

Section01 이 장에서 만들 프로그램

- [프로그램] 파일 암호화 및 암호 해독
  - 파일의 내용을 암호화하거나 암호화된 파일을 다시 해독하는 프로그램



Section02 파일 입출력의 기본

- 파일 입출력의 개념
  - 표준 입출력과 파일 입출력 함수

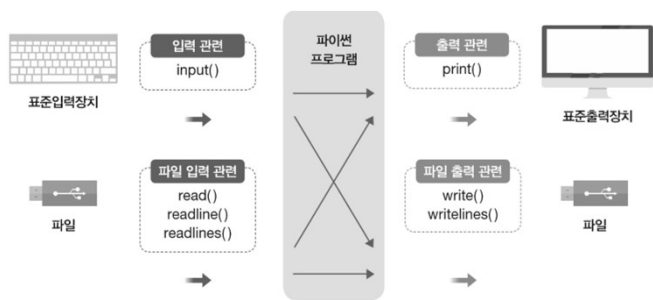


그림 11-1 표준 입출력과 파일 입출력 함수

Section02 파일 입출력의 기본

- 파일 입출력의 기본 과정



그림 11-2 파일 처리의 3단계

- 1단계 : 파일 열기
  - 읽기용 : 변수명 = open("파일명", "r")
  - 쓰기용 : 변수명 = open("파일명", "w")

Section02 파일 입출력의 기본

- 모드( Mode) : open() 함수의 마지막 매개변수

표 11-1 파일의 열기 모드

종류	설명
생략	r과 동일하다.
r	읽기 모드, 기본값이다.
w	쓰기 모드, 기존에 파일이 있으면 덮어쓴다.
r+	읽기/쓰기 겸용 모드이다.
a	쓰기 모드, 기존에 파일이 있으면 이어서 쓴다. append의 약어이다.
t	텍스트 모드, 텍스트 파일을 처리한다. 기본값이다.
b	이진 모드, 이진 파일을 처리한다.

Section02 파일 입출력의 기본

- 모드( Mode ) : open() 함수의 마지막 매개변수

표 11-1 파일의 열기 모드

종류	설명
생략	r과 동일하다.
r	읽기 모드, 기본값이다.
w	쓰기 모드, 기존에 파일이 있으면 덮어쓴다.
r+	읽기/쓰기 겸용 모드이다.
a	쓰기 모드, 기존에 파일이 있으면 이어서 쓴다. append의 약어이다.
t	텍스트 모드, 텍스트 파일을 처리한다. 기본값이다.
b	이진 모드, 이진 파일을 처리한다.

- 2단계 : 파일 처리
- 3단계 : 파일 닫기
  - 1단계에서 open() 함수로 연 변수명

```
변수명.close();
```

5/50

Section03 텍스트 파일 입출력

■ 파일을 이용한 입력



그림 11-3 파일 입력과 표준 출력

6/50

Section03 텍스트 파일 입출력

■ 한 행씩 읽어 들이기

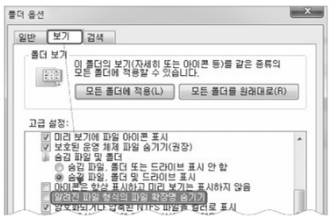
- readline() 함수 사용
- 파일로 데이터 입력 후 이를 화면에 출력하는 프로그램

```
CookBook 파이썬을 공부합니다.  
완전 재미있어요. ^^  
파이썬을 공부하기 잘했네요~~
```

7/50

Section03 텍스트 파일 입출력

Tip • 파일의 확장명 확인 : 윈도우 7에서는 파일 탐색기에서 [구 성]-[폴더 및 검색 옵션] 메뉴를 선택. [폴더 옵션] 대화상자 → [보기] 탭에서 '알려진 파일 형식의 파일 확장명 숨기기' 체크표시를 해제, <확인> 버튼 누름



