



# 모듈과 라이브러리

## Python 기본 사항

발표자 이름

날짜

# 학습 내용

## 강의 핵심 내용

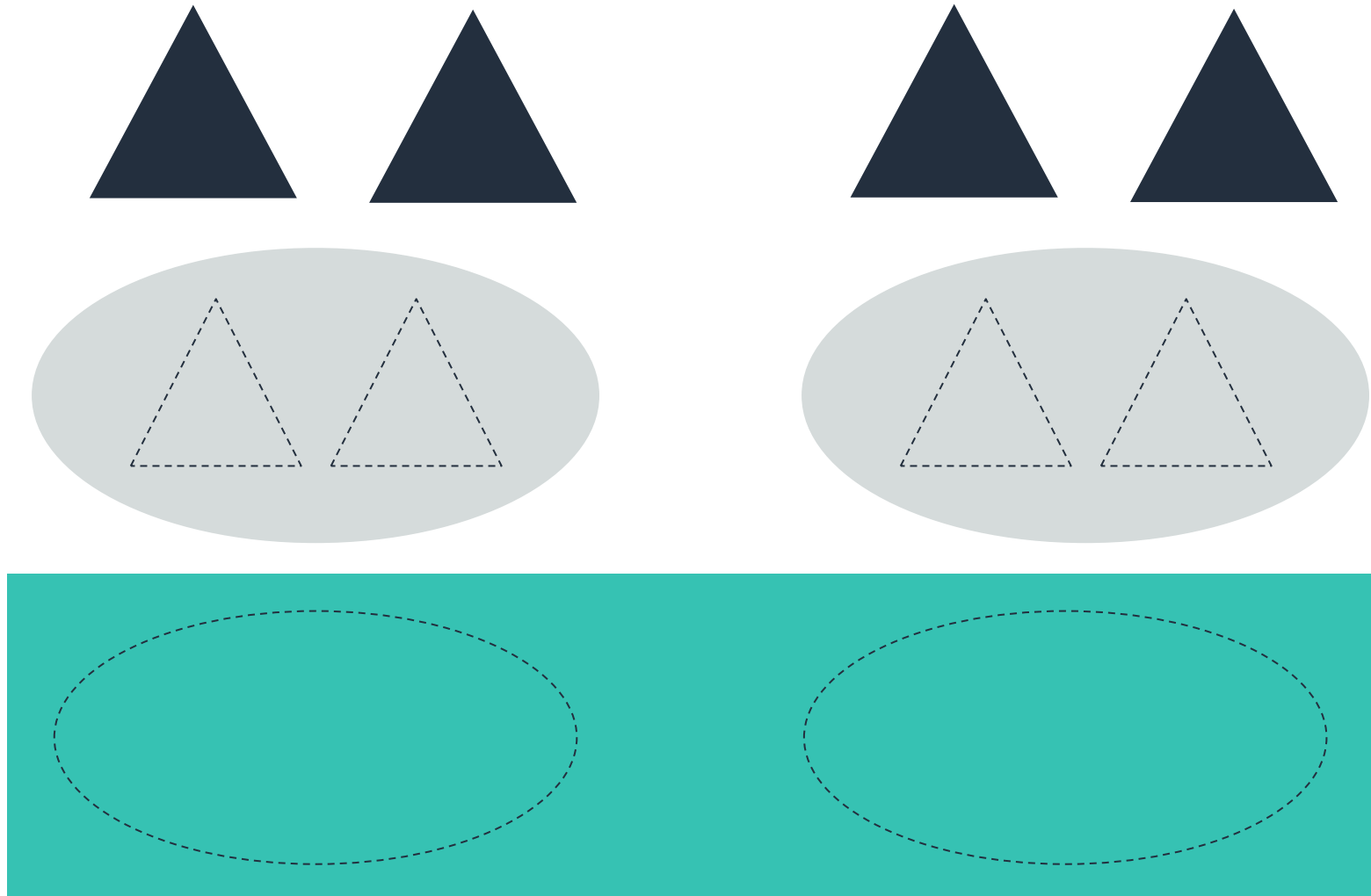
학습 내용:

