

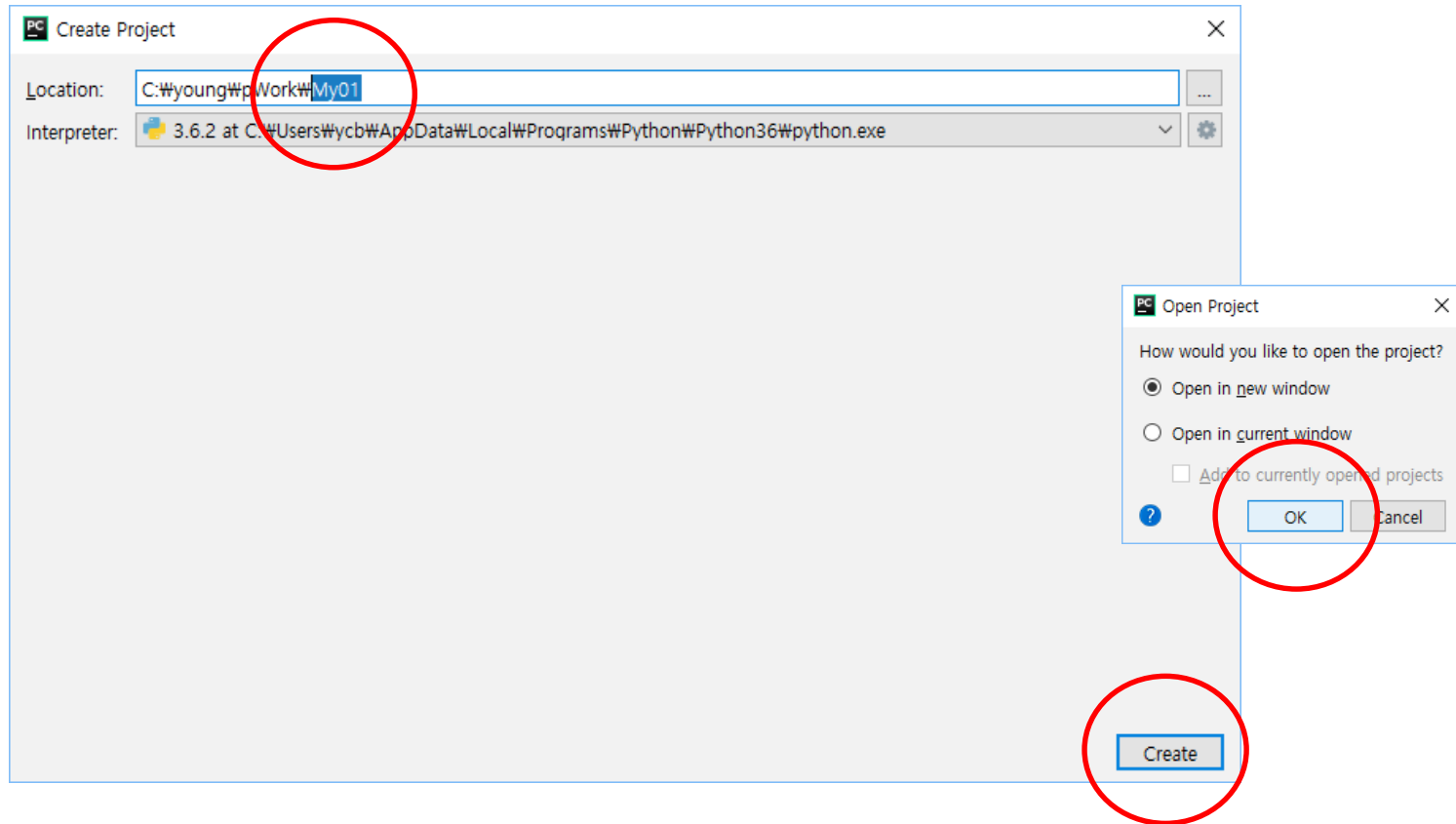
Python 클래스 작성 및 모듈화

변영철 교수

(ycb@jejunu.ac.kr)

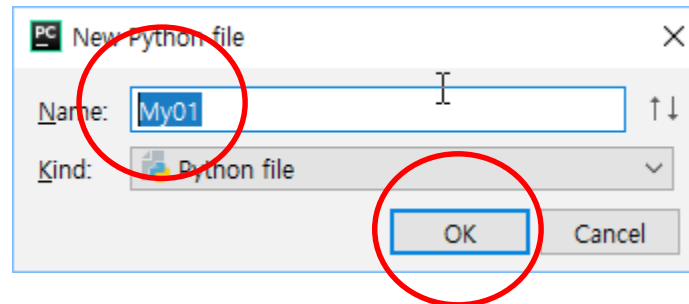
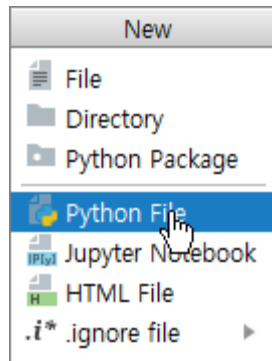
1절. 아주 간단한 Python 프로그램

- File | New Project



1절. 아주 간단한 Python 프로그램

- File | New...