윈도 8, 8.1, 10에서는 파일 탐색기의 [보기] 메뉴를 선택한 후 '파일 확장명'에 체크



8/50

Section03 텍스트 파일 입출력

Code11-01.py

```
1 inFp = None      # 입력 파일      4행 : 파일 열기
2 inStr = ""       # 읽어 온 문자열
3
4 inFp = open("C:/Temp/data1.txt", "r") : readline() 함수는 inFp로 열린 파일에서 한 행 읽어
5                                       inStr에 저장
6 inStr = inFp.readline()              7행 : 화면에 출력
7 print(inStr, end = "")               이런 처리를 9~10행, 12~13행에서도 반복해 총 3회 수행
8
9 inStr = inFp.readline()
10 print(inStr, end = "")
11
12 inStr = inFp.readline()             15행 : 파일 닫기
13
14 Cookbook 파이썬을 공부합니다.      텍스트 파일의 내용 중 정확히 행 3개만 처리가 가능
15 완전 재미있어요. ^^              텍스트 파일 에 행이 1~2개만 있으면 오류 발생, 4개
    파이썬을 공부하기 잘했네요~~~    이상이 있어도 3개밖에 읽지 못함
```

Section03 텍스트 파일 입출력

- 텍스트 파일에 있는 모든 행을 다 읽기

Code11-02.py

```
1 inFp = None      # 입력 파일
2 inStr = ""       # 읽어 온 문자열
3
4 inFp = open("C:/Temp/data1.txt", "r")
5
6 while True :      6~10행 : 무한 반복
7     inStr = inFp.readline()  7행 : 파일에서 행을 1개 읽음
8     if inStr == "" :        8행에서 읽어 온 것이 없다면 9행 break로 무한 반복에서
9         break;              빠져나옴
10    print(inStr, end = "")    그렇지 않으면 10행에서 읽어 온 내용 출력
11 a
12 inFp.close()
```

Section03 텍스트 파일 입출력

SELF STUDY 11-1

Code11-02.py를 수정해서 다음과 같이 앞에 각 행 번호를 출력해 보자.

**힌트** 1, 2, 3...으로 변경되는 변수를 하나 추가하고, print() 함수에서 "%d %s"를 사용한다.

**출력 결과**

1: Cookbook 파이썬을 공부합니다.  
2: 완전 재미있어요. ^^  
3: 파이썬을 공부하기 잘했네요~~

Section03 텍스트 파일 입출력

- 한 번에 모두 읽어 들이기
  - readlines() 함수 : 파일의 내용을 통째로 읽어서 리스트에 저장


Code11-03.py

```
1 inFp = None
2 inList = ""
3
4 inFp = open("C:/Temp/data1.txt", "r")
5
6 inList = inFp.readlines()  6행 : 한 번에 읽어서 inList에 저장
7 print(inList)
8
9 inFp.close()
```

**출력 결과**

['CookBook 파이썬을 공부합니다.\n', '완전 재미있어요. ^^\\n', '파이썬을 공부하기 잘했네요~~']

Section03 텍스트 파일 입출력

여기서 잠깐 

**with~as 문** 프로그램을 만들다 보면 close() 함수를 호출하지 않고 프로그램을 종료해서 종종 문제가 발생한다. 이에 close() 함수를 사용하지 않으려면 다음과 같이 with~as 문으로 파일을 연다. Code11-03.py에서 4~9행을 다음과 같이 수정하면 된다. 즉 inFp.close()는 사용하지 않는다.

```
with open("C:/Temp/data1.txt", "r") as inFp :
    inList = inFp.readlines()
    print(inList)
```

Section03 텍스트 파일 입출력

- Code11-03.py에서 파일의 내용을 한 행씩 출력

Code11-04.py

```
1 inFp = None
2 inList, inStr = [], ""
3
4 inFp = open("C:/Temp/data1.txt", "r")
5
6 inList = inFp.readlines()
7 for inStr in inList :
8     print(inStr, end = "")
9
10 inFp.close()
```

출력 결과

CookBook 파이썬을 공부합니다.  
완전 재미있어요. ^^  
파이썬을 공부하기 좋았네요~~

Section03 텍스트 파일 입출력

SELF STUDY 11-2

Code11-04.py를 수정해서 다음과 같이 앞에 각 행 번호를 출력해 보자.

출력 결과

- 1: CookBook 파이썬을 공부합니다.
- 2: 완전 재미있어요. ^^
- 3: 파이썬을 공부하기 좋았네요~~

Section03 텍스트 파일 입출력

- 도스 명령어 type의 구현
- 지정한 파일의 내용을 화면에 출력

type 파일명

- 명령 프롬프트창을 열고 다음과 같이 명령어를 입력하면 win.ini 파일의 내용을 화면에 출력



그림 11-4 type 명령어

Section03 텍스트 파일 입출력

출력 프로그램

Code11-05.py

```
1 inFp = None
2 fName, inList, inStr = "", [], ""
3
4 fName = input("파일명을 입력하세요 : ")
5 inFp = open(fName, "r")
6
7 inList = inFp.readlines()
8 for inStr in inList :
9     print(inStr, end = "")
10
11 inFp.close()
```

출력 결과

파일명을 입력하세요 : C:/Windows/win.ini  
; for 16-bit app support  
[fonts]  
... 생략 ...

