

[Opensource Basic Project] 오픈소스 기초 프로젝트

- 5주차 : Functions and module

2024. 4. 3(수).

우창우

Dr.woo@chungbuk.ac.kr

□ 주차별 강의계획

※4.10(국회의원 선거날), 5.15(부처님 오신날)

주차(변경 전)	주차(변경 후)	강의계획	과제	번호	제출 마감일
1주(03.06)	1주(03.06)	Python Started	VScode 설치화면, "Hello World" 출력화면 및 작성코드	1	03.06~ 03.09(토)
2주(03.13)	2주(03.13)	Variables and simple data types	다이아몬드 모양, 진수 변환 프로그램 및 동전 변환 프로그램의 출력화면/작성코드	2	03.13~ 03.16(토)
3주(03.20)	3주(03.20)	Control Statement	종합계산기, 구구단출력 프로그램의 출력화면/작성코드	3	03.20~ 03.23(토)
4주(03.27)	4주(03.27)	Lists, Dictionaries, String	음식 궁합 출력, 문자열 거꾸로 출력 프로그램의 출력화면/작성코드	4	03.27~ 03.30(토)
5주(04.03)	5주(04.03)	Functions and module	GitHub Copilot Extension 설치화면, 로또번호 추첨 출력화면/작성코드	5	04.03~ 04.06(토)
6주(04.10)	6주(04.17)	Classes		6	04.17~ 04.20(토)
7주(04.17)	7주(04.24)	Window programming		7	04.24~04.27(토)
8주(04.24)	8주(05.01)	Midterm (20%)		-	-

□ 주차별 강의계획

※ 4.10(국회의원 선거날), 5.15(부처님 오신날)

주차(변경 전)	주차(변경 후)	강의계획	과제	번호	제출 마감일
9주(05.01)	9주(05.08)	Files and Exceptions		8	05.08~05.11(토)
10주(05.08)	10주(05.22)	Database		9	05.22~05.25(토)
11주(05.15)	11주(05.29)	Data visualization		10	05.29~06.01(토)
12주(05.22)	12주(06.05)	Quiz (10%), Team project implementation I		-	-
13주(05.29)	13주(06.12)	Team project implementation II		-	-
14주(06.05)	14주(06.17)	Team project implementation III		-	-
15주(06.12)	15주(06.18)	Project Presentation (50%)		-	-

Chapter 9. 함수와 모듈

Selection 01. 이장에서 만들 프로그램

Selection 02. 함수기본

Selection 03. 지역 변수, 전역 변수

Selection 04. 함수의 반환값과 매개변수

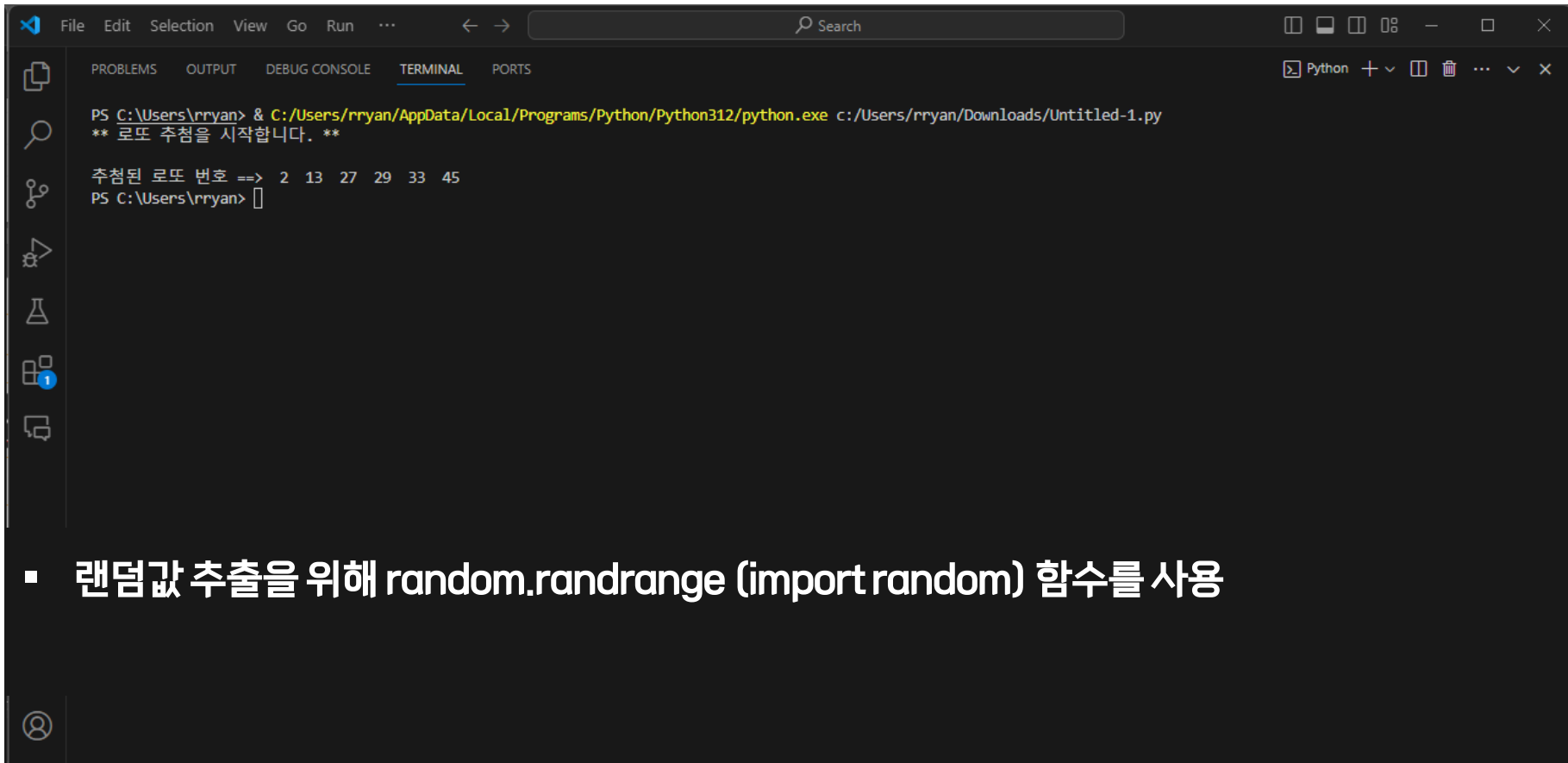
Selection 05. 모듈

Selection 06. 함수의 심화 내용

Selection 01. 이장에서 만들 프로그램 (실습&과제#1)

