python부터 딥러닝까지

Python 중급반

주제 | 파일 입출력~Tkinter

Contents

Next

예외처리 (이어서)

예외처리가 필요한 이유

- 사용자들은 우리가 의도한 대로 프로그램을 사용하지 않는다.

  예시)

```
첫번째 숫자를 입력해주세요.(숫자로 입력해주세요)>>2
두번째 숫자를 입력해주세요.(숫자로 입력해주세요)>>싫은뎅 ㅎㅎ
Traceback (most recent call last):
  File "D:/My_file/Github_asdfrv20/Python_deep/mission_answer/11th_week_answer.py", line 193, in <module>
    num2 = float(input("두번째 숫자를 입력해주세요.(숫자로 입력해주세요)>>"))
ValueError: could not convert string to float: '싫은뎅 ㅎㅎ'
```

- 예외 처리는 “프로그램 실행 중에 발생하는” Error(예외)를 미연에 방지하여 프로그램이 강제로 종료되는 것을 방지하기 위해 사용한다.

try-except 구문

[문법]

try:

    예외가 발생할 수 있는 코드

except:

    예외 발생 시 실행할 코드

else, finally 구문

[문법]

try:
    예외가 발생할 수 있는 코드

except:
    예외 발생 시 실행할 코드

else:
    예외 발생하지 않을 경우 실행할 코드(잘 사용하지 않음)

finally:
    항상 실행할 코드(리소스 반환 등)

예외처리 연습문제

- 연습문제 – 환율 계산하는 프로그램
- 문제 내용
  - 원화금액과 달러를 입력하면 '환율'을 출력하는 프로그램을 작성해보자.
  - 예외처리1 – 숫자가 아닌 문자열 입력 시(ValueError)
    "문자열 예외가 발생하였습니다." 출력
  - 예외처리2 – 달러가 0이 입력될 경우, 0으로 나눌 수 없으므로(ZeroDivisionError)
    "나누기 0은 불가능합니다"를 출력
  - else – "예외가 발생하지 않았습니다" 출력
  - finally – "예외가 발생하건, 하지 않건 무조건 실행하는 코드입니다" 출력

에러 발생시키기: raise 구문

- 에러를 임의로 발생시킬 수도 있다.
- [문법]

  raise 예외

  또는

  raise 예외(“에러 메시지”)

- 간단한 연습문제를 풀어보자

- <예시>

  raise ValueError

  또는

  raise ValueError (“에러 메시지”)

예외 계층 구조

- 예외에도 계층 구조가 있어 포함관계가 있다.
- 포함 관계가 있어도 예외처리의 원인을 알 경우, 세세하게 작성해준다.
- [url:https://docs.python.org/ko/3/library/exceptions.html#exception-hierarchy](https://docs.python.org/ko/3/library/exceptions.html#exception-hierarchy) (참고)

```
BaseException
 +-- SystemExit
 +-- KeyboardInterrupt
 +-- GeneratorExit
 +-- Exception
      +-- StopIteration
      +-- StopAsyncIteration
      +-- ArithmeticError
      |    +-- FloatingPointError
      |    +-- OverflowError
      |    +-- ZeroDivisionError
      +-- AssertionError
      +-- AttributeError
      +-- BufferError
      +-- EOFError
      +-- ImportError
      |    +-- ModuleNotFoundError
      +-- LookupError
      |    +-- IndexError
      |    +-- KeyError
      +-- MemoryError
      +-- NameError
      |    +-- UnboundLocalError
      +-- OSError
      |    +-- BlockingIOError
      |    +-- ChildProcessError
      |    +-- ConnectionError
      |    |    +-- BrokenPipeError
      |    |    +-- ConnectionAbortedError
      |    |    +-- ConnectionRefusedError
      |    |    +-- ConnectionResetError
      |    +-- FileExistsError
      |    +-- FileNotFoundError
      |    +-- InterruptedError
      |    +-- IsADirectoryError
      |    +-- NotADirectoryError
      |    +-- PermissionError
      |    +-- ProcessLookupError
      |    +-- TimeoutError
```

예외 처리 Mission

- **Mission: 반복문 Missoin3 단어 공부하기 문제를 “while문 + 예외처리”를 활용하여 풀어보자.**
- **조건**
  - **word_list에 테스트를 원하는 단어들을 저장**
  - **while 반복문을 사용**
  - **while 반복문 실행 중 오류가 발생(IndexError) 발생 시, while 반복문을 종료(break)**
  - **최종 결과로 ‘전체 문제수, 맞은 문제수, 틀린 문제 수’를 출력**

```
[English word typing practice]
apple
apple
banana
banana
pitch
pich
전체 문제 수: 3
맞은 문제 수: 2
틀린 문제 수: 1
```

Next

파일 입출력

파일 입출력을 사용하는 이유

1. 파일로부터 데이터를 읽어와서 프로그램에 사용하기 위해서

2. 프로그램에서 만든 데이터를 파일 형태로 저장하기 위해서

파일 열기 모드와 파일 입출력 과정

[파일 열기 모드]