Section03 텍스트 파일 입출력

- 파일을 열 때 오류 처리
- Code11-05.py를 다시 실행한 후 없는 파일명 입력

출력 결과

파일명을 입력하세요 : C:/abc.txt ← 없는 파일

Traceback (most recent call last):  
File "C:\CookPython\Code11-05.py", line 5, in <module>  
inFp = open(fName, "r")  
FileNotFoundError: [Errno 2] No such file or directory: 'C:/abc.txt'

Section03 텍스트 파일 입출력

파일이 없을 때 오류가 발생하지 않게 하려면 os.path.exists(파일명) 형식 사용

Code11-06.py

```
1 import os
2
3 inFp = None
4 fName, inList, inStr = "", [], ""
5
6 fName = input("파일명을 입력하세요 : ")
7
8 if os.path.exists(fName) :
9     inFp = open(fName, "r")
10
11     inList = inFp.readlines()
12     for inStr in inList :
13         print(inStr, end = "")
14
15     inFp.close()
16 else :
17     print("%s 파일이 없습니다." % fName)
```

출력 결과

파일명을 입력하세요 : C:/abc.txt ← 없는 파일  
C:/abc.txt 파일이 없습니다.

Section03 텍스트 파일 입출력

- 파일을 이용한 출력
- 출력 결과를 파일에 저장하는 방식(파일에 내용 쓸 때 write()나 writelines() 함수 사용)



그림 11-5 표준 입력과 파일 출력

Section03 텍스트 파일 입출력

- 한 행씩 파일에 쓰기
- 키보드에서 입력한 내용을 한 행씩 파일에 쓰는 코드(input() 함수 반복 사용)

```
Code11-07.py
1 outFp = None
2 outStr = ""
3
4 outFp = open("C:/Temp/data2.txt", "w")
5
6 while True :
7     outStr = input("내용 입력 : ")
8     if outStr != "" :
9         outFp.writelines(outStr + "\n")
10    else :
11        break
12
13 outFp.close()
14 print("--- 정상적으로 파일에 씀 ---")
```

4행 : 파일을 열 때 쓰기 모드인 w 사용  
6~11행 : 무한 반복  
7행 : 파일에 쓸 내용을 입력  
8행 : 입력한 글자가 비어 있지 않으면 9행에서 입력한 내용을 파일에 씀, 입력한 글자가 비어 있다면 10~11행에서 무한 반복 종료  
9행 : outStr만 파일에 쓴다면 여러 줄을 써도 줄바꿈 되지 않고 한 줄에 이어서 써짐. 이를 방지하려고 행을 넘기는 \n을 함께 사용  
13행 : 파일 닫기  
14행 : 메시지 출력

출력 결과

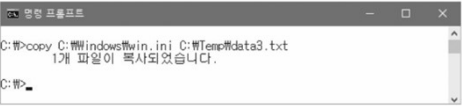
```
내용 입력 : 파이썬을
내용 입력 : 열공하고
내용 입력 : 있습니다. ^^
내용 입력 : (Enter)
--- 정상적으로 파일에 씀 ---
```

Section03 텍스트 파일 입출력

- 도스 copy 명령어의 구현

copy 소스파일 타깃파일

- 명령 프롬프트창을 열고, 다음과 같이 명령어를 입력하면 win.ini 파일이 data3.txt 파일로 복사



C:\>copy C:\Windows\win.ini C:\Temp\data3.txt  
1개 파일이 복사되었습니다.