1. 로또 번호 추천

- 1~45의 숫자 중에서 6개를 뽑는 프로그램



```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
** 로또 추첨을 시작합니다. **

추첨된 로또 번호 ==> 2 13 27 29 33 45
PS C:\Users\rryan>
```

- 랜덤값 추출을 위해 `random.randrange (import random)` 함수를 사용

Selection 02. 함수 기본

1. 함수의 개념과 필요성

- 함수(Function) : '무엇'을 넣으면 '어떤 것'을 반환하는 기능으로 특정 용도의 코드를 한곳에 모아 놓은 것
- 메서드(Method)와 차이점 : 함수는 외부에 별도로 있지만 메서드는 클래스 안에 존재
- 함수는 파이썬 자체에서 제공하기도 하지만 사용자가 직접 만들어서 사용할 수도 있음
- 함수의 생성을 위해서는 def에 함수 이름을 지정(header)하고 ()와 :을 붙인 뒤 다음줄부터 내용(body)을 작성
- 함수의 형식

```
함수명()
```

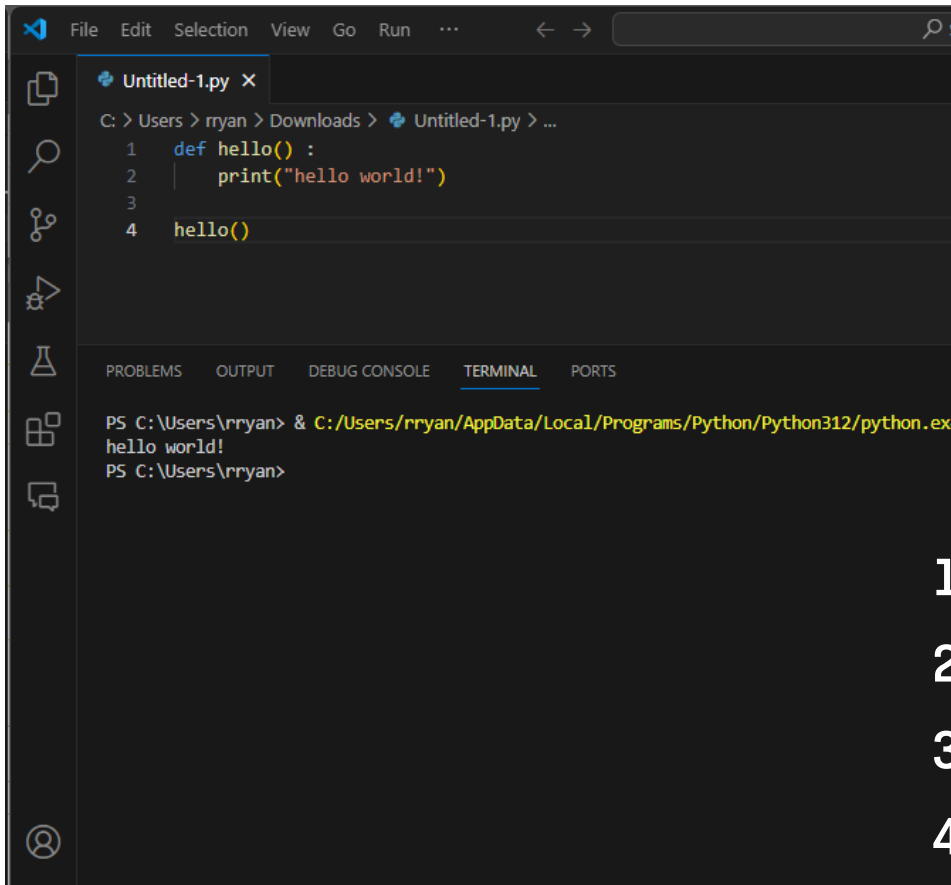
- Ex) print() 함수

```
print("CookBook-파이썬")
```

Selection 02. 함수 기본

1. 함수의 개념과 필요성

- (예시) hello world!를 출력하는 함수의 생성과 호출 예시



파이썬 스크립트

```
1 def hello():
2     print("hello world!")
3
4 hello()
```

① def hello(): ③
⑤ print('Hello, world!') ④
hello() ②
⑥

Hello()
print('Hello, world!')