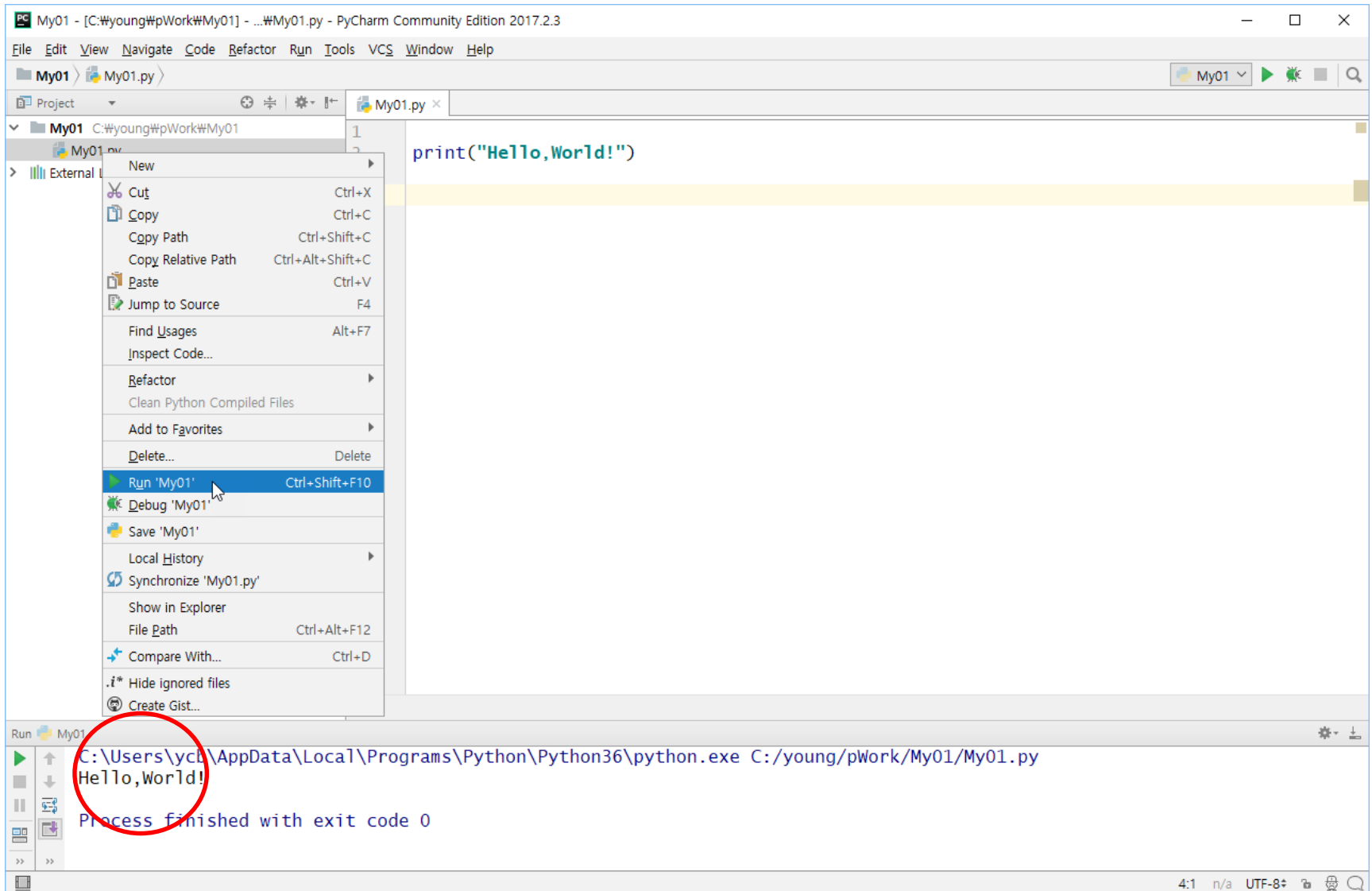
1절. 아주 간단한 Python 프로그램

- 코드 입력

```
print("Hello,World!")
```

- 파이썬은 실행이 시작 되는 메인 함수가 없음.
- 그 대신 들여쓰기 하지 않은 모든 코드(level 0 코드)가 실행됨.

1절. 아주 간단한 Python 프로그램



1절. 아주 간단한 Python 프로그램

```
if __name__ == "__main__":  
    print("Hello, World!")
```

"실행된 코드라면"

- `__name__` 내장 변수
- 해당 코드가 직접 실행되는 경우 `__name__`에 `__main__`이 자동으로 설정됨.