그림 11-7 copy 명령어를 구현하는 파일 입출력

Section03 텍스트 파일 입출력

```
Code11-08.py
1 inFp, outFp = None, None
2 inStr = ""
3
4 inFp = open("C:/Windows/win.ini", "r")
5 outFp = open("C:/Temp/data3.txt", "w")
6
7 inList = inFp.readlines()
8 for inStr in inList :
9     outFp.writelines(inStr)
10
11 inFp.close()
12 outFp.close()
13 print("--- 파일이 정상적으로 복사되었음 ---")
```

4~5행 : 읽기 모드 및 쓰기 모드로 파일 열기  
7행 : 파일의 내용을 통째로 읽기  
8~9행 : 내용을 쓰기 파일에 쓰기  
11~12행 : 파일 닫기

출력 결과

```
--- 파일이 정상적으로 복사되었음 ---
```

Section03 텍스트 파일 입출력

SELF STUDY 11-4

Code11-08.py를 수정해서 두 파일 모두 파일명을 입력받아 보자.

출력 결과

소스 파일명을 입력하세요 : C:/Windows/win.ini  
타깃 파일명을 입력하세요 : C:/Temp/data4.txt  
--- C:/Windows/win.ini 파일이 C:/Temp/data4.txt 파일로 복사되었음 ---

## Section04 이진 파일 입출력

## ■ 이진 파일의 개념

- 이진(Binary : 바이너리) 파일 : 글자가 아닌 비트(Bit) 단위로 의미가 있는 파일
  - 텍스트 파일을 제외한 나머지 파일
  - 그림 파일, 음악 파일, 동영상 파일, 엑셀 파일, 실행 EXE 파일 등이 모두 이진 파일
- 텍스트 파일과 이진 파일을 구분하는 간단한 방법(파일을 메모장에서 열기)
  - 열었을 때 글자처럼 보이면 텍스트 파일, 이상하게 보이면 이진 파일

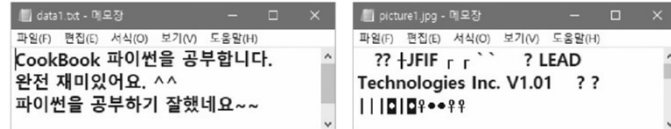


그림 11-9 텍스트 파일과 이진 파일을 메모장에서 열 때

25/50

## Section04 이진 파일 입출력

- 이진 파일의 복사

- Code11-08.py를 수정해서 이진 파일 복사

Code11-10.py

```

1 inFp, outFp = None, None
2 inStr = ""
3
4 inFp = open("C:/Windows/notepad.exe", "rb")
5 outFp = open("C:/Temp/notepad.exe", "wb")
6
7 while True :
8     inStr = inFp.read(1)
9     if not inStr :
10         break
11     outFp.write(inStr)
12
13 inFp.close()
14 outFp.close()
15 print("---- 이진 파일이 정상적으로 복사되었음 ----")

```

### 출력 결과

--- 이진 파일이 정상적으로 복사되었음 ---

26/50

## Section05 파일 입출력의 심화 내용

## ■ 파일 및 디렉터리 다루기

- shutil 모듈과 os 모듈, os.path 모듈 : 파일, 디렉터리(폴더) 다룰 수 있는 다양한 함수 제공

```
import shutil
dir(shutil)
```

## 출력 결과

... 생략 ...

```
'chown', 'collections', 'copy', 'copy2', 'copyfile', 'copyfileobj', 'copymode',
'copystat', 'copytree', 'disk_usage', 'errno', 'fnmatch', 'get_archive_formats',
'get_terminal_size', 'get_unpack_formats', 'getgrnam', 'getpwnam', 'ignore_patterns',
'make_archive', 'move', 'nt', 'os',
```

... 생략 ...

27/50

## Section05 파일 입출력의 심화 내용

```
import os
dir(os)
```

### 출력 결과

... 생략 ...

```
'chdir', 'chmod', 'close', 'closerange', 'cpu_count', 'curdir', 'defpath',
'device_encoding', 'devnull', 'dup', 'dup2', 'environ', 'errno', 'error', 'execl',
'execle', 'execlp', 'execlpe', 'execv', 'execve', 'execvp', 'execvpe', 'extsep', 'fdopen',
'fsdecode', 'fsencode', 'fspath', 'fstat', 'fsync', 'ftruncate', 'get_exec_path',
... 생략 ...
```

28/50

Section05 파일 입출력의 심화 내용

```
import os.path
dir(os.path)
```

```
출력 결과
... 생략 ...
'dirname', 'exists', 'expanduser', 'expandvars', 'extsep', 'genericpath', 'getatime',
'getctime', 'getmtime', 'getsize', 'isabs', 'isdir', 'isfile', 'islink', 'ismount',
'join', 'lexists', 'normcase', 'normpath', 'os', 'pardir', 'pathsep', 'realpath',
'relpath', 'samefile', 'sameopenfile', 'samestat', 'sep', 'split',
... 생략 ...
```