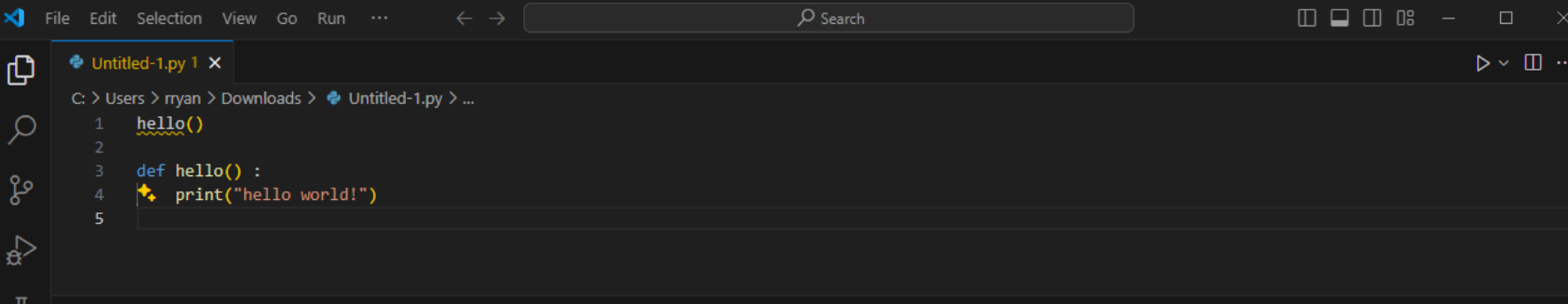
〈함수의 실행 순서〉

1. hello 함수 호출
2. hello 함수 실행
3. Print 함수 실행 및 'hello world!' 출력
4. Hello 함수 종료

Selection 02. 함수 기본

1. 함수의 개념과 필요성

- (예시) hello world!를 출력하는 함수의 생성과 호출 예시



The screenshot shows a code editor with a Python file named 'Untitled-1.py'. The code contains a function definition for 'hello()' and a call to 'hello()' on the same line. The terminal window shows the command to run the script, followed by a traceback indicating a 'NameError: name 'hello' is not defined. Did you mean: 'help'?'. The error message is highlighted with a red box.

```

1  hello()
2
3  def hello() :
4      print("hello world!")
5

```

```

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
Traceback (most recent call last):
  File "c:\Users\rryan\Downloads\Untitled-1.py", line 1, in <module>
    hello()
    ^^^^^
NameError: name 'hello' is not defined. Did you mean: 'help'?
PS C:\Users\rryan>

```

- ✓ 함수를 만들기 전에 함수를 호출하는 경우 함수가 정의되지 않았다는 오류가 발생
- ❌ 파이썬 코드는 위에서 아래로 순차적으로 실행되기 때문 (=함수를 만든 뒤 호출이 필요)

Selection 02. 함수 기본

1. 함수의 개념과 필요성

- (예시) 커피 자판기가 없는 상황에서 커피를 서비스 하려는 과정

```
1 coffee = 0
2
3 coffee = int(input("어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) "))
4
5 print()
6 print("#1. 뜨거운 물을 준비한다.");
7 print("#2. 종이컵을 준비한다.");
8
9 if coffee == 1 :
10     print("#3. 보통커피를 탄다.")
11 elif coffee == 2 :
12     print("#3. 설탕커피를 탄다.")
13 elif coffee == 3 :
14     print("#3. 블랙커피를 탄다.")
15 else :
16     print("#3. 아무거나 탄다.\n")
17
18 print("#4. 물을 붓는다.");
19 print("#5. 스푼으로 젓는다.");
20 print()
21 print("손님~ 커피 여기 있습니다.");
22
23
24
25
26
27
28
```

보통커피
설탕커피
블랙커피

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) 1

#1. 뜨거운 물을 준비한다.
#2. 종이컵을 준비한다.
#3. 보통커피를 탄다.
#4. 물을 붓는다.
#5. 스푼으로 젓는다.

손님~ 커피 여기 있습니다.
PS C:\Users\rryan>

Selection 02. 함수 기본

1. 함수의 개념과 필요성

- (예시) 커피 자판기를 보유한 상태로 커피를 서비스 하려는 과정

```
1  ## 전역 변수 선언 부분 ##
2  coffee = 0
3
4  ## 함수 선언 부분 ##
5  def coffee_machine(button) :
6      print()
7      print("#1. (자동으로) 뜨거운 물을 준비한다.");
8      print("#2. (자동으로) 종이컵을 준비한다.");
9
10     if button == 1 :
11         print("#3. (자동으로) 보통커피를 탄다.");
12     elif button == 2 :
13         print("#3. (자동으로) 설탕커피를 탄다.");
14     elif button == 3 :
15         print("#3. (자동으로) 블랙커피를 탄다.");
16     else :
17         print("#3. (자동으로) 아무거나 탄다.\n")
18
19     print("#4. (자동으로) 물을 붓는다.");
20     print("#5. (자동으로) 스푼으로 젓는다.");
21     print()
22
23 ## 메인 코드 부분 ##
24 coffee = int(input("어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙)"))
25 coffee_machine(coffee)
26 print("손님~ 커피 여기 있습니다.");
27
```

Diagram illustrating the coffee service process:

- A barista (server) is shown interacting with a coffee machine.
- The coffee machine is shown dispensing coffee into a cup.
- The barista is shown serving the coffee to a customer.