- Python에서 모듈의 목적을 설명합니다.
- Python 스탠더드 라이브러리를 설명합니다.
- 라이브러리에서 모듈을 가져옵니다.
- 파일 핸들러를 사용해 Python 코드 내에서 파일을 열고, 읽고, 쓰고, 닫습니다.
- 예외 핸들러를 사용해 코드에서 오류를 포착합니다.
- JavaScript Object Notation(JSON) 모듈을 가져오고 JSON 함수를 사용합니다.
- Pip를 사용해 Python의 제3자 모듈을 다운로드하고 설치합니다.



# 모듈과 라이브러리란?

함수  
↓  
모듈  
↓  
라이브러리



# 스탠더드 라이브러리

Python 스탠더드 라이브러리는 Python 프로그램이 액세스할 수 있는 스크립트 모듈의 모음입니다. 프로그래밍 프로세스를 간소화하고 자주 사용되는 명령을 다시 써야 할 필요성을 줄여 줍니다.

Python 라이브러리는 C로 작성된 모듈도 포함할 수 있습니다.

실제 사례: Saanvi Sarkar는 휠을 만들지 않았습니다. 라이브러리에서 휠 모듈을 가져와 새로운 제품, 즉 자동차를 만드는 데 도움을 주었습니다.

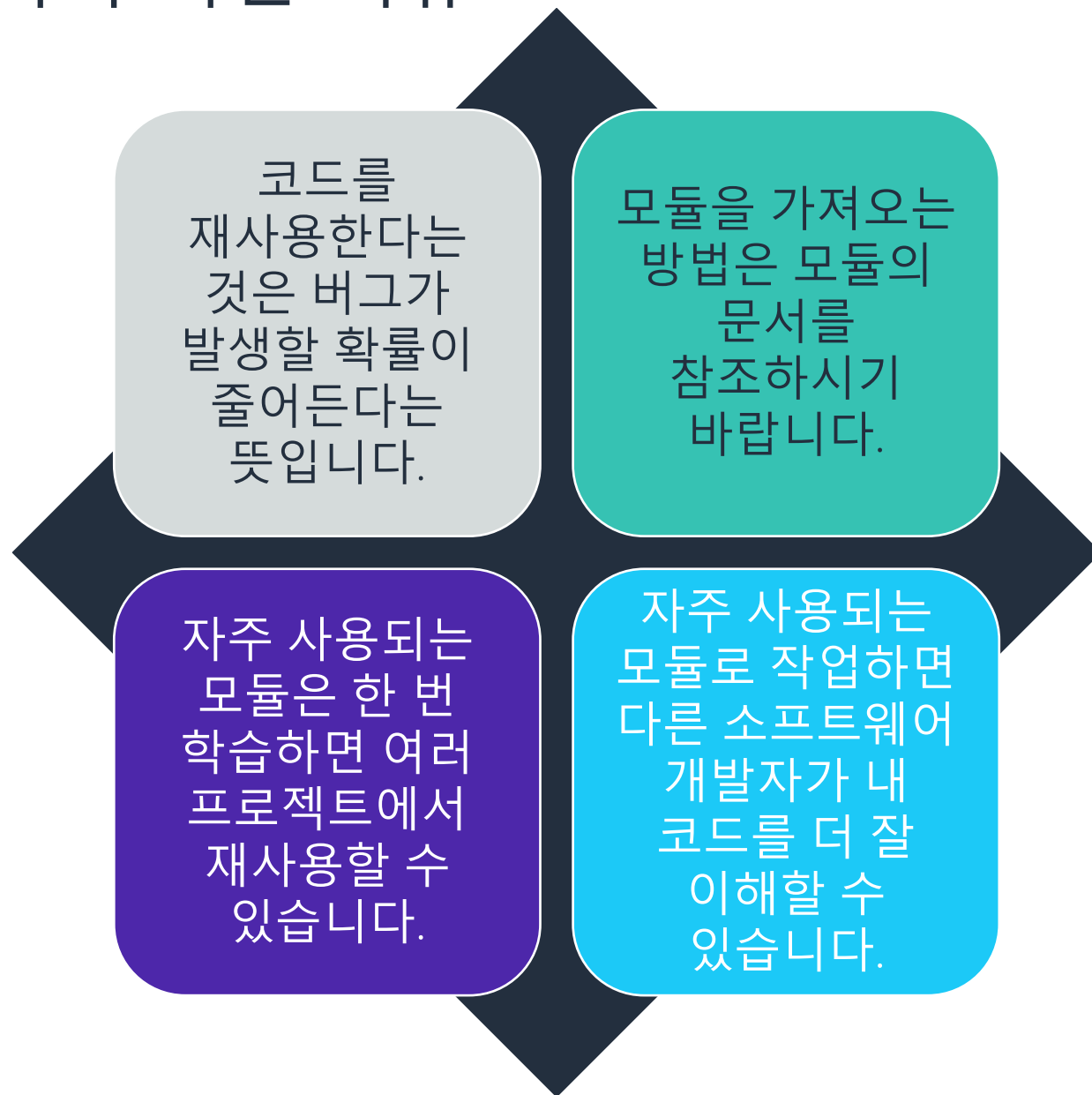
# Python 스탠더드 라이브러리 탐색

Python 스탠더드 라이브러리는 자주 사용되는 명령을 다시 쓸 필요가 없게 만들어 쉽게 프로그래밍할 수 있게 해 주는 모듈 모음입니다.

라이브러리에는 시간(time), 시스템 정보 가져오기(sys), 운영 체제 쿼리(os) 등의 기능에 액세스하기 위한 기본 제공 모듈이 포함되어 있습니다.

스탠더드 라이브러리를 최대한 활용하여 보다 쉽게 코드를 유지 관리하고 다른 플랫폼으로 복사할 수 있습니다.

# 모듈을 가져와야 하는 이유



모듈 유형은 다음과 같습니다.

1. 내가 모듈을 만들 수 있습니다. Python을 사용하여 그룹화된 특정 세트의 태스크 또는 함수를 완성해야 할 경우 태스크 또는 함수를 하나의 모듈로 묶으면 더 쉽습니다.
2. 다른 사람이 만든 외부 소스일 수 있습니다.
3. Python에 기본적으로 패키징되어 있거나 스탠더드 라이브러리에 속해 있을 수 있습니다.
  - 예: **math**, **time**, **random**

# 모듈 가져오기

스탠더드 라이브러리 모듈은 `import` 명령으로 가져옵니다. `from` 명령을 사용하면 모듈의 특정 함수 또는 상수를 가져올 수도 있습니다.

예를 들어 다음과 같습니다.

```
import math
```

```
-> use math.pi to access the pi constants
```

```
-> use math.exp(x) to access the exp function
```

```
from math import pi
```

```
-> use pi directly in your code
```

```
from math import exp
```

```
-> use exp directly in your code
```

