 변수를 사용하여 구분자를 제공할 수 있다.
  - CSV 파일의 각 행의 데이터가 리스트에 저장되어 전달된다.

```
import csv                          # CSV 모듈을 불러온다

F = open( ' weather.csv ' )         # CSV 파일을 열어서 f에 저장한다.
data = csv.reader(f)                # reader() 함수를 이용하여 읽는다.
for row in data:
        print(row)
f.close()
```

```
['날짜', '지점', '평균기온(℃)', '최저기온(℃)', '최고기온(℃)']
['1980-04-01', '108', '6.5', '3.2', '11.7']
['1980-04-02', '108', '6.5', '1.4', '12.9']
['1980-04-03', '108', '11.1', '4.1', '18.4']
['1980-04-04', '108', '15.5', '8.6', '21']
...
```

CSV 데이터 처리하기. p347

- 서울이 언제 가장 추웠는지를 조사해보자. -> 교재코드 수정
  - 헤더를 제거하고 나머지 데이터만을 읽는다. – next()
  - 반복하면서 최저기온과 해당날짜를 찾는다 – row[3], row[1]

```
import csv

f = open('weather.csv')
data = csv.reader(f)
header = next(data)
temp = 1000
when = ""
for row in data:
    if temp > float(row[3]):
        temp = float(row[3])
        when = row[0]
print(temp, "날짜 :", when)
f.close()
```

-19.2 날짜: 1986-01-05

Lab 인구 데이터. p349

- 2019년 4월 현재 우리나라의 행정구역별 인구 분포가 저장된 ages.csv 파일이 있다. 이 파일에서 서울의 인구 구조만을 추출해보자.

ages - 메모장

파일(F) 편집(E) 서식(O) 보기(V) 도움말

```
행정구역,2019년04월_계_총인구수,2019년04월_계_연령구간인구수,2019년04월_계_0~9세,2019년04월_계_10~19세,2019년04월_계_20~29세,2019년04월_계_30~39세,2019년04월
전국 (0000000000),"51,836,763","51,836,763","4,252,921","5,068,750","6,815,660","7,198,993","8,438,563","8,658,999","6,070,217","3,524,973","1,567,865","220,388","19,434"
서울특별시 (1100000000),"9,766,886","9,766,886","680,098","843,949","1,458,045","1,535,973","1,579,826","1,549,531","1,159,875","667,968","247,754","37,855","6,012"
부산광역시 (2600000000),"3,431,750","3,431,750","248,549","291,745","441,953","447,924","522,444","590,864","494,077","276,664","102,489","13,323","1,718"
대구광역시 (2700000000),"2,454,154","2,454,154","192,597","245,182","328,309","313,937","399,715","431,461","297,457","166,821","69,727","8,298",650
인천광역시 (2800000000),"2,956,700","2,956,700","248,107","288,855","409,266","429,231","492,213","516,940","323,497","166,561","70,374","10,797",859
광주광역시 (2900000000),"1,458,940","1,458,940","129,244","169,087","206,651","197,954","245,937","232,778","147,936","87,809","36,134","5,032",378
대전광역시 (3000000000),"1,485,509","1,485,509","126,303","157,801","213,885","205,895","246,611","244,338","162,156","85,768","37,208","5,218",326
울산광역시 (3100000000),"1,152,293","1,152,293","106,946","119,309","151,514","164,088","192,791","211,282","128,127","54,950","20,345","2,781",160
세종특별자치시 (3600000000),"324,417","324,417","45,618","37,984","34,845","59,706","60,619","39,997","25,344","12,826","6,444",979,55
경기도 (4100000000),"13,129,508","13,129,508","1,200,597","1,373,979","1,754,439","1,940,021","2,285,992","2,170,600","1,327,051","718,326","309,784","44,976","3,743"
강원도 (4200000000),"1,540,794","1,540,794","111,278","147,872","181,271","173,485","231,350","271,670","212,545","136,287","64,455","9,832",749
충청북도 (4300000000),"1,599,488","1,599,488","132,016","157,987","200,429","203,465","246,165","273,262","198,738","118,363","60,461","8,056",546
충청남도 (4400000000),"2,125,912","2,125,912","186,524","211,490","247,176","285,662","329,783","340,549","252,582","164,559","94,159","12,616",812
전라북도 (4500000000),"1,829,273","1,829,273","141,184","187,390","215,583","204,252","278,583","305,612","236,060","162,419","85,484","11,923",783
전라남도 (4600000000),"1,873,183","1,873,183","142,949","180,002","203,878","200,761","269,034","319,797","249,472","191,037","101,881","13,581",791
경상북도 (4700000000),"2,670,375","2,670,375","204,935","241,542","296,155","315,047","395,784","468,033","371,092","235,602","124,577","16,713",895
경상남도 (4800000000),"3,368,933","3,368,933","292,679","341,522","390,403","434,824","548,317","582,708","413,205","234,962","114,687","14,926",700
제주특별자치도 (5000000000),"668,648","668,648","63,297","73,054","81,858","86,768","113,399","109,577","71,003","44,051","21,902","3,482",257
```