Section05 파일 입출력의 심화 내용

- 파일 및 디렉터리 복사
  - shutil.copy(소스파일, 타겟파일) 함수 사용

```
import shutil
shutil.copy('C:/Windows/notepad.exe', 'C:/Temp/myNote.exe')
```

```
출력 결과
'C:/Temp/myNote.exe'
```

- 복사할 때는 원본 파일이 있어야 하고, 복사하려는 폴더(C:\Temp)도 있어야 함
- 디렉터리 통째로 복사

```
shutil.copytree('C:/CookPython/GIF/', 'C:/Temp/GIF')
```

Section05 파일 입출력의 심화 내용

- 디렉터리의 생성 및 삭제
  - 디렉터리의 생성 : os.mkdir(폴더명) 함수 사용
  - 디렉터리의 삭제 : shutil.rmtree(폴더명) 함수 사용

```
1 import os
2 import shutil
3 os.mkdir('C:/myDir/')
4 os.mkdir('C:/myDir/dir1/')
5 shutil.rmtree('C:/myDir/')
```

```
출력 결과
정상적으로 처리되면 아무런 메시지도 표시하지 않음
```

- os.mkdir(폴더명) 함수 : 미리 상위 폴더 만들어야 함
- 3행을 먼저 실행하지 않고 4행을 실행하면 오류 발생
- shutil.rmtree(폴더명) 함수는 폴더 안의 모든 파일 삭제

Section05 파일 입출력의 심화 내용

- 디렉터리의 목록 모두 보기
  - os.walk(폴더) 함수 사용

```
1 import os
2 for dirName, subDirList, fnames in os.walk('C:\\Windows\\debug') :
3     for fname in fnames :
4         os.path.join(dirName, fname)
```

```
출력 결과
'C:\\Windows\\debug\\ESE.TXT'
'C:\\Windows\\debug\\lrmr.log'
'C:\\Windows\\debug\\netlogon.log'
'C:\\Windows\\debug\\NetSetup.LOG'
'C:\\Windows\\debug\\PASSWD.LOG'
'C:\\Windows\\debug\\WIA\\wiatrace.bak.log'
'C:\\Windows\\debug\\WIA\\wiatrace.log'
```



Section05 파일 입출력의 심화 내용

- 파일 또는 폴더가 이미 존재하는지 확인
  - os.path.exists(파일명 또는 폴더명) 사용

```
import os.path
os.path.exists('C:/Windows/notepad.exe')
os.path.isfile('C:/Windows/notepad.exe')
os.path.isdir('C:/Windows')
```

출력 결과

모두 True

Section05 파일 입출력의 심화 내용

- 파일 삭제

```
import os
os.remove('C:/Temp/myNote.exe')
os.remove('C:/Temp/myNote.exe')
```

출력 결과

FileNotFoundError: [WinError 2] 지정된 파일을 찾을 수 없습니다: 'C:/Temp/myNote.exe'

- 파일 크기 확인(바이트 단위로 출력)

```
import os.path
os.path.getsize('C:/CookPython/GIF/dog.gif')
```

출력 결과

22332

Section05 파일 입출력의 심화 내용

- 파일 압축과 압축 풀기
  - 압축 기능은 zipfile 모듈에서 제공

```
1 import zipfile
2 newZip = zipfile.ZipFile('C:/Temp/new.zip', 'w')
3 newZip.write('C:/Windows/notepad.exe', compress_type = zipfile.ZIP_DEFLATED)
4 newZip.close()
```

2행 : 새로 압축될 파일을 쓰기(w)로 준비  
3행 : 압축 수행  
4행 : 압축 파일 닫기

출력 결과

정상적으로 처리되면 아무런 메시지도 표시하지 않음

Section05 파일 입출력의 심화 내용

- 파일 압축과 압축 풀기
  - 압축 기능은 zipfile 모듈에서 제공

```
1 import zipfile
2 newZip = zipfile.ZipFile('C:/Temp/new.zip', 'w')
3 newZip.write('C:/Windows/notepad.exe', compress_type = zipfile.ZIP_DEFLATED)
4 newZip.close()
```