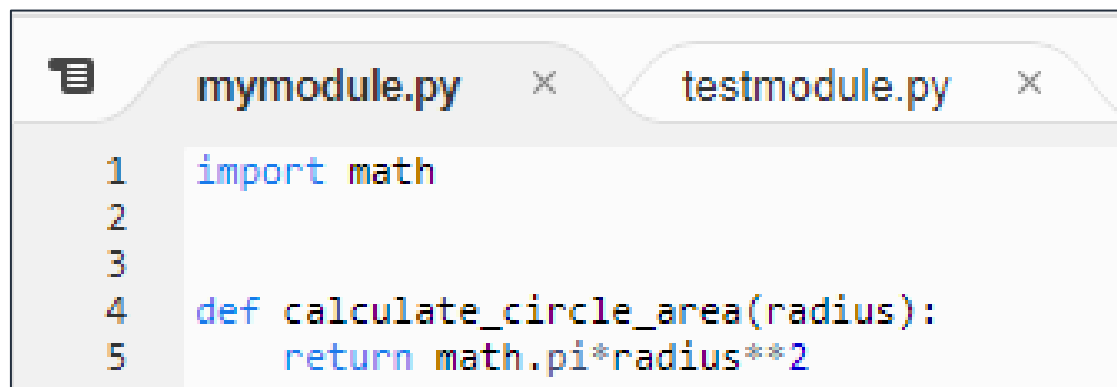
스탠더드 라이브러리에 속해 있다고 하더라도 모듈 또는 함수를 사용하려면 먼저 가져와야 합니다.



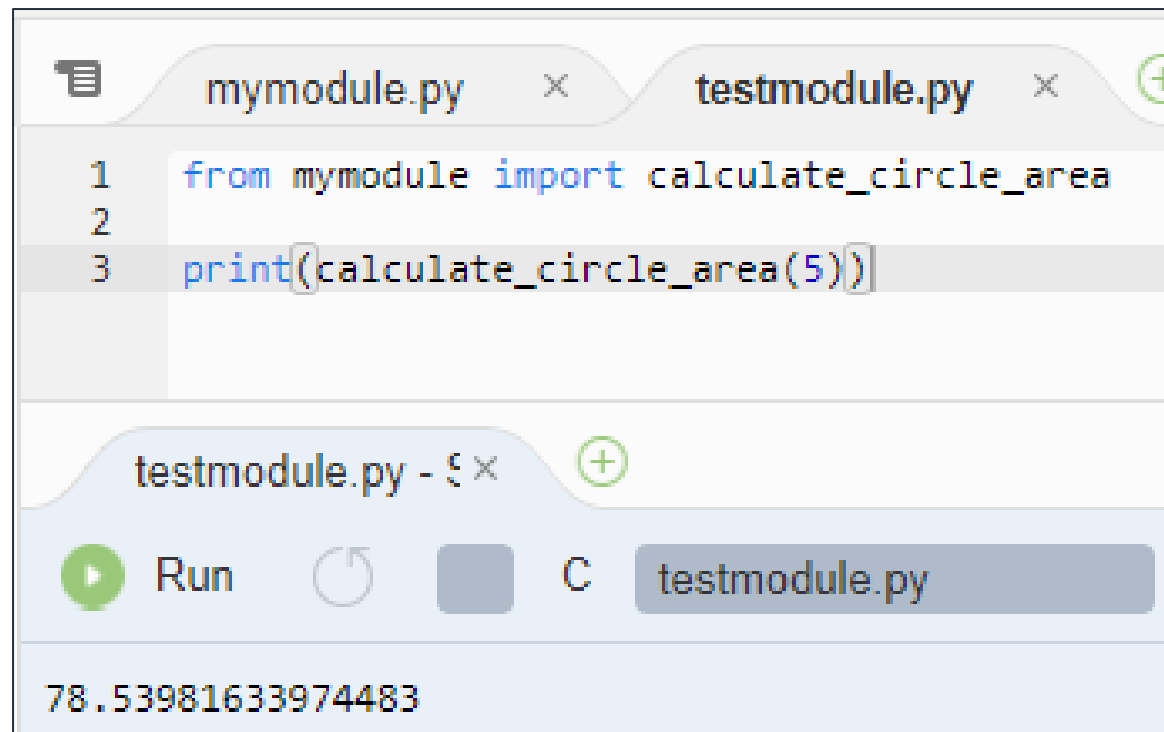
# 모듈 만들기

직접 모듈을 만들려면 다음 단계를 따르십시오.