Terminal Output:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) 1

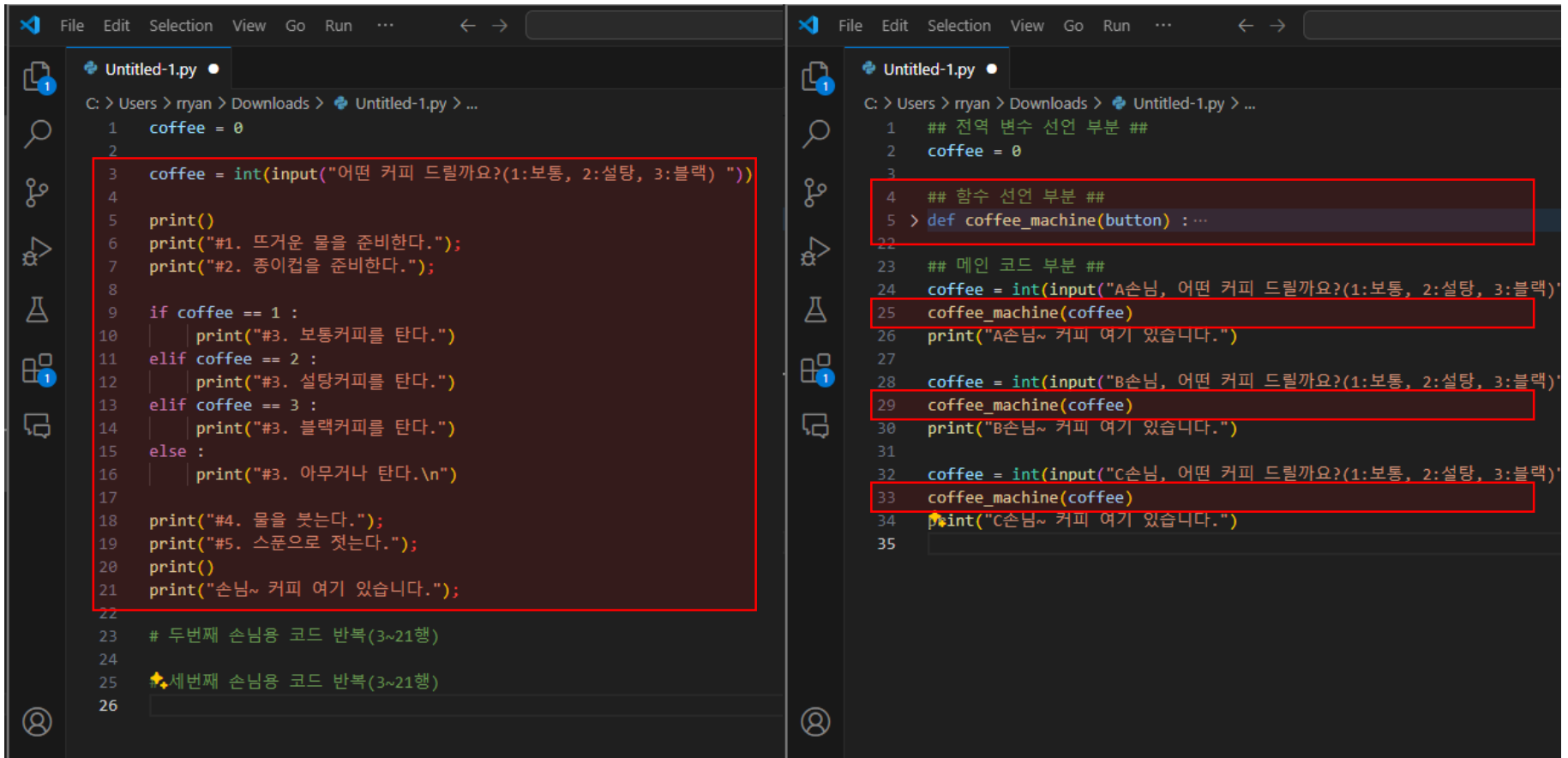
#1. (자동으로) 뜨거운 물을 준비한다.
#2. (자동으로) 종이컵을 준비한다.
#3. (자동으로) 보통커피를 탄다.
#4. (자동으로) 물을 붓는다.
#5. (자동으로) 스푼으로 젓는다.

손님~ 커피 여기 있습니다.
PS C:\Users\rryan>
```

Selection 02. 함수 기본

1. 함수의 개념과 필요성

- (예시) 손님 3명이 연속해서 들어온다고 했을 때



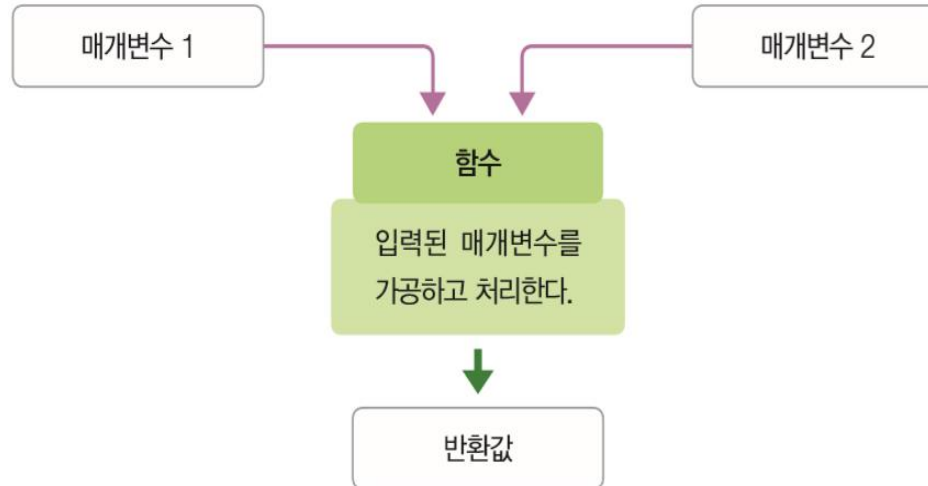
```
File Edit Selection View Go Run ... < ->
Untitled-1.py
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 coffee = 0
2
3 coffee = int(input("어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) "))
4
5 print()
6 print("#1. 뜨거운 물을 준비한다.");
7 print("#2. 종이컵을 준비한다.");
8
9 if coffee == 1 :
10     print("#3. 보통커피를 탄다.")
11 elif coffee == 2 :
12     print("#3. 설탕커피를 탄다.")
13 elif coffee == 3 :
14     print("#3. 블랙커피를 탄다.")
15 else :
16     print("#3. 아무거나 탄다.\n")
17
18 print("#4. 물을 붓는다.");
19 print("#5. 스푼으로 젓는다.");
20 print()
21 print("손님~ 커피 여기 있습니다.");
22
23 # 두번째 손님용 코드 반복(3~21행)
24
25 # 세번째 손님용 코드 반복(3~21행)
26
```