2행 : 새로 압축될 파일을 쓰기(w)로 준비  
3행 : 압축 수행  
4행 : 압축 파일 닫기

출력 결과

정상적으로 처리되면 아무런 메시지도 표시하지 않음

- 압축 풀기

```
1 extZip = zipfile.ZipFile('C:/Temp/new.zip', 'r')
2 extZip.extractall('C:/Temp/')
3 extZip.close()
```

1행 : 압축 풀 파일을 읽기(r)로 준비  
2행 : 해당 폴더에 압축 풀기  
3행 : 압축 파일 닫기

출력 결과

정상적으로 처리되면 아무런 메시지도 표시하지 않음

Section06 예외처리

예외의 개념과 사례

- 예외(exception) 란 프로그램을 개발하면서 예상하지 못한 상황이 발생한 것이다. 프로그래밍의 예외는 크게 예측 가능한 예외와 예측 불가능한 예외로 나눌 수 있다.



(a) 아이디 생성 오류 입력

(b) 자동 저장 기능

[예외에 대비한 사례]

Section06 예외처리

예측 가능한 예외와 예측 불가능한 예외

- 예측 가능한 예외** : 발생 여부를 개발자가 사전에 인지할 수 있는 예외이다. 개발자는 예외를 예측하여 명시적으로 예외가 발생할 때는 어떻게 대응하라고 할 수 있다. 대표적으로 사용자 입력란에 값이 잘못 들어갔다면, if문을 사용하여 사용자에게 잘못 입력하였다고 응답하는 방법이 있다. 매우 쉽게 대응할 수 있다
- 예측 불가능한 예외** : 대표적으로 매우 많은 파일을 처리할 때 문제가 발생할 수 있다. 예측 불가능한 예외가 발생했을 경우, 인터프리터가 자동으로 이것이 예외라고 사용자에게 알려준다. 대부분은 이러한 예외가 발생하면서 프로그램이 종료되므로 적절한 조치가 필요하다.

Section06 예외처리

예외 처리 구문 : try -except문

- 파이썬 예외 처리의 기본 문법은 try -except문이다.

```
try:
    예외 발생 가능 코드
except 예외 타입:
    예외 발생 시 실행되는 코드
```

Section06 예외처리

예외 처리 구문 : try-except-else문

- try-except-else문은 if-else문과 비슷한데, 해당 예외가 발생하지 않을 경우 수행할 코드를 else문에 작성하면 된다.

```
try:
    예외 발생 가능 코드
except 예외 타입:
    예외 발생 시 실행되는 코드
else:
    예외가 발생하지 않을 때 실행되는 코드
```

Section06 예외처리

예외 처리 구문 : try-except-finally문

- try-except-finally문에서 finally문은 try-except문 안에 있는 수행 코드가 아무런 문제 없이 종료되었을 경우, 최종으로 호출하는 코드이다

```
try:
    예외 발생 가능 코드
except 예외 타입:
    예외 발생 시 실행되는 코드
finally:
    예외 발생 여부와 상관없이 실행되는 코드
```

Section06 예외처리

예외 처리

- try, except 문
- 예외 처리(Exception Handling) : 오류가 발생할 때 파이썬이 처리하지 않고 프로그래머가 작성 한 코드를 실행하는 방식

```
import os
os.remove('C:/Temp/noFile.exe') # 이 파일이 없음
```

출력 결과

```
FileNotFoundError: [WinError 2] 지정된 파일을 찾을 수 없습니다: 'C:/Temp/noFile.exe'
```

- try 문, except 문을 활용, 이런 파이썬의 오류 대신 직접 오류 메시지를 작성 정상적으로 흐름을 이어가게 함

```
try :
    os.remove('C:/Temp/noFile.exe')
except :
    print('파일이 없네요. 확인 바랍니다.')
```

출력 결과

```
파일이 없네요. 확인 바랍니다.
```

Section06 예외처리

- 문자열 중에서 '파이썬' 글자의 위치를 모두 찾아서 출력하는 코드
- '파이썬' 글자가 위치한 첨자 0과 13을 찾은 후에는 더 이상 없기 때문에 오류가 발생