- .py 확장명(예: mymodule.py)으로 파일 이름을 만듭니다.
- 함수를 정의하기 위한 코드를 추가합니다.
- 이제 다른 python 파일에 있는 mymodule을 가져와 모듈에서 정의된 코드를 사용할 수 있습니다.



```
1 import math
2
3
4 def calculate_circle_area(radius):
5     return math.pi*radius**2
```



```
1 from mymodule import calculate_circle_area
2
3 print(calculate_circle_area(5))
```

testmodule.py - 5x

Run [Refresh] [Close] testmodule.py

78.53981633974483

# 데모: Math 모듈



## 지침

1. **modules.py**라는 파일을 만듭니다.
2. **math** 모듈을 가져옵니다.
3. 절댓값 함수를 사용합니다. `math.fabs(x)`
4. 다음으로, 플로어 함수를 사용합니다. `math.floor(x)`
  - 이 함수는 부동 소수점 수를 취해  $x$ 보다 작거나 같은 정수 중 가장 큰 값을 반환합니다.
5. 결과를 출력하고 파일을 실행합니다.

## 추가 챌린지:

1. 내가 한 일을 설명하도록 **print** 스테이트먼트를 바꿀 수 있습니까?
2. `math.floor(x)`에 문자열을 입력하려고 하면 어떤 일이 발생합니까?

# 파일 핸들러

기본 제공되는 ()  
함수를 사용합니다.

문자열인 두 개의  
인수(파일 경로, 파일  
모드(읽기 또는 쓰기))  
는 해당 파일을  
나타내는 파일 핸들러  
객체를 반환합니다.

파일 핸들러는  
파일에서 데이터를  
읽거나 파일에  
데이터를 쓸 수  
있습니다. 또한 사용자  
대신 데이터를 전달할  
수 있는 다른 함수에  
데이터를 전달할 수  
있습니다(예: **json.load**,  
**json.dump**).

파일 핸들러는 사용  
후 연결된 **.close()**  
함수를 호출하여  
닫아야 합니다.

# 예제: 파일 핸들러

The screenshot displays the AWS re/start IDE interface. On the left, the 'Environment' pane shows a project named 'MyCloud9' with a subdirectory 'files' containing 'diary.txt' and 'diary2.txt'. The 'Source Control' pane is empty. The 'AWS' pane shows the 'readfiles.py' script. The main editor window shows the code for 'readfiles.py' and the output of the script.

```
1 print("Here is my diary: \n")
2 f1 = open("files/diary.txt", "r")
3 print(f1.read())
4 f1.close()
5 print("\nNow let's create another diary ! ")
6 f2 = open("files/diary2.txt", "w")
7 f2.write("Writing in my diary file!")
8 f2.close()
9
```