```
File Edit Selection View Go Run ... < ->
Untitled-1.py
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 ## 전역 변수 선언 부분 ##
2 coffee = 0
3
4 ## 함수 선언 부분 ##
5 > def coffee_machine(button) : ...
22
23 ## 메인 코드 부분 ##
24 coffee = int(input("A손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) "))
25 coffee_machine(coffee)
26 print("A손님~ 커피 여기 있습니다.")
27
28 coffee = int(input("B손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) "))
29 coffee_machine(coffee)
30 print("B손님~ 커피 여기 있습니다.")
31
32 coffee = int(input("C손님, 어떤 커피 드릴까요?(1:보통, 2:설탕, 3:블랙) "))
33 coffee_machine(coffee)
34 print("C손님~ 커피 여기 있습니다.")
35
```

Selection 02. 함수 기본

2. 함수의 형식과 활용

- 반복적으로 코딩해야 할 내용을 함수로 만들어 두면 필요할 때 사용이 가능하고, 함수로 구현하면 동일한 동작을 계속 하므로 내부의 내용이 바뀔 일이 없음



- 매개변수 (Parameter)를 입력받고 그 변수를 가공 및 처리해서 값을 반환

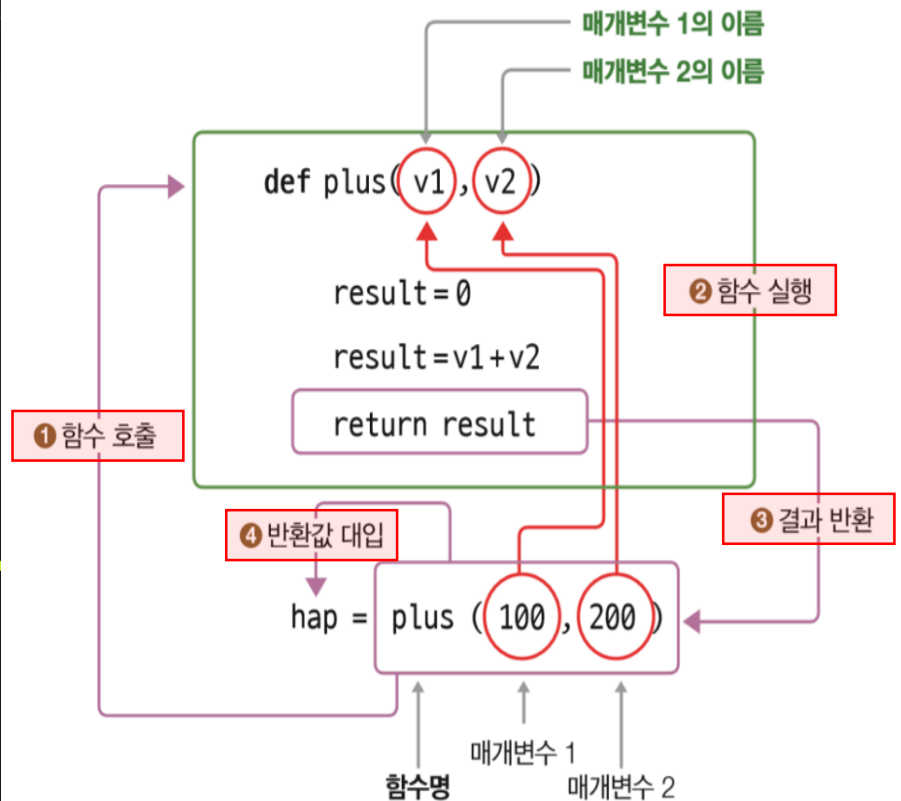
Selection 02. 함수 기본

2. 함수의 형식과 활용

- (예시) 두 정수를 입력 받아 합계를 반환하는 Plus() 함수구현의 예시

```
File Edit Selection View Go Run ...
Untitled-1.py
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  ## 함수 선언 부분 ##
2  def plus(v1, v2) :
3      result = 0
4      result = v1 + v2
5      return result
6
7  ## 전역 변수 선언 부분 ##
8  hap = 0
9
10 ## 메인 코드 부분 ##
11 hap = plus(100, 200)
12 print("100과 200의 plus() 함수 결과는 %d" % hap)
13

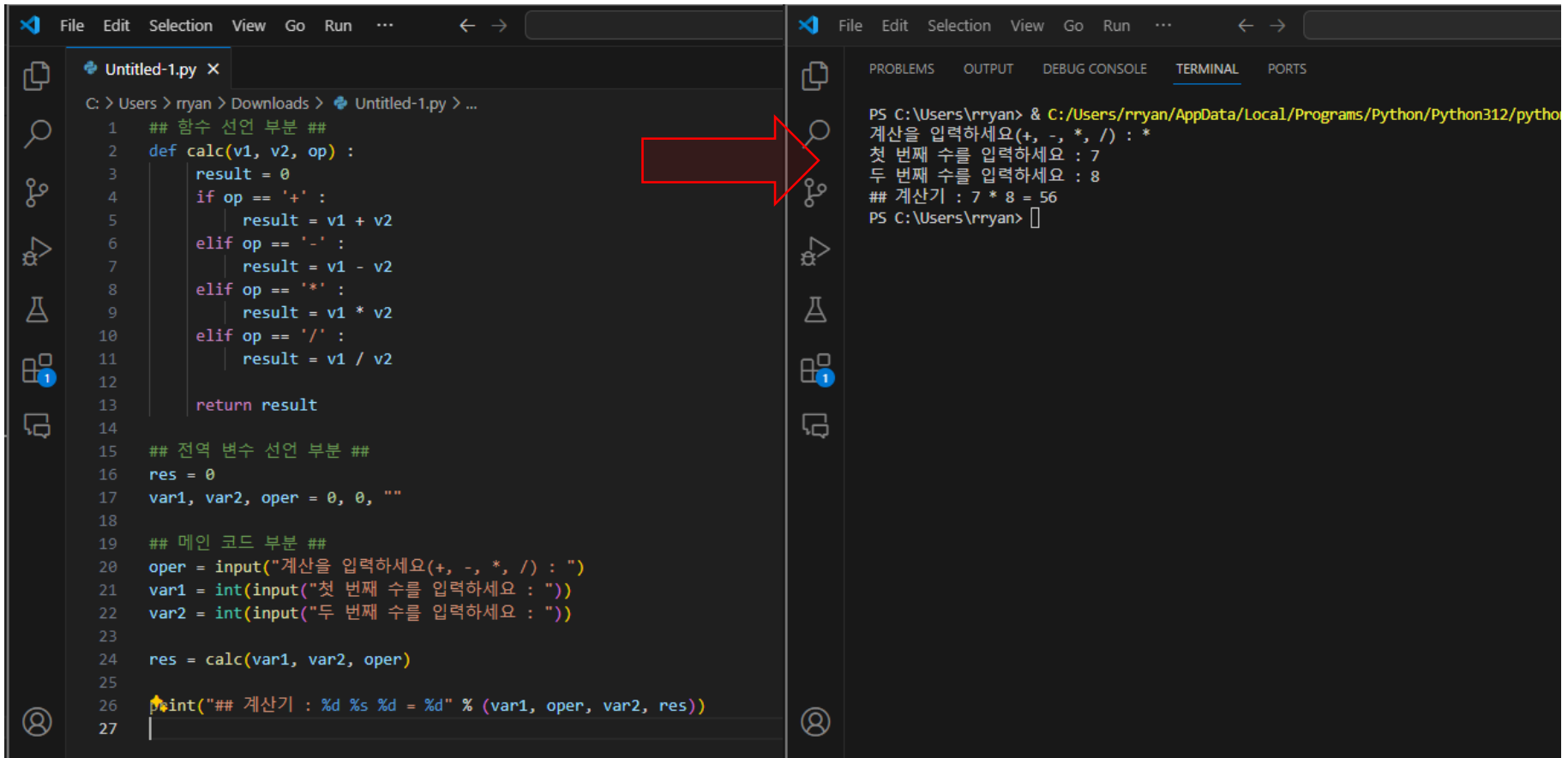
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
100과 200의 plus() 함수 결과는 300
PS C:\Users\rryan>
```



Selection 02. 함수 기본

2. 함수의 형식과 활용

- (예시) 덧셈, 뺄셈, 곱셈, 나눗셈을 하는 계산기 함수구현의 예시



