모두를 위한 파이썬 프로그래밍

12주 프로그램의 완성도 높이기

(유용한 함수들, 문서 읽고 단어분석, 예외상황처리)

학습내용

- 리스트를 이용한 수학함수 풀기
- 문서를 읽어서 단어 통계 알아보기
- 예상치 못한 상황을 대비한 예외처리하기
  - 읽어야 할 파일이 없는 경우
  - 웹사이트 접속오류가 발생한 경우

유용한 함수들

1

문자열 식을 계산하는(eval)함수

■ 문자열로 된 수식을 계산할 수 있는 함수

- 숫자와 연산자로 구성된 문자열, 함수와 숫자로 구성된 문자열을 계산가능 하도록 처리

```
1  from math import sqrt
2  print('2+3')     #  2+3
3  print(2 + 3)     # 5
4  print(eval('2+3'))  # 5
5  print('sqrt(2)')     # sqrt(2)
6  print(eval('sqrt(2)'))  # 1.4142135623730951
```

계산기 프로그램을 작성할 때 +, -, *, / 등의 연산자는 문자열로 처리됨
→ eval 함수 적용

람다 함수(lambda) 만들기

함수이름없이 한 줄로 간단하게 함수를 만드는 방법

- 두 수의 합을 구하는 함수와 람다함수 비교

```
1  def my_add(x, y):
2      return x + y
3  # 2개의 인자를 받아 더한 결과를 리턴 하는 함수
4
5  my_sum = my_add(2, 5)
6  print(my_sum)
```

```
1  my_sum = lambda x, y: x + y
2  print(my_sum(2, 5))
```

lambda 인자1, 인자2... : 실행코드

간단한 함수는 람다함수로 활용하는 것이 편리함

람다 함수(lambda) 만들기

tkinter 버튼 함수에서 인자 전달 함수로 활용 가능

- Button 함수에서 버튼을 눌렀을 때 실행되는 함수는
  command=함수이름 으로 받게 되는 데, 이때 함수이름에 인자 값을 사용할 수 없음
- 인자를 전달하는 다른 수단으로 람다 함수를 활용

```
1    def press(x):
2        print(x)
3
4
5    commandBtn = lambda: press(2)
6    commandBtn()
```

lambda 인자1, 인자2... : 실행코드

↓

lambda : 실행코드

인자 값을 전달하지 않게 생략하여 작성함 → 그러나 실행코드를 통해 실제로는 인자 값을 전달가능

데이터 분리하기(split)

저장된 데이터를 구분자로 분리하기

```
date = '1985.02.15'
new_date = date.split('.')
print('year: ', new_date[0])
print('month: ', new_date[1])
print('day : ', new_date[2])
```

- 한꺼번에 여러 개의 데이터를 입력 받아 분리하여 저장하기

```
scores = input('3과목의 점수를 입력 하시오 > ').split('/')
print(scores)
```

여러 개의 과목을 한번에 입력 받고 리스트에 저장

```
3과목의 점수를 입력 하시오 > 88/87/98
['88', '87', '98']
```

데이터의 개수, 길이를 알아내기(len)

- 리스트 또는 문자열 변수의 데이터 개수(또는 길이) 확인하는 함수

```
date = '1985.02.15'
new_date = date.split('.')
print('yeqr: ', new_date[0])
print('month: ', new_date[1])
print('day :', new_date[2])
print(len(date))    # 문자열 변수의 데이터 길이
print(len(new_date))    # 리스트내에 데이터 개수
print(len(new_date[0])) # new_date[0] 의 데이터 길이
```

함수를 반복해서 호출하기(map)

■ 함수에 인자를 자동으로 바꾸어 반복적으로 함수를 호출

- 문자열로 구성된 리스트를 int형으로 변환하기

```
original_list = ['2', '4', '6', '8', '10', '12']
new_int_list = list(map(int, original_list))
print(new_int_list)
# 출력 결과: [2, 4, 6, 8, 10, 12]
```

- 한꺼번에 여러 개의 데이터를 입력 받아 정수형으로 저장하는 예 #1

```
scores = input('3과목의 점수를 입력 하시오 > ').split()
print(scores)
print(list(map(int, scores)))
```

여러 개의 과목을 입력 받고 정수형으로 변환 후 리스트에 저장

```
3과목의 점수를 입력 하시오 > 88 87 98
['88', '87', '98']
[88, 87, 98]
```

함수를 반복해서 호출하기(map)

- 한꺼번에 여러 개의 데이터를 입력 받아 정수형으로 저장하는 코딩 #2

```
1  scores = list(map(int, input('3과목 점수를 입력하시오 ').split()))
2  print(scores)
```

여러 개의 과목을 입력 받고 정수형으로 변환 후 리스트에 저장

- 리스트의 값을 모두 곱하는 함수를 계속 호출하는 예

  집합 A가 있을 때, 함수 $f(x) = x^2$ 에 대해 집합 A의 모든 값은 어떤 값으로 대응되는지 그 결과를 구하시오

```
1  f = lambda x: x * x
2
3  A = [1, 2, 3, 4, 5]
4  B = list(map(f, A))
5  print(B)  # [1, 4, 9, 16, 25]
```

```
def f(x):
    return x * x
```