1절. 아주 간단한 Python 프로그램

- 코드 입력

```
def main():  
    print("Hello,World!")
```

```
main()
```

2절. 조금 복잡한 Python 프로그램

- 아래와 같이 2개의 지역변수를 이용하자.

```
def main():  
    iX = 2  
    iY = 3  
    iResult = iX + iY  
  
    print("Sum = ", iResult)  
  
main()
```


2절. 조금 복잡한 Python 프로그램

- 전역변수 만들기(?)

```
iX = 0
```

```
iY = 0
```

```
def main():
```

```
    iX = 2
```

```
    iY = 3
```

```
    iResult = iX + iY
```

```
    print("Sum = ", iResult)
```

```
main()
```

3절. 함수를 이용한 프로그램

- 추상화(abstraction)
 - 복잡한 내용을 간단하게 줄여서 표현하는 것
 - 예) 어제 무엇을 했나요?
- 코드 추상화
 - 복잡한 코드를 간단히 표현하는 것
 - 코드 추상화 하기 : Assign, Add

3절. 함수를 이용한 프로그램

- 코드 추상화 – Assign(), Add()

iX = 0

iY = 0

```
def Assign(a, b):
```

```
    iX = a
```

```
    iY = b
```

실행결과:

Sum = 0

Process finished with exit code 0

```
def Add():
```

```
    return iX + iY
```

```
def main():
```

```
    Assign(2, 3)
```

```
    iResult = Add()
```

```
    print("Sum = ", iResult)
```

3절. 함수를 이용한 프로그램

```
iX = 0
```

```
iY = 0
```

```
def Assign(a, b):
```

```
    global iX, iY
```

```
    iX = a
```

```
    iY = b
```

```
def Add():
```

```
    global iX, iY
```

```
    return iX + iY
```

```
def main():
```

```
    Assign(2, 3)
```

```
    iResult = Add()
```

```
    print("Sum = ", iResult)
```

```
main()
```

실행결과:

Sum = 5

Process finished with exit code 0

4절. 클래스 모듈화

```
class Point:
```

```
    iX = 0
```

```
    iY = 0
```

```
    def Assign(self, a, b):
```

```
        self.iX = a
```

```
        self.iY = b
```

```
    def Add(self):
```

```
        return self.iX + self.iY
```

```
def main():
```

```
    gildong = Point()
```

```
    gildong.Assign(2, 3)
```

```
    iResult = gildong.Add()
```

```
    print("Sum:", iResult)
```

```
main()
```