```
File Edit Selection View Go Run ...  
Untitled-1.py X  
C: > Users > rryan > Downloads > Untitled-1.py > ...  
1  ## 함수 선언 부분 ##  
2  def calc(v1, v2, op) :  
3      result = 0  
4      if op == '+' :  
5          result = v1 + v2  
6      elif op == '-' :  
7          result = v1 - v2  
8      elif op == '*' :  
9          result = v1 * v2  
10     elif op == '/' :  
11         result = v1 / v2  
12  
13     return result  
14  
15  ## 전역 변수 선언 부분 ##  
16  res = 0  
17  var1, var2, oper = 0, 0, ""  
18  
19  ## 메인 코드 부분 ##  
20  oper = input("계산을 입력하세요(+, -, *, /) : ")  
21  var1 = int(input("첫 번째 수를 입력하세요 : "))  
22  var2 = int(input("두 번째 수를 입력하세요 : "))  
23  
24  res = calc(var1, var2, oper)  
25  
26  print("## 계산기 : %d %s %d = %d" % (var1, oper, var2, res))  
27
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python  
계산을 입력하세요(+, -, *, /) : *  
첫 번째 수를 입력하세요 : 7  
두 번째 수를 입력하세요 : 8  
## 계산기 : 7 * 8 = 56  
PS C:\Users\rryan>
```

Selection 02. 함수 기본

2. 함수의 형식과 활용

- (예시) 덧셈, 뺄셈, 곱셈, 나눗셈을 하는 계산기 함수구현의 예시

```
File Edit Selection View Go Run ...
Untitled-1.py X
C: > Users > rryan > Downloads > Untitled-1.py > calc
1  ## 함수 선언 부분 ##
2  def calc(v1, v2, op) :
3      (function) def calc(
4          v1: Any,
5          v2: Any,
6          op: Any
7      ) -> (Any | Literal[0])
8      이 함수는 두 수를 입력받아 사칙연산을 수행하는 함수입니다.
9
10     result = v1 ~ v2
11
12     elif op == '/' :
13         result = v1 / v2
14
15     return result
16
17 ## 전역 변수 선언 부분 ##
18 res = 0
19 var1, var2, oper = 0, 0, ""
20
21 ## 메인 코드 부분 ##
22 oper = input("계산을 입력하세요(+, -, *, /) : ")
23 var1 = int(input("첫 번째 수를 입력하세요 : "))
24 var2 = int(input("두 번째 수를 입력하세요 : "))
25
26 res = calc(var1, var2, oper)
27
28 print("## 계산기 : %d %s %d = %d" % (var1, oper, var2, res))
```

