세종사이버대학교

파이썬기초

4주차 - 2교시

람다함수와 예외처리

학습내용

- 람다함수
- 예외처리

학습목표

- 익명함수인 람다함수를 정의하는 방법을 알아보고 람다함수를 정의할 수 있다.
- 예외에 대한 의미를 알아보고 예외처리하는 구문으로 예외처리를 할 수 있다.

생각해 봅시다

함수를 간단히 정의하는 방법과
오류가 발생이 예상된다면
어떻게 오류를 대처할지에 대해
생각해봅시다.

01

람다함수

1 | 람다함수

01 정의

람다함수

함수를 한 줄로 간단하게 만들어 주는 함수로서,
매개변수로 함수를 전달할 때 함수를 간단하고
쉽게 선언하는 방법임

→ 1회용 함수를 만들어야 할 때 많이 사용함

1 | 람다함수

02 함수의 매개변수로 함수 전달하기

filter( )함수 map( )함수

“ 함수를 매개변수로 전달하는 대표적인 표준함수 ”

1 | 람다함수

03 map( ) 함수

- 전달된 리스트의 요소를 함수에 넣고 리턴된 값으로 새로운 리스트를 생성함

```
map(함수, 리스트)
```

1 | 람다함수

03 map( ) 함수

- 사용 예제

In [48] :
```
# map 함수에 매개변수로 전달할 함수
def power(num) :
    return num * num;
```

In [49] :
```
# map 함수에 전달할 리스트
list_num = [1, 2, 3, 4]
```

1 | 람다함수

03 map( ) 함수

- 사용 예제

In [50] :
```
result_map = map(power, list_num)
result_map
```

Out [50] : <map at 0x208c15e3010>

In [51] :
```
list(result_map)
```

Out [51] : [1, 4, 9, 16]

1 | 람다함수

04 filter( ) 함수

- 전달된 리스트의 요소를 함수에 넣고 리턴된 값이 Ture인 요소로 새로운 리스트를 생성함

```
filter(함수, 리스트)
```

1 | 람다함수

04 filter( ) 함수

- 사용 예제

In [52] :
```
# filter 함수에 전달할 리스트
list_num = [1, 2, 3, 4, 5, 6, 7]
```

In [53] :
```
# filter 함수에 매개변수로 전달할 함수
def underCheck(num) :
    return num < 5;
```

In [54] :
```
result_filter = filter(underCheck, list_num)
list(result_filter)
```

Out [54] :
```
[1, 2, 3, 4]
```

1 | 람다함수

05 람다의 정의

- lambda 매개변수 : 리턴값

1 | 람다함수

05 람다의 정의

- 람다 정의 예제

In [52] :
```
# 람다 함수 정의
lambda_power = lambda num : num*num
lambda_underCheck = lambda num : num < 5
```

In [50] :
```
lambda_power(10)
```

Out [50] : 100

In [51] :
```
lambda_underCheck(3)
```

Out [51] : True

1 | 람다함수

06 람다함수를 이용한 map, filter 함수 적용

In [52] :

```
list_num = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# map, filter 함수를 이용해서 새로운 리스트를 생성
result_filter = filter(lambda num : num < 5, list_num)

result_map = map(lambda num : num*num, list_num)
```

1 | 람다함수

06 람다함수를 이용한 map, filter 함수 적용

In [52] :
```
# 새로운 리스트 결과 확인
print('result_map', list(result_map))
print('result_filter', list(result_filter))
```

```
result_map [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
result_filter [1, 2, 3, 4]
```

02

예외처리

1 | 예외처리

01 예외

예외(Exception)

프로그램을 개발하면서 예상하지 못한 상황이 발생하는 것

예 사용자의 입력 오류, 구문 오류 등

1 | 예외처리

01 예외

예외처리

예외 상황을 예측하고 예외 발생시 비정상적인 종료를
하지 않고 프로그램이 정상적으로 실행되도록 하는 것

1 | 예외처리

02 예외의 구분

예측 가능한 예외
발생 여부를 개발자가 사전에 인지할 수 있는 예외

→ 개발자는 예외를 예측하여 예외가 발생할 때
처리해야 할 것을 정의할 수 있음

1 | 예외처리

02 예외의 구분

예측 불가능한 예외
발생 여부를 개발자가 사전에 인지할 수 없는 예외

예 매우 많은 파일을 처리할 때 문제가 발생하는 것

→ 예측 불가능한 예외 발생시 인터프리터가
자동으로 사용자에게 알려줌

예외처리
제품의 완성도를 높이는 차원에서 매우 중요함

1 | 예외처리

03 예외가 발생하는 예제

In [1] :
```
for i in range(5):
print(10 / i)
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[1], line 2
      1 for i in range(5):
----> 2     print(10 / i)

ZeroDivisionError: division by zero
```