Windows (CRLF) Ln 1, Col 1 100%

디렉토리 작업. p350

- os 모듈에서 제공하는 도구들을 사용

현재 작업 디렉토리. p350

- 현재 작업 디렉토리(CWD: Current Working Directory)
- getcwd() 메소드 사용

```
>>> import os
>>> os.getcwd()
'D:\\test'
>>> os.getcwdb()              # 바이트 배열 객체로 가져오기
b'D:\\test'
```

디렉토리 변경. p350

- chdir() 메소드 사용

```
>>> os.chdir('D:\\sources')
>>> print(os.getcwd())
D:\source
```

디렉토리 안의 파일 나열. p351

- listdir() 메소드 사용.
- 서브 디렉터리 및 파일 리스트를 반환한다.

```
>>> os.listdir()
['kr', 'PackageTest.java’]

>>> os.listdir('D:\\')
['$RECYCLE.BIN', '.metadata', '10.1.1.335.3398.pdf', ... ]
```

```
for filename in os.listdir() :          # 리스트에서 파일 이름을 하나씩 꺼내서 출력
        print(filename)
```

```
if os.path.isfile(filename):            # 파일만 처리
        print(“파일입니다.”)
```

디렉토리 안의 파일 나열. p351

```
import os

cwd = os.getcwd()
files = os.listdir()
for name in files :                 # .jpg 파일만 찾아서 출력
        if os.path.isfile(name) :
                if name.endswith(".jpg") :
                        print(name)
```

```
DSC04886_11.jpg
DSC04886_12.jpg
DSC04886_13.jpg
```

새 디렉토리 만들기. p352

- mkdir() 메소드 사용

```
>>> os.mkdir('test')
>>> os.listdir()
['kr', 'PackageTest.java', 'test']
```

디렉토리 또는 파일 이름 바꾸기. p352

- rename() 메소드 사용

```
>>> os.listdir()
['kr', 'PackageTest.java', 'test']
>>> os.rename('test','test2')
>>> os.listdir()
['kr', 'PackageTest.java', 'test2']
```

디렉토리 또는 파일 제거. p352

- remove() 메소드 : 파일을 제거
- rmdir() 메소드 : 빈 디렉터리를 제거

```
>>> os.listdir()
['kr', 'PackageTest.java', 'test2']
>>> os.remove('PackageTest.java')
>>> os.listdir()
['kr', 'test2']
>>> os.rmdir('test2')
['kr']
```

Lab 디렉토리 안의 파일 처리. P353

- 파일 중에서 "Python"을 포함하고 있는 줄이 있으면 파일의 이름과 해당 줄을 출력한다. -> 교재코드 수정 : 도전문제 내용 추가

```
file.py :         if "Python" in e:
summary.txt : The joy of coding Python should be in seeing short
summary.txt : Python is executable pseudocode.
```

```
import os
arr = os.listdir()

for f in arr:
    if f.endswith(".txt") or f.endswith(".py"):     # 대상 파일을 제한
        infile = open(f, "r", encoding="utf-8")
        linecnt = 0
        for line in infile:
            linecnt += 1            # 줄번호
            e = line.rstrip()       # 오른쪽 줄바꿈 문자를 없앤다.
            if "Python" in e:
                print(f, linecnt, ":", e)
        infile.close()
```