```
File Edit Selection View Go Run ...
Untitled-1.py X
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  ## 함수 선언 부분 ##
2  def calc(v1, v2, op) :
3      """ 이 함수는 두 수를 입력받아 사칙연산을 수행하는 함수입니다. """
4      result = 0
5      if op == '+' :
6          result = v1 + v2
7      elif op == '-' :
8          result = v1 - v2
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
계산을 입력하세요(+, -, *, /) : *
첫 번째 수를 입력하세요 : 7
두 번째 수를 입력하세요 : 8
## 계산기 : 7 * 8 = 56
Help on function calc in module __main__:
calc(v1, v2, op)
    이 함수는 두 수를 입력받아 사칙연산을 수행하는 함수입니다.
PS C:\Users\rryan> 
```

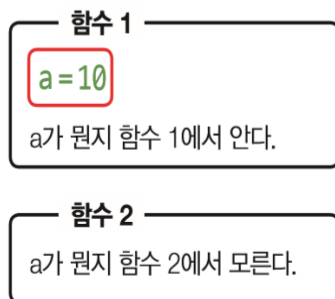
✓ 함수 독스트링 (docstring) 이라고 하며,
함수에 대한 설명을 넣을 수 있음

Selection 03. 지역 변수, 전역 변수

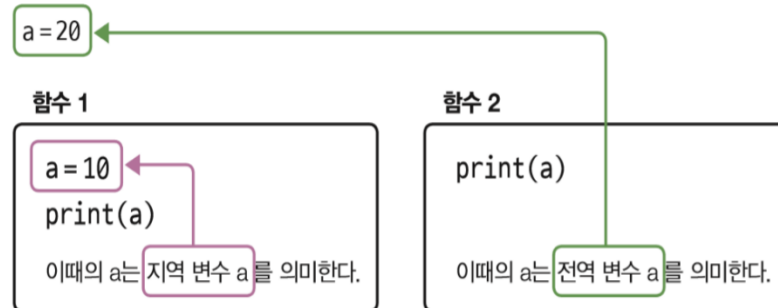
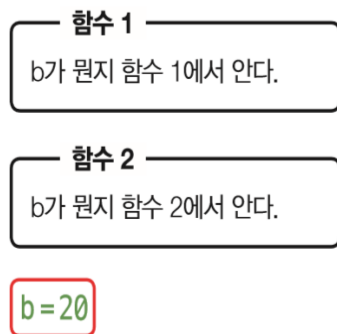
1. 지역 변수와 전역 변수의 이해

- 변수는 사용할 수 있는 범위에 따라 '함수 안에서만 사용'하는 지역 변수와, '함수 밖에서도 사용'할 수 있는 전역 변수로 구분
- 지역(local) 변수 : 한정된 지역에서만 사용
- 전역(global) 변수 : 프로그램 전체에서 사용

❶ 지역 변수의 생존 범위



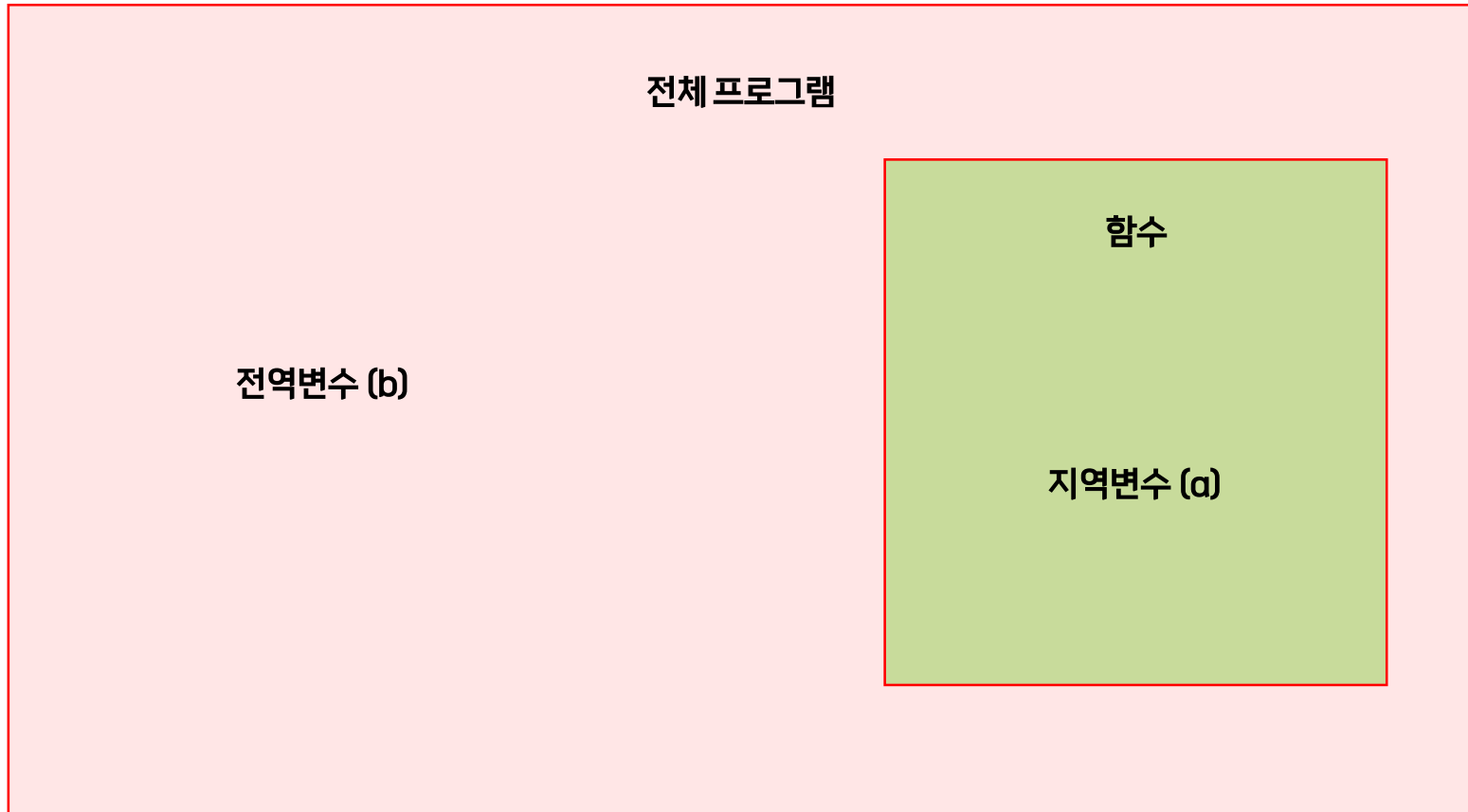
❷ 전역 변수의 생존 범위



- 지역 변수의 변수명과 전역 변수의 변수명이 같을 경우 지역 변수가 우선함

Selection 03. 지역 변수, 전역 변수

1. 지역 변수와 전역 변수의 이해



Selection 03. 지역 변수, 전역 변수

1. 지역 변수와 전역 변수의 이해

- (예시) 지역 변수와 전역 변수가 함께 구현되어 있는 예시

The image displays two side-by-side screenshots of a Python IDE (VS Code) showing a script with local and global variables. The script defines two functions, `func1()` and `func2()`, and a global variable `a`.

Left Screenshot (Successful Execution):

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      a = 10      # 지역 변수
4      print("func1()에서 a값 %d" % a)
5
6  def func2() :
7      print("func2()에서 a값 %d" % a)
8
9  ## 전역 변수 선언 부분 ##
10 a = 20          # 전역 변수
11
12 ## 메인 코드 부분 ##
13 func1()
14 func2()
15
```

The terminal output shows:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
func1()에서 a값 10
func2()에서 a값 20
PS C:\Users\rryan>
```

Right Screenshot (NameError):

The script is identical to the left screenshot, but the global variable `a` is commented out (line 10 is `#a = 20`). The terminal output shows a `NameError`:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
func1()에서 a값 10
Traceback (most recent call last):
  File "c:\Users\rryan\Downloads\Untitled-1.py", line 14, in <module>
    func2()
  File "c:\Users\rryan\Downloads\Untitled-1.py", line 7, in func2
    print("func2()에서 a값 %d" % a)
                                ^
NameError: name 'a' is not defined
PS C:\Users\rryan>
```

Selection 03. 지역 변수, 전역 변수

2. global 예약어

- 함수 안에서 사용되는 변수를 지역 변수 대신 전역 변수로 사용하고 싶을 때 활용

The image displays two side-by-side screenshots of a Python IDE, likely VS Code, illustrating the use of the `global` keyword.