1 | 예외처리

03 예외가 발생하는 예제

```
In [3] : print(“abc)
```

```
  Cell In[3], line 1
    print("abc)
          ^
SyntaxError: unterminated string literal (detected at line 1)
```

2 | 예외처리 구문

01 try-except문

try-except문

파이썬에서 예외처리의 기본 문법

try문에 예외 발생이 예상되는 코드를 적음 → except문에 예외 발생 시 대응하는 코드를 작성

2 | 예외처리 구문

01 try-except문

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

2 | 예외처리 구문

01 try-except문

- 예외처리 예제

In [1] :
```
for i in range(5):
    try:
        print(10 / i)
    except ZeroDivisionError:
        print('0으로 나누는 연산을 할 수 없습니다.')
```

2 | 예외처리 구문

01 try-except문

- 예외처리 예제

결과

```
0으로 나누는 연산을 할 수 없습니다.
10.0
5.0
3.3333333333333335
2.5
```

2 | 예외처리 구문

01 try-except문

- 자주 사용되는 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

2 | 예외처리 구문

01 try-except문

- 예외 타입의 별칭 사용

In [7] :
```
for i in range(5):
    try:
        print(10 / i)
    except ZeroDivisionError as e:
        print(‘에러 메시지 : “, e)
        print('0으로 나누는 연산을 할 수 없습니다.’)
```

2 | 예외처리 구문

01 try-except문

- 예외 타입의 별칭 사용

결과

```
에러 메시지 : division by zero
0으로 나누는 연산을 할 수 없습니다.
10.0
5.0
3.3333333333333335
2.5
```

2 | 예외처리 구문

02 try-except-finally문

try-except-finally문

finally 블록은 try-except문 안에 있는 코드의
예외 발생 여부와 상관 없이 무조건 처리되는 블록

2 | 예외처리 구문

02 try-except-finally문

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

2 | 예외처리 구문

02 try-except-finally문

- 예외처리 예제

In [7] :
```
for i in range(5):
    try:
        print(10 / i)
    except ZeroDivisionError:
        print('0으로 나누는 연산을 할 수 없습니다.')
    finally:
        print(">>> ",i+1,"번째 연산 실행")
```

2 | 예외처리 구문

02 try-except-finally문

- 예외처리 예제

결과

```
0으로 나누는 연산을 할 수 없습니다.
>>> 1 번째 연산 실행
10.0
>>> 2 번째 연산 실행
5.0
>>> 3 번째 연산 실행
3.3333333333333335
>>> 4 번째 연산 실행
2.5
>>> 5 번째 연산 실행
```

3 | 사용자가 발생시키는 예외

01 raise문

raise문

프로그램 내부에서 필요할 때 예외를 발생시키는 키워드

- raise 예외 타입(예외 타입)

3 | 사용자가 발생시키는 예외

01 raise문

- 사용 예제

In [7] :
```
while True:
    value = input("숫자를 입력하세요 >>> ")
    for digit in value:
        if digit not in "0123456789":
            raise ValueError("숫자값을 입력하지 않았습니다.")
    print("입력한 숫자 : ", int(value))
```

```
숫자를 입력하세요 >>> 10
입력한 숫자 : 10
숫자를 입력하세요 >>> aa
```

3 | 사용자가 발생시키는 예외

01 raise문

- 사용 예제

결과

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[11], line 5
      3 for digit in value:
      4     if digit not in "0123456789":
----> 5         raise ValueError("숫자값을 입력하지 않았습니다.")
      6 print("입력한 숫자 : ", int(value))

ValueError: 숫자값을 입력하지 않았습니다.
```

세종사이버대학교

파이썬기초

NEXT

내장함수와 외장함수