Code11-13.py

```
1 myStr = '파이썬은 재미 있어요. 파이썬만 매일매일 공부하고 싶어요. ^^'
2 strPosList = []
3 index = 0
4
5 while True :
6     index = myStr.index('파이썬', index)
7     strPosList.append(index)
8     index = index + 1 # 다음 위치부터 찾을
9
10 print('파이썬 글자 위치 ->', strPosList)
```

출력 결과

```
Traceback (most recent call last):
  File "C:\CookPython\Code11-13.py", line 6, in <module>
    index = myStr.index('파이썬', index)
ValueError: substring not found
```

Section06 예외처리

- 예외 처리 추가

Code11-14.py

```
1 myStr = '파이썬은 재미 있어요. 파이썬만 매일매일 공부하고 싶어요. ^^'
2 strPosList = []
3 index = 0
4
5 while True :
6     try :
7         index = myStr.index('파이썬', index)
8         strPosList.append(index)
9         index = index + 1 # 다음 위치부터 찾을
10    except :
11        break
12
13 print('파이썬 글자 위치 ->', strPosList)
```

출력 결과

```
파이썬 글자 위치 -> [0, 13]
```

Section06 예외처리

- except 문 뒤에 아무것도 명시하지 않으면 모든 종류의 오류 처리
  - 필요하다면 오류의 종류에 따라서 다른 오류 처리도 가능

```
try :
    실행할 문장들
except 예외_종류 1 :
    오류일 때 실행할 문장들
except 예외_종류 2 :
    오류일 때 실행할 문장들
```

Section06 예외처리

- 대표적인 예외종류

표 11-2 대표적인 예외 종류

종류	설명
ImportError	import 문에서 오류가 발생할 때
IndexError	리스트 등 첨자의 범위를 벗어날 때
KeyError	딕셔너리에서 키가 없을 때
KeyboardInterrupt	프로그램 실행 중 <b>Ctrl+C</b> 를 누를 때
NameError	변수명이 없는 것에 접근할 때
RecursionError	재귀 호출의 횟수가 시스템에서 설정한 것보다 넘칠 때(1000번)
RuntimeError	실행이 발생할 때
SyntaxError	exec()나 eval()에서 문법상 오류가 발생할 때
TypeError	변수형의 오류가 발생할 때. 예 '문자열-문자열' 연산
ValueError	함수의 매개변수에 잘못된 값을 넘길 때. 예 int('파이썬')
ZeroDivisionError	0으로 나눌 때
IOError	파일 처리 등 오류일 때

Section06 예외처리

- 오류의 종류에 따라서 다른 처리를 진행하는 코드

Code11-15.py

```
1 num1 = input('숫자1 -->')
2 num2 = input('숫자2 -->')
3
4 try :
5     num1 = int(num1)
6     num2 = int(num2)
7     while True :
8         res = num1 / num2
9
10 except ValueError :
11     print('문자열은 숫자로 변환할 수 없습니다.')
12
13 except ZeroDivisionError :
14     print('0으로 나눌 수 없습니다.')
15
16 except KeyboardInterrupt :
17     print('Ctrl+C를 눌렀군요.')
```

Section06 예외처리

출력 결과

숫자1 -->이건 뭐죠?  
숫자2 -->10  
문자열은 숫자로 변환할 수 없습니다.  
숫자1 -->10  
숫자2 -->0  
0으로 나눌 수 없습니다.  
숫자1 -->10  
숫자2 -->5  
Ctrl+C를 눌렀군요. ← 무한 반복에서 **Ctrl+C**를 눌렀을 때

Section06 예외처리

- try, except, else, finally 문
  - try 문에서 오류가 발생하면 except 문이 실행
  - 오류가 발생하지 않으면 else 문이 실행
  - finally 문은 오류가 발생하든 그렇지 않은 무조건 실행

Code11-16.py

```
1 num1 = input('숫자1 -->')
2 num2 = input('숫자2 -->')
3
4 try :
5     num1 = int(num1)
6     num2 = int(num2)
7
8 except :
9     print('오류가 발생했습니다.')
10
11 else :
12     print(num1, '/', num2, '=', num1 / num2)
13
14 finally :
15     print('이 부분은 무조건 나옵니다.')
```

Section06 예외처리

출력 결과

숫자1 -->100  
숫자2 -->50  
100 / 50 = 2.0  
이 부분은 무조건 나옵니다.  
숫자1 -->100  
숫자2 -->모름  
오류가 발생했습니다.  
이 부분은 무조건 나옵니다.