Left Screenshot: The code defines `func1()` with a `global a` statement (highlighted with a red box) and `a = 10`. `func2()` prints the value of `a`. The terminal output shows `func1()에서 a값 10` and `func2()에서 a값 10` (the second line is highlighted with a red box).

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      global a  # 이 함수 안에서 a는 전역 변수
4      a = 10
5      print("func1()에서 a값 %d" % a)
6
7  def func2() :
8      print("func2()에서 a값 %d" % a)
9
10 ## 함수 변수 선언 부분 ##
11 a = 20  # 전역 변수
12
13 ## 메인 코드 부분 ##
14 func1()
15 func2()
```

Right Screenshot: The code defines `func1()` with a local variable `a = 10` and `func2()` which prints the value of `a`. The terminal output shows `func1()에서 a값 10` and `func2()에서 a값 20` (the second line is highlighted with a red box).

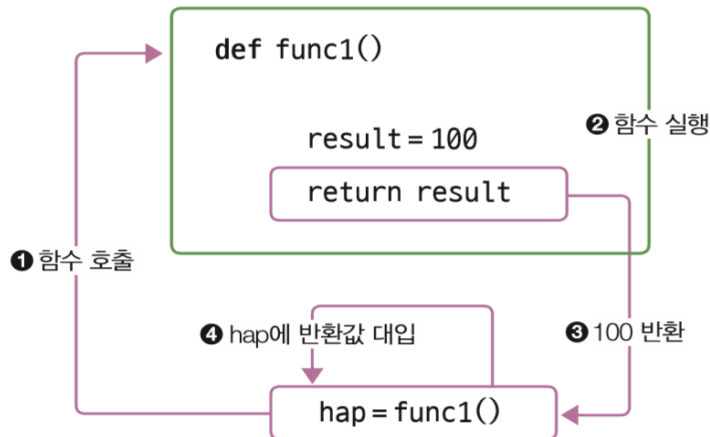
```
1  ## 함수 선언 부분 ##
2  def func1() :
3      a = 10  # 지역 변수
4      print("func1()에서 a값 %d" % a)
5
6  def func2() :
7      print("func2()에서 a값 %d" % a)
8
9  ## 전역 변수 선언 부분 ##
10 a = 20  # 전역 변수
11
12 ## 메인 코드 부분 ##
13 func1()
14 func2()
```

Selection 04. 함수의 반환값과 매개변수

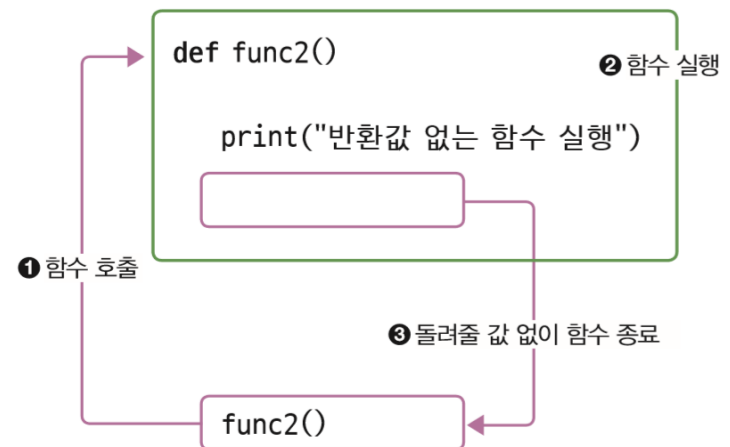
1. 함수의 반환값

- 함수는 실행 후 값을 돌려줄 때 (return)도 있고 돌려주지 않을 때 (not return)도 있음
- 함수는 반환값이 있는 함수와 반환값이 없는 함수로 나뉨
- Return에 반환값을 지정하지 않으면 None을 반환

〈반환값이 있는 함수〉



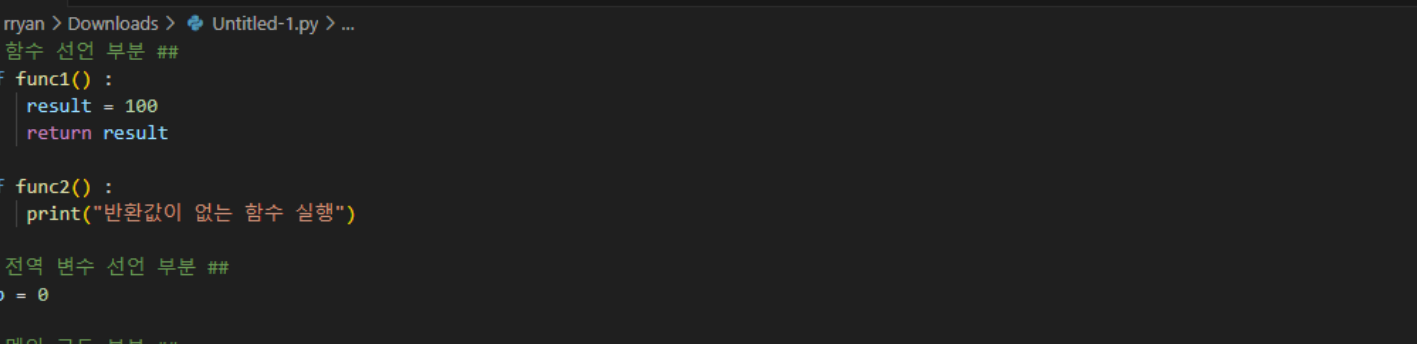
〈반환값이 없는 함수〉



Selection 04. 함수의 반환값과 매개변수

1. 함수의 반환값

- (예시) 함수의 반환값이 있는 경우(func1())와 반환값이 없는 경우(func2())의 예시



The screenshot shows a code editor