- w : 쓰기 모드(write)
- a : 추가 모드(append)
- r : 읽기 모드(read)

[파일 입출력 과정]

1. 파일 열기
2. 파일 작업(쓰기, 추가, 읽기)
3. 파일 닫기

파일 쓰기(write)

[문법]

파일 객체 = open('디렉토리', 'w', encoding='utf-8')

파일 객체.write(데이터)

파일 객체.close()

파일 내용 추가하기(append)

[문법]

파일 객체 = open('디렉토리', 'a', encoding='utf-8')

파일 객체.write(데이터)

파일 객체.close()

파일 읽기(read): readline()함수 활용

- readline(): 파일에서 '한 줄 씩' 읽어 들이는 함수
- [문법]
  파일객체 = open('디렉토리', 'r', encoding='utf-8')

```
while True:
    data = file.readline()
    print(data)
    if data == "":
        break
file.close()
```

파일 읽기(read): readlines()함수 활용

- readlines(): 모든 줄을 읽어와, 각 줄을 “리스트의 요소”로 반환하는 함수
- [문법]

```
파일객체 = open('디렉토리', 'r', encoding='utf-8')
data = file.readlines()
for line in data:
    print(line)
파일객체.close()
```

파일 읽기(read): read함수 활용

- read(): 파일의 전체 내용을 '하나의 문자열(srt)'로 읽어들이는 함수
- [문법]

  파일객체 = open('디렉토리', 'r', encoding='utf-8')

  data = file.read()

  print(data)

  파일객체.close()

Next

이벤트 처리

이벤트 처리란?

- 이벤트 기반 프로그래밍

  : 순서에 관계없이 이벤트가 발생하면 해당 이벤트에 등록된 명령 묶음(함수)이 실행되는 프로그래밍(순차적 프로그래밍과는 대비되는 개념)

- 이벤트의 종류

  : 마우스 클릭, 키보드 입력, 다양한 센서들의 출력, 다른 시스템 or 프로그램으로부터 오는 메시지 등

이벤트 핸들러

- 이벤트 핸들러(**event handler**)

  : 이벤트가 발생했을 때 해당 이벤트를 처리할 수 있도록 작성된 명령어 묶음(함수)

  즉, 이벤트가 발생할 경우 실행되는 " 함수“

- 생성 방법: 일반 "함수 만들기"와 동일

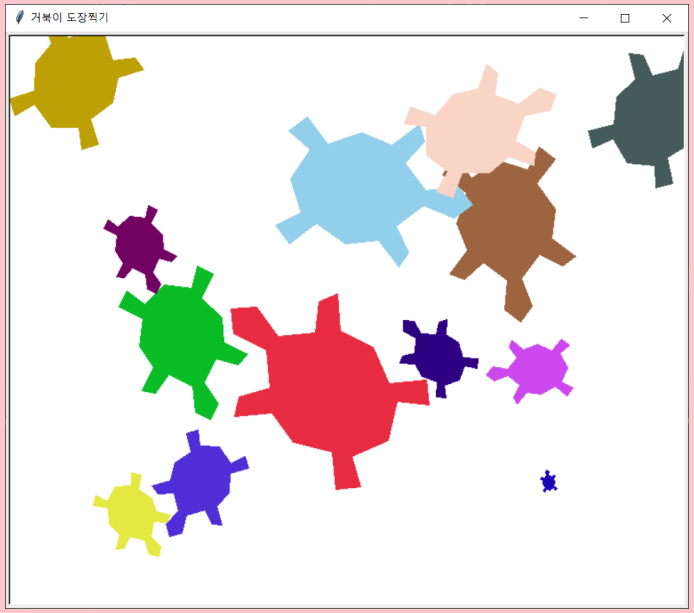
turtle 모듈 마우스 이벤트 처리

- turtle 모듈에서의 " 마우스 관련 이벤트“ 메서드
  - onscreenclick(함수, 번튼번호): 화면을 마우스 버튼으로 클릭 시 실행될 함수를 등록한다.
  - onclick(함수, 버튼 번호) : 거북이를 마우스 버튼으로 클릭 시 실행될 함수를 등록한다.
  - onrelease(함수, 버튼 번호) : 마우스 버튼에서 ‘뗄 때’실행될 함수를 등록한다.
  - ondrag(함수, 버튼번호) : 마우스 '드래그’시 실행될 함수를 등록한다.
  - 여기서 “버튼 번호”는
    - 2-버튼 마우스의 경우: 1=왼쪽 버튼, 2= 오른쪽 버튼
    - 3-버튼 마우스의 경우: 1=왼쪽 버튼, 2=가운데 버튼(휠 누르기), 3=오른쪽 버튼

turtle 마우스 이벤트 처리 연습문제

- 연습문제

  : turtle 모듈을 활용하여, 거북이 모양 스탬프를 찍는 프로그램을 만들어 보자.