함수를 반복해서 호출하기(map)

■ 두 개 이상의 반복되는 인자에 적용 가능

- f(x, y) = x^2 + Y의 결과를 구하는 프로그램을 작성 하시오

$$f(x, y) = x^2 + Y$$

```
1  f = lambda x, y: x * x + y
2
3  X = [1, 2, 3, 4, 5]
4  Y = [10, 9, 8, 7, 6]
5  result = list(map(f, X, Y)) # 각각의 X, Y값에 대한 함수 f 수행
6  print(result)  # [11, 13, 17, 23, 31]
```

파일처리

2

파일 다루기

파일의 종류

- 컴퓨터에서 파일의 종류는 다양하지만, 기본적으로 바이너리 파일(binary file)과 텍스트 파일(text file), 두 가지로 분류됨
- 메모장으로 파일을 열어보면, 간단히 구분할 수 있음

바이너리 파일	텍스트 파일
• 컴퓨터만 이해할 수 있는 형태인 이진(법) 형식으로 저장된 파일 • 일반적으로 메모장으로 열면 내용이 깨져 보임(메모장에서 해석 불가) • 엑셀 파일, 워드 파일 등	• 사람도 이해할 수 있는 형태인 문자열 형식으로 저장된 파일 • 메모장으로 열면 내용 확인이 가능 • 메모장에 저장된 파일, HTML 파일, 파이썬 코드 파일 등

파일 다루기

■ 파일 읽기

- 책을 읽기 위해서는 펼쳐야 함, 공책에 쓰기 위해서도 펼쳐야 함, 공책을 수정하기 위해서도 펼쳐야 함

→ 텍스트 파일을 다루기(읽고/쓰기/추가) 위해 open( ) 함수를 사용

```
f = open("파일명", "파일 열기 모드")
f.close()
```

종류	설명
r	읽기 모드: 파일을 읽기만 할 때 사용
w	쓰기 모드: 파일에 내용을 쓸 때 사용
a	추가 모드: 파일의 마지막에 새로운 내용을 추가할 때 사용

파일 다루기

파일 읽기

- 불러오기 예.

```
f = open('alice.txt')
book = f.read()
print(book)
f.close()
```

출처: http://www.umich.edu/~umfandsf/other/ebooks/alice30.txt

ALICE'S ADVENTURES IN WONDERLAND
Lewis Carroll
THE MILLENNIUM FULCRUM EDITION 3.0
CHAPTER I Down the Rabbit-Hole
Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do: once or twice she had peeped into the book her sister was reading, but it had no pictures or conversations in it, `and what is the use of a book,' thought Alice `without pictures or conversation?'

파일 다루기

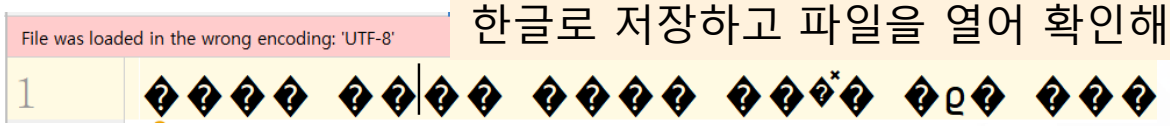
파일 쓰기

- 사용자로부터 입력 받은 텍스트를 파일로 저장하려면 파일을 텍스트 쓰기 모드로 오픈하고 파일 객체의 write()함수를 이용해 데이터를 파일에 기록함
- 사용자로부터 텍스트 문장을 입력 받고 memo.txt 파일로 저장하는 예제

```
memo = input('저장한 내용을 입력해 주세요 > ')
f = open('memo.txt', 'w')
f.write(memo)
f.close()
```

helloworld

영어로 저장하고 파일을 열어 확인해 보자



한글로 저장하고 파일을 열어 확인해 보자

파일 다루기

한글 파일 쓰기

- 문자를 저장하기 위해서는 텍스트 파일을 저장할 때 사용하는 표준을 지정해야 하는데, 이것을 인코딩(encoding) 형식이라고 함

```
memo = input('저장한 내용을 입력해 주세요 > ')
f = open('memo.txt', 'w', encoding='utf8')
f.write(memo)
f.close()
```

안녕하세요 hello

파일 다루기

파일에 글자 추가 하기

- 파일 열기 모드 a로 새로운 글 추가하기

```
memo = input('저장한 내용을 입력해 주세요 > ') + '\n'
f = open('memo.txt', 'a', encoding='utf8')
f.write(memo)
f.close()
```

- 옵션 'a'로 지정하면 기존 파일에 글자가 append됨
- 파일이 존재하지 않으면 새로 생성됨
- ₩n은 다음줄에 글자가 추가되도록 강제 개행 처리

```
helloworld
안녕하세요
오늘은 날씨 맑음 파인썬하기 좋은 날
```