4절. 다른 파일로 분리

```
#My.py
import point

def main():
    gildong = point.Point()
    gildong.Assign(2, 3)
    iResult = gildong.Add()
    print("Sum:", iResult)

main()
```

```
#point.py
class Point:
    iX = 0
    iY = 0

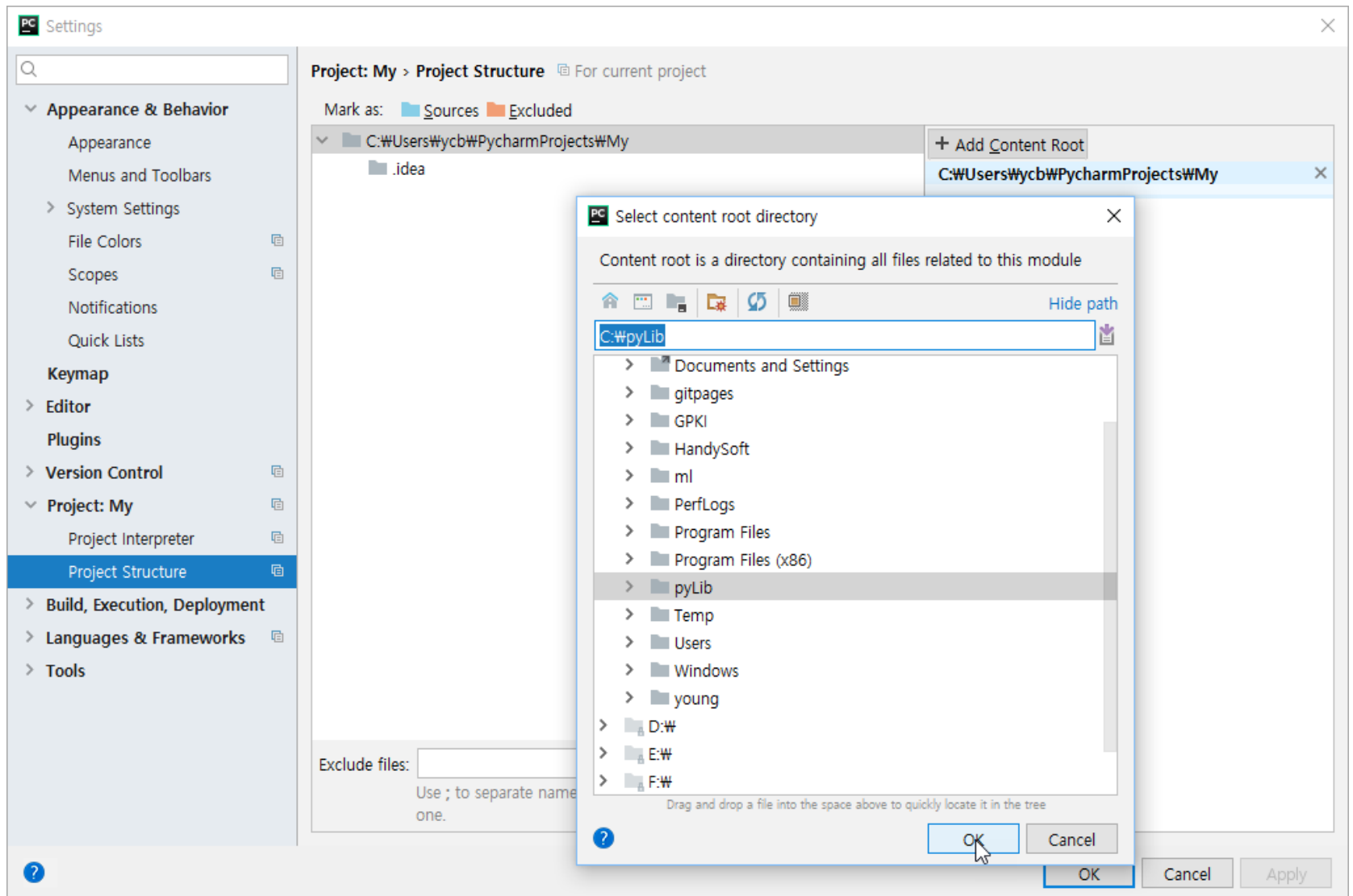
    def Assign(self, a, b):
        self.iX = a
        self.iY = b

    def Add(self):
        return self.iX + self.iY
```

5절. 나만의 라이브러리 폴더로 저장

- c:\WpyLib 폴더 생성 후 point.py를 그곳으로 이동

5절. 나만의 라이브러리 폴더 설정



My > My.py

Project

- My
 - My C:\Users\ycb\PycharmProjects\My
 - My.py
 - pyLib C:\pyLib
 - point.py
- External Libraries

```
1 #My.py
2 import point
3
4 def main():
5     gildong = point.Point()
6     gildong.Assign(2, 3)
7     iResult = gildong.Add()
8     print("Sum:", iResult)
9
10 main()
11
12
13
```

Run My

C:\Users\ycb\AppData\Local\Programs\Python\Python36\python.exe C:/Users/ycb/PycharmProjects/My/My.py

Sum: 5

Process finished with exit code 0

6절. import as

```
#My.py
import point as p

def main():
    gildong = p.Point()
    gildong.Assign(2, 3)
    iResult = gildong.Add()
    print("Sum:", iResult)

main()
```

assert a==7, "식을 **확인**해보고 아니면 이 문장 출력해줘."

a = 7

assert a == 7, "7이 아님!"

```
def checkio(number: int) -> str:
```

```
    return str(number)
```

```
assert checkio(15) == "Fizz Buzz", "Fizz Buzz를 반환해야 합니다!"
```

```
assert checkio(6) == "Fizz", "Fizz를 반환해야 합니다!"
```

```
assert checkio(5) == "Buzz", "Buzz를 반환해야 합니다!"
```

```
assert checkio(7) == "7", "입력한 숫자를 반환해야 합니다!"
```

```
def checkio(number: int) -> str:
    if number % 15 == 0:
        return "Fizz"
    elif number % 3 == 0:
        return "Fizz"
    elif number % 5 == 0:
        return "Buzz"
    return str(number)
```

```
assert checkio(15) == "Fizz Buzz", "Fizz Buzz를 반환해야 합니다!"
assert checkio(6) == "Fizz", "Fizz를 반환해야 합니다!"
assert checkio(5) == "Buzz", "Buzz를 반환해야 합니다!"
assert checkio(7) == "7", "입력한 숫자를 반환해야 합니다!"
```

CheckiO

JavaScript

Python

Coding games for beginners and advanced programmers where you can improve your coding skills by solving engaging challenges and fun task using [Python](#) and [JavaScript](#)

Partners:



BETA

Educational coding games for students. Build your base and defeat your enemies through a variety of unique, entertaining and challenging coding activities starting from the basic level upward to advanced.

Empire of Code

[Continue the game](#)

JavaScript

Python

<https://checkio.org>