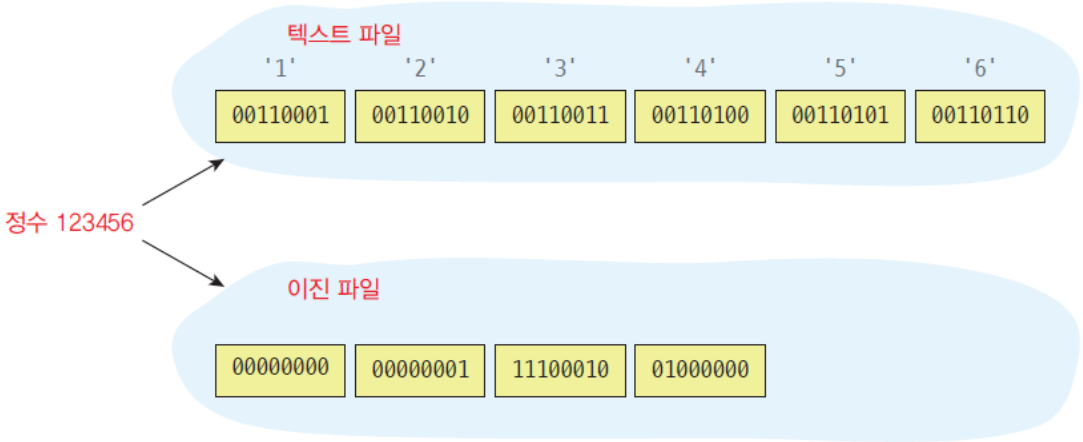
Lab 수학문제지 100개 만들기. p354

- 간단한 사칙 연산 문제가 10개가 들어 있는 문제지가 필요하다. 100명의 초등학생에게 서로 다른 문제지를 주고 싶다. 파이썬으로 만들 수 있을까? 쓰기용 파일을 100개 만들고 여기에 사칙 연산 문제를 랜덤하게 출제하면 된다.

```
다음의 문제를 풀어서 제출하세요
이름:       점수:

30 - 17 =
82 + 47 =
69 * 11 =
88 / 40 =
80 / 35 =
2 / 73 =
70 * 87 =
13 * 93 =
85 - 35 =
4 + 11 =
```

이진 파일. p355

- 텍스트 파일 : 모든 정보가 문자열로 변환되어 파일에 저장
- 이진 파일(binary file)은 데이터가 직접 저장되어 있는 파일. 즉 정수 123456는 문자열로 변환되지 않고 0x0001E240와 같은 이진수 형태로 그대로 파일에 기록.
  - 장점 - 효율성, 저장 공간도 더 적게 차지
  - 단점 – 파일 내용을 확인하기 어렵다. 컴퓨터 시스템마다 다르다.



이진 파일. p356

- 이진 파일에서 읽기

```
infile = open(filename, "rb")

입력 파일에서 8 바이트를 읽기
bytesArray = infile.read(8)

첫 번째 바이트를 꺼내기
byte1 = bytesArray[0]
```

- 이진 파일에 바이트를 저장하기

```
outfile = open(filename, "wb")
bytesArray = bytes([255, 128, 0, 1])
outfile.write(bytesArray)
```

Lab 이진 파일 복사하기. p356

- 하나의 이미지 파일을 다른 이미지 파일로 복사하는 프로그램을 작성하여 보자.

```
123.png를 kkk.png로 복사하였습니다.
```

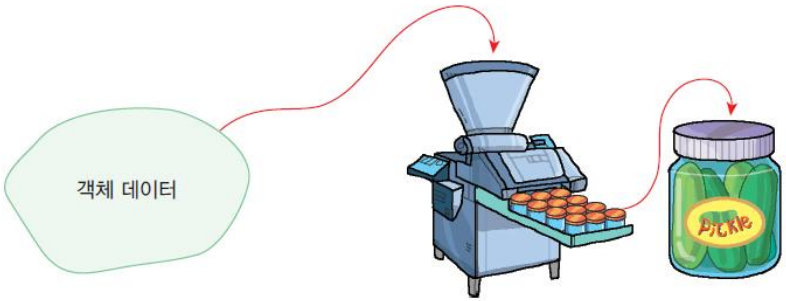
```
infile = open("123.png", "rb")
outfile = open("kkk.png", "wb")

# 입력 파일에서 1024 바이트씩 읽어서 출력 파일에 쓴다.
while True:
    copy_buffer = infile.read(1024)
    if not copy_buffer:
        break
    outfile.write(copy_buffer)

infile.close()
outfile.close()
print(str(infile)+"를 " +str(outfile)+"로 복사하였습니다. ")
```

객체 출력. p357

- 딕셔너리나 리스트와 같은 객체도 파일에 쓸 수 있을까?

```
gameOption = {
                    "Sound": 8,
                    "VideoQuality": "HIGH",
                    "Money": 100000,
                    "WeaponList": ["gun", "missile", "knife" ]
}
```