실습

텍스트 파일을 읽어 들이고 단어 통계 구하기

- 글자 수, 단어 수, 라인 수를 구하는 함수
  split( ) 함수와 len( ) 함수 활용

```
f = open('alice.txt', 'r', encoding='utf8')
book = f.read()
word_list = book.split(' ')
line_list = book.split('\n')
f.close()
```

빈칸을 기준으로 단어를 분리하여 리스트에 저장

한 줄 단위로 분리하여 리스트에 저장

```
print('총 글자:', len(book))  # character count
print('총 단어:', len(word_list))  # word count
print('총 라인:', len(line_list))  # line count
```

총 글자: 148544
총 단어: 28919
총 라인: 3599

참고 – 단어 빈도수 구하기

collections 모듈을 활용한 텍스트 마이닝

- 좀 더 정확한 단어분석이 가능하다.
- 그러나 한글 분석은 형태소 분석이 되지 않는다.
- 워드 클라우드 기능을 활용하는 것을 권장

```
from collections import defaultdict
from collections import OrderedDict
word_count = defaultdict(lambda :0)

f = open( 'alice.txt', 'r', encoding='utf8'))
book = f.read().lower().split()

for word in book:
    word_count[word] += 1

for i, v in
OrderedDict(sorted(word_count.items(),key = lambda
t:t[1], reverse=True)).items():
    print(i, v)
f.close()
```

```
the 1603
and 766
to 706
a 614
she 518
of 493
said 421
it 362
in 351
was 333
you 265
i 261
as 249
that 222
alice 221
her 208
at 206
had 176
with 169
all 155
on 142
very 139
be 138
for 135
so 126
`i 121
little 120
they 118
but 117
he 111
```

예외처리

3

생각해보기

■ 에러와 예외의 차이점

✓에러처리
- 프로그램 개발 시 발생하는 에러 수정

✓예외처리
- 프로그램이 개발된 후 발생할 수 있는 상황에 대비

✓앱을 실행할 때 일어나는 다양한 예외상황들

1. 기상청 웹 크롤링 분석 프로그램을 작성하여 기상정보를 제공하는 앱을 개발함

--> 기상청 홈페이지가 일시 점검 중일 때 앱 실행에 예외 상태 발생

2. PDF과 TXT파일을 읽어 편리하게 읽을 수 있는 전자책 리더 앱을 개발함

--> 파일이 지워진 경우에 앱 실행에 예외상황 발생

3. 계산할 수 없는 산술연산을 시도하고자 하는 경우

--> 0으로 나누고자 하는 상황 등

예외 처리

예측을 통한 예외처리

- 입력 예외 처리: 사용자의 미숙한 입력을 사전에 인지하고 알려주도록 함.
  - 잘못된 값이 입력되었을 때, if문을 사용하여 잘못 입력하였다고 안내

- 외부 입력(파일, 값, 주소 등)의 부재로 발생하는 예외
  - 프로그램이 동작하기 위해 필요한 외부입력이 실행 시점에서 없어진 경우
  - 파일이 없는 경우(미설치, 삭제 등), 사이트가 안 열리는 경우, 잘못된 연산자를 사용한 경우, 예외처리로 안내

예외 처리 구문 : try ~ except문

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

예외 처리

■ 예외 처리 구문 : try ~ except문

```
1  for i in range(10):
2      try:
3          print(10 / i)
4      except ZeroDivisionError:
5          print('0으로 나눌 수 없습니다')
```

에러 타입은 생략해도 된다

```
0으로 나눌 수 없습니다
10.0
5.0
3.3333333333333335
2.5
2.0
1.6666666666666667
1.4285714285714286
1.25
1.1111111111111112
```

예외의 종류와 예외 에러 메시지

■ 예외의 종류

예외	내용
IndexError	리스트의 인덱스 범위를 넘어갈 때
NameError	존재하지 않는 변수를 호출할 때
ZeroDivisionError	0으로 숫자를 나눌 때
ValueError	변환할 수 없는 문자나 숫자를 변환할 때
FileNotFoundError	존재하지 않는 파일을 호출할 때

```
C:₩python₩untitled11₩Scripts₩python.exe C:/python/untitled11/fo
Traceback (most recent call last):
  File "C:/python/untitled11/forecast.py", line 3, in <module>
    print(10 / i)
ZeroDivisionError: division by zero
```

실행을 통해 사전에 오류를 확인해 볼 수 있다.

예외의 종류와 예외 에러 메시지

■ 예외 처리 구문 : try-except-else문

- 10을 i로 나누는 코드를 실행하여 제대로 나누었을 경우 else문에 의해 결과가 화면에 출력되고, 그렇지 않을 경우 사전에 정의된 except문이 수행됨

```
1  for i in range(5):
2      try:
3          result = 10/i
4      except:
5          pass
6      else:
7          print(10/i)
```

실행할 코드가 없을 때 pass처리하면 통과처리 됨

```
10.0
5.0
3.3333333333333335
2.5
```

Thank you