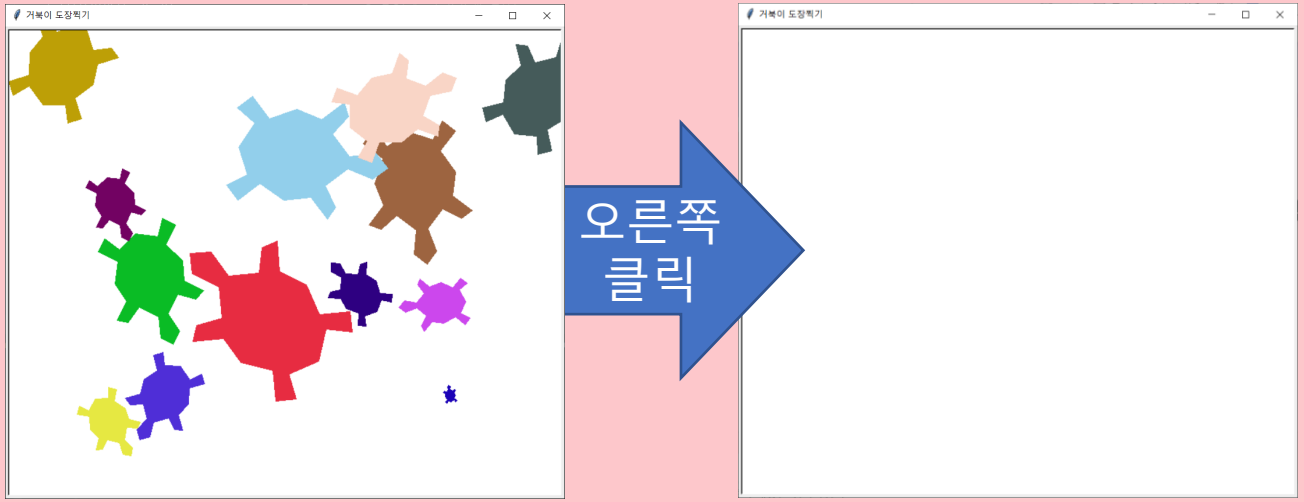
turtle 마우스 이벤트 처리 Mission

- **Mission**

  : turtle 마우스 이벤트 처리 연습문제에서 오른쪽 마우스 클릭을 하면,

  화면이 지워지도록 이벤트 처리를 추가해보자.

  ※ 화면을 지우는 명령 **t.clear()**



turtle 키보드 이벤트 처리

- turtle 모듈에서의 " 키보드 관련 명령어“
  - onkey(함수, 키): 주어진 키를 눌렀다 뗐을 때 호출된 함수를 등록한다.
  - onkeypress(함수, 키): 주어진 키를 눌렀을 때 호출될 함수를 등록한다.

    ※ 키 생략 시 아무 키나 눌러도 함수 실행
  - onkeyrelease(함수, 키): 주어진 키를 뗐을 때 호출될 함수를 등록한다.

turtle 키보드 이벤트 처리

- 키보드 관련 이벤트 '키' 자리에 들어갈 특수키 목록

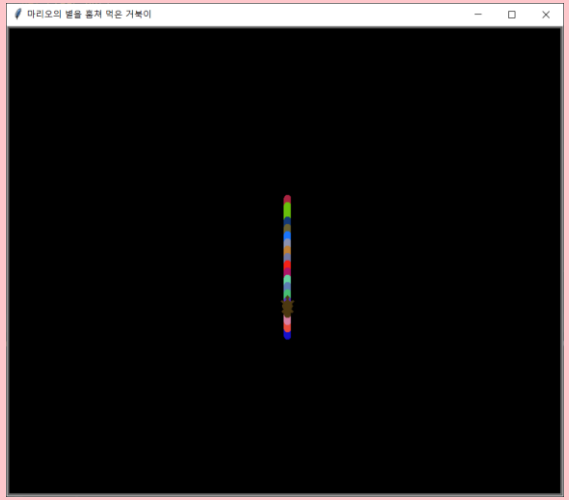
onkeypress(함수, 키)

이름	설명
Right	오른쪽 화살표 키
Left	왼쪽 화살표 키
Up	위쪽 화살표 키
Down	아래쪽 화살표 키
space	스페이스 바
Escape	Esc 키
Tab	탭 키

turtle 키보드 이벤트 처리 연습문제

- 연습문제

  : turtle 모듈을 활용하여 검은 배경에서 거북이가 위아래 방향키에 따라 움직이고, 움직인 자리에 랜덤한 색상의 선이 그어지는 프로그램을 만들어 보자.



turtle 키보드 이벤트 처리 Mission

- **Mission**
  - **turtle** 키보드 이벤트 처리 연습문제에서 좌우 키보드 이벤트를 위아래와 방향만 같고 같은 동작을 수행하도록 만들어 보자.
  - 또한, **space** 키를 누르면 화면이 지워지도록 만들어 보자.