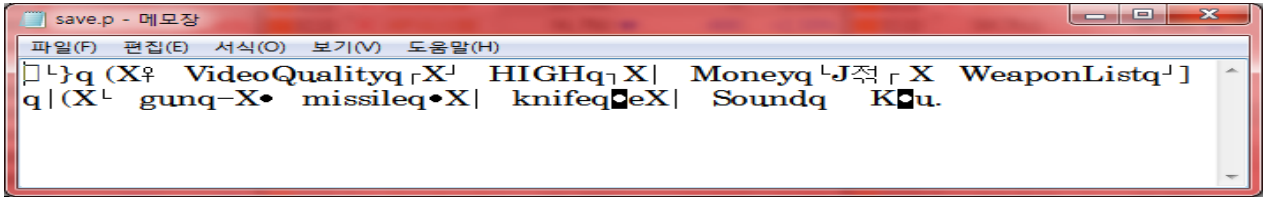
- pickle 모듈의 dump()와 load() 메소드를 사용하면 객체를 쓰고 읽을 수 있다.



Example 딕셔너리 저장하기. p358

- pickle 객체의 dump() 함수를 호출하여서 gameOption 딕셔너리를 전달한다.

```
import pickle

gameOption = {
                        "Sound": 8,
                        "VideoQuality": "HIGH",
                        "Money": 100000,
                        "WeaponList": ["gun", "missile", "knife" ]
}

file = open( "save.p", "wb" )                    # 이진 파일 오픈
pickle.dump( gameOption, file )                  # 딕셔너리를 피클 파일에 저장
file.close()                                     # 파일을 닫는다.
```



Example 딕셔너리 복원하기. p359

- pickle 객체의 load() 함수를 호출하면 파일에 저장된 딕셔너리를 복원할 수 있다.

```
import pickle

file = open( "save.p", "rb" )                # 이진 파일 오픈
obj = pickle.load( open( "save.p", "rb" ) )       # 피클 파일에 딕셔너리를 로딩
print(obj)
```

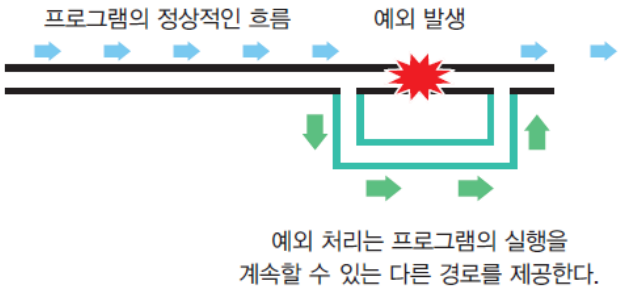
```
{'WeaponList': ['gun', 'missile', 'knife'], 'Money': 100000, 'VideoQuality':
'HIGH', 'Sound': 8}
```

예외처리. p359

- 예외(exception) : 실행 도중에 발생하는 오류

```
>>> (x, y)=(2, 0)
>>> z=x/y
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    z=x/y
ZeroDivisionError: division by zero
```

- 오류가 발생했을 때 오류를 사용자에게 알려주고 모든 데이터를 저장하게 한 후에 사용자가 우아하게(gracefully) 프로그램을 종료할 수 있도록 하는 것이 바람직하다. 또 오류를 처리한 후에 계속 실행할 수 있다면 더 나은 프로그램이 될 수 있다.



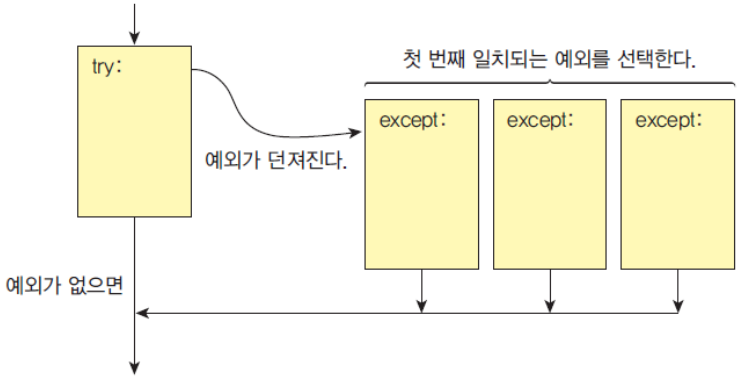
오류의 종류. p361

- 사용자 입력 오류: 사용자가 정수를 입력하여야 하는데 실수를 입력할 수 있다.
- 장치 오류: 네트워크가 안 된다거나 하드 디스크 작동이 실패할 수 있다.
- 코드 오류: 잘못된 인덱스를 사용하여서 배열에 접근할 수 있다.