The output of the script is displayed in the bottom pane:

```
Here is my diary:
Dear diary,
I started learning Python today...

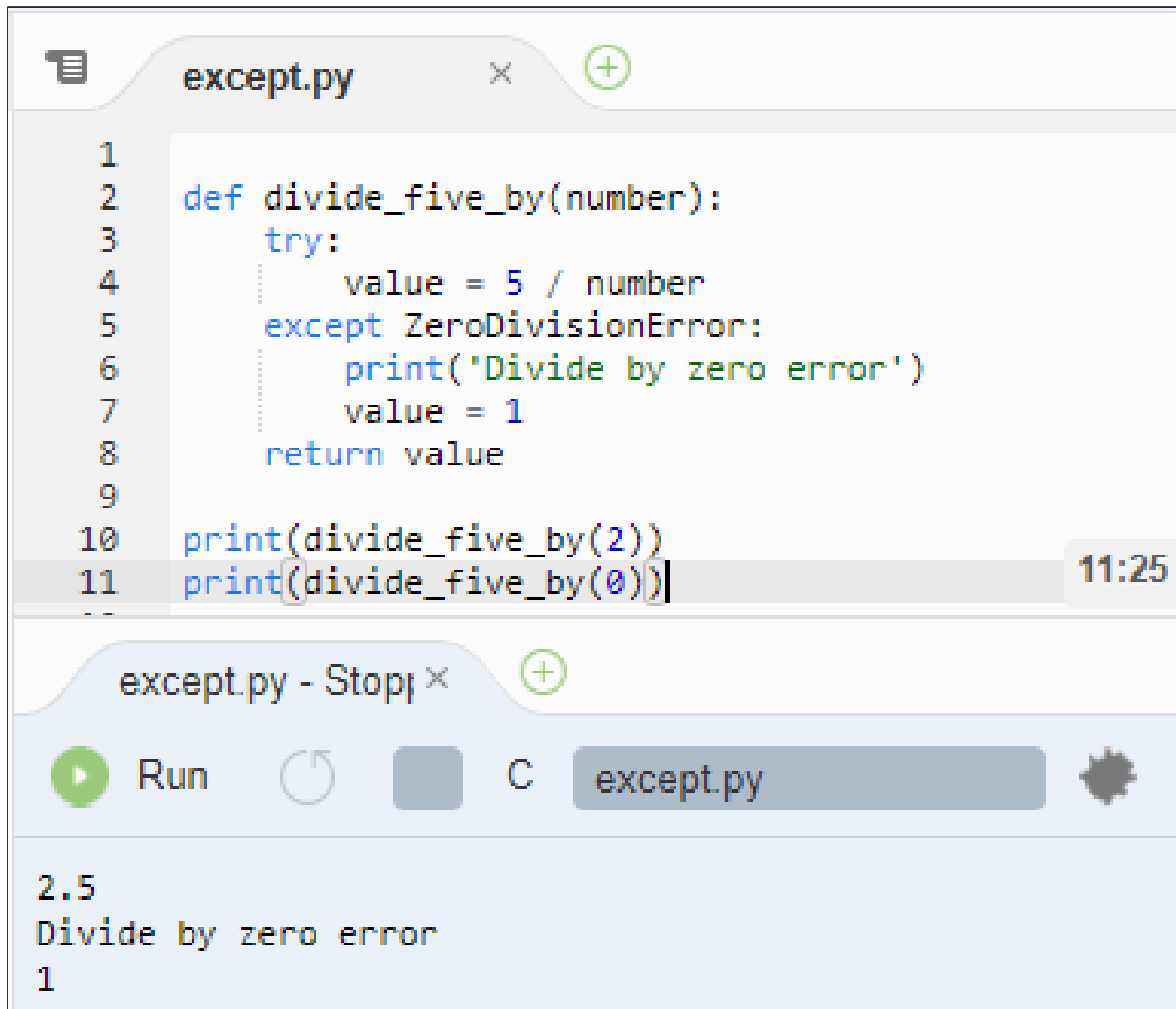
Now let's create another diary !

Process exited with code: 0
```

A zoomed-in view of the 'diary2.txt' file is shown on the right, displaying the text: 'Writing in my diary file!'.

# 예외 처리

- 예외 처리는 흐름 제어의 한 형태입니다.
- 오류가 발생하면 프로그램을 중단하는 대신 **try/except** 블록을 사용할 수 있습니다.
- 발생할 것으로 예상되는 예외를 하나 이상을 명시해야 합니다.



```
1
2 def divide_five_by(number):
3     try:
4         value = 5 / number
5     except ZeroDivisionError:
6         print('Divide by zero error')
7         value = 1
8     return value
9
10 print(divide_five_by(2))
11 print(divide_five_by(0))
```

except.py - Stop | Run | C | except.py

2.5  
Divide by zero error  
1

# 예제: 예외 처리



# OS: 운영 체제 모듈

- OS는 Python 스탠더드 라이브러리의 일부입니다.
- OS 모듈은 운영 체제 기능을 제공합니다. 모듈의 출력은 입력이 거의 같을 때 기본 운영 체제에 따라 다릅니다.
- OS 모듈의 일반적인 기능은 환경 변수 정보, 파일 조작, 디렉터리 순회, 프로세스 관리입니다.
- OS 모듈을 가져와서 사용하는 프로그램은 일반적으로 서로 다른 플랫폼 간의 이동이 보다 용이합니다.