- IOError: 파일을 열 수 없으면 발생한다.
- importError: 파이썬이 모듈을 찾을 수 없으면 발생한다.
- ValueError: 연산이나 내장 함수에서 인수가 적절치 않은 값을 가지고 있으면 발생한다.
- KeyboardInterrupt: 사용자가 인터럽트 키를 누르면 발생한다. (Control-C나 Delete)
- EOFError: 내장 함수가 파일의 끝을 만나면 발생한다.

오류의 종류. p361

- 오류를 처리하는 방법 : try-except 블록 사용. try 블록에서 발생된 예외를 except 블록에서 처리



예외 처리

형식

```
try:
    예외가 발생할 수 있는 문장
except(오류):
    예외를 처리하는 문장
```

예

```
try:
    z = x/y
except ZeroDivisionError:
    print ("0으로 나누는 예외")
```

예외가 발생할 수 있는 문장

예외

오류의 종류. p362

```
(x,y) = (2,0)
try:
    z = x/y
except ZeroDivisionError:
    print ("0으로 나누는 예외")
```

0으로 나누는 예외

```
(x,y) = (2,0)
try:
    z = x/y
except ZeroDivisionError as e:
    print (e)
```

division by zero

```
>>> n = int(input("숫자를 입력하시오 : "))
숫자를 입력하시오 :  23.5
...
ValueError: invalid literal for int() with base 10: '23.5'
```

오류의 종류. p363

```
while True:
    try:
        n = input("숫자를 입력하시오 :  ")
        n = int(n)
        break
    except ValueError:
        print("정수가 아닙니다. 다시 입력하시오. ")
print("정수 입력이 성공하였습니다!")
```

```
숫자를 입력하시오 : 23.5
정수가 아닙니다. 다시 입력하시오.
숫자를 입력하시오 : 10
정수 입력이 성공하였습니다!
```

```
try:
    fname = input("파일 이름을 입력하세요: ")
    infile = open(fname, "r")
except IOError:
    print("파일 " + fname + "을 발견할 수 없습니다.")
```

```
파일 이름을 입력하세요: kkk.py
파일 kkk.py을 발견할 수 없습니다.
```

try/except 블록에서의 실행 흐름. p363

```
try:
  result = 10 / 20

except ZeroDivisionError:
  print("오류발생")
```

예외가 발생하지 않은 경우

```
try:
  result = 10 / 0

except ZeroDivisionError:
  print("오류발생")
```

예외가 발생하은 경우

Lab 파일 암호화. p365

- 로마의 유명한 정치가였던 쥴리어스 시저(Julius Caesar, 100-44 B.C.)는 친지들에게 비밀리에 편지를 보내고자 할 때 다른 사람들이 알아보지 못하도록 문자들을 다른 문자들로 치환하였다.

평 문	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
암호문	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c

원문: the language of truth is simple.
암호문: wkh odqjxdjh ri wuxwk lv vlpsoh.

Mini Project 파일 압축 . p367

- 파이썬을 사용하면 파일을 ZIP 형식으로 압축하거나 압축해제할 수 있다. zipfile 모듈을 사용하면 개별 또는 여러 파일을 한 번에 추출하거나 압축할 수 있다. 아주 간단하다. 먼저 zipfile 모듈을 가져온 다음 두 번째 매개 변수를 'w'로 지정하여 쓰기 모드에서 ZipFile 객체를 연다. 첫 번째 매개 변수는 파일 경로이다.

```
import zipfile

choice = input("어떤 작업을 하겠습니까?(1-압축 또는 2-해제): ")
if choice == "1":
    fname = input("압축할 파일 이름을 입력하시오: ")
    obj = zipfile.ZipFile('test.zip', 'w')  # 파일 압축
    obj.write(fname)
    obj.close()
elif choice == "2":
    fname = input("압축을 풀 파일 이름을 입력하시오: ")   # 압축 풀기
    obj = zipfile.ZipFile(fname, 'r')
    obj.extract("test.txt")
    obj.close()
```

Mini Project 파일 암호화. p368

연습문제. P370

Programming. P372