# OS 기능

- 호스트 운영 체제:
  - **getlogin** - 로그인한 사용자의 이름을 반환합니다.
  - **getgrouplist** - 사용자가 속한 그룹 ID의 목록을 반환합니다.
  - **getenv** - 전달된 환경 변수의 값을 반환합니다.
  - **uname** - 현재 OS를 식별하기 위한 정보를 반환합니다.
  - **System** - 시스템의 하위 셸에서 명령을 실행하는 데 사용됩니다.
- 일반적인 파일 함수:
  - **Chown** - 파일의 소유권을 변경합니다.
  - **chmod** - 파일의 액세스 권한을 변경합니다.
  - **remove** - 지정된 경로의 파일을 제거합니다.
- 디렉터리에 사용하는 일반적인 OS 모듈 함수:
  - **getcwd** - 현재 작업 중인 디렉터리를 가져옵니다.
  - **listdir** - 현재 디렉터리의 내용을 나열합니다.
  - **mkdir** - 새로운 디렉터리를 만듭니다.



# OS: 호스트 시스템에서 명령 실행

```
>>> os.system("command on the system to run")
```

```
>>> os.system("adduser newuser")
```

```
>>> os.system("whoami")
```

```
>>> os.system("powershell.exe")
```

- 사용한 형식
- 사용자를 추가하는 Linux 명령
- Linux 또는 Microsoft Windows: 현재 사용자 표시
- 일반적으로 Microsoft Windows에만 해당

## 예제: OS 모듈 - 디렉터리

```
>>> import os
>>> os.getcwd()'/home/username/folder'
>>> os.listdir()'"oldfolder"'
>>> os.mkdir("newfolder")
>>> os.listdir()'"newfolder", "oldfolder"'
```

# 데모: OS 모듈로 작업



## 지침

1. 현재 작업하는 디렉터리를 나열합니다.
2. 현재 작업하는 디렉터리에서 새 폴더를 만든 후 삭제합니다.
3. 현재 로그인한 사용자에 관한 정보를 출력합니다.

# JSON

- JSON은 JavaScript Object Notation의 약어입니다.
- JSON은 데이터 객체를 전송하는 표준 파일 형식으로, 언어와 별개로 사용할 수 있습니다. 처음에는 JavaScript에서 파생되었지만 이제 Python을 비롯해 최근에 개발된 언어에서도 JSON을 생성하고 구문 분석하는 기능을 제공합니다.
- JSON은 객체를 **직렬화**하는 데 사용됩니다. 즉, JSON은 채널을 통해 코드를 전송할 수 있는 문자열로 변환합니다.

```
{
  "users": [
    {
      "name": "John Doe",
      "age" : 25
    },
    {
      "name" : "Li Juan",
      "age" : 29
    },
    {
      "name" : "Sofía Martínez",
      "age" : 22
    }
  ],
  "dataTitle": "JSON Tutorial!",
  "swiftVersion" : 2.1
}
```

# JSON(계속)

- JSON 모듈의 유용한 네 가지 함수는 다음과 같습니다.
  - **dump**
  - **dumps**
  - **load**
  - **loads**
- **dump**와 **dumps**: 다양한 유형의 구조화된 데이터를 문자열로 변환하여 파일에 쓸 수 있습니다.
- **load**와 **loads**: 문자열을 다시 구조화된 데이터로 변환합니다.
- **dump**와 **load**: 파일에서 직접 수행됩니다.
- **Dumps**와 **loads**: 문자열에서 수행됩니다.
  - 이름 끝의 s가 **문자열(string)**을 나타냅니다.

# JSON의 용도

앞서 언급한 바와 같이 JSON은 대용량의 데이터를 문자열로 바꾸고 올바른 데이터 유형으로 되돌리는 데 사용됩니다.

JSON은 부동 소수점 수를 취해 문자열로 바꿈으로써 정보를 쉽게 전달할 수 있도록 합니다. 그런 다음, 정보를 받으면 문자열을 다시 부동 소수점 수로 바꿉니다.

이 기법은 데이터 세트 내에 데이터 유형이 여러 개일 때 특히 유용합니다. Boolean, Integer, Float, String을 하나의 문자열로 변환할 수 있습니다.

그런 다음 이 정보를 파일이나 웹페이지에 저장할 수 있습니다.

데이터를 가져와서 되돌리면 데이터의 형식은 여전히 Boolean, Integer, Float, String입니다.

## 예제: JSON 모듈

```
>>> import json
>>> json.dumps(5)
'5'
>>> json.dumps([1, "string", 3])
'[1, "string", 3]'
>>> json.loads('[1, "string", 3]')
[1, "string", 3]
```

# 데모: 사람을 인식하고 이름으로 인사



## 지침

1. JSON 모듈을 사용하여 과거 실행 기록을 저장하고 불러옵니다.
  - a. **helloworld.py**를 만듭니다.
  - b. JSON 모듈을 가져옵니다.
  - c. 파일에 데이터를 씁니다.
  - d. 파일을 검사합니다.
2. 흐름 제어에 if 스테이트먼트를 사용합니다.
  - a. 파일에서 읽도록 helloworld.py를 업데이트합니다.
  - b. 파일에 데이터가 있으면 인사말에 사용합니다. 그렇지 않으면 사용자의 이름을 요청합니다.
3. Try/except 블록은 예외를 처리하는 데 사용됩니다.
  - a. 프로그램을 다시 업데이트하여 파일 오류 처리에 try/catch 블록을 사용하도록 합니다.



# Pip란?

Pip는 Python용 패키지 관리자이며 Linux의 apt와 유사합니다.

- 제3자 패키지를 설치하는 데 사용됩니다. 패키지에는 하나 이상의 Python 모듈이 포함되며, 이를 코드에 사용할 수 있습니다.
- Python과 함께 설치됩니다.

Pip는 Python 내에서 호출되지 않고 Python처럼 명령줄에서 호출됩니다.

# 데모: pip로 Requests 설치



## 지침

1. Requests가 이미 설치되어 있는지 확인하기 위해 테스트합니다.
  - 테스트하려면 명령줄에 **pip show requests**를 입력합니다.
2. 결과를 검토합니다.
3. pip로 Requests를 설치합니다.
  - **pip install requests** 명령을 입력합니다.
4. 결과를 검토합니다.
5. Requests가 설치되었는지 테스트합니다.
  - 이전에 사용한 것과 동일한 명령을 사용합니다.
6. 결과를 검토합니다.

# 학습 내용 확인

함수, 모듈, 라이브러리의 관계는 무엇입니까?

파일 핸들러의 목적은 무엇입니까?

시스템 명령을 실행하는 데 Python을 사용할 수 있습니까?

Python에서 예외는 어떻게 사용됩니까?

True 또는 False: JSON 라이브러리를 사용하면 데이터의 올바른 데이터 유형이 자동 생성됩니다.

Pip란 무엇입니까?

Pip를 사용해야 하는 이유는 무엇입니까?

# 요점



- Python은 모듈과 라이브러리를 이용해 자주 사용되거나 많은 시스템에 배포된 코드를 저장합니다.
- os와 같은 모듈을 사용하여 개발자가 운영 체제 명령을 실행할 수 있습니다.
- JSON 모듈로 JSON 파일을 만들고, 읽고, 쓸 수 있습니다.
- Pip는 Python이 사용하는 Linux 패키지를 쉽고 빠르게 관리하는 데 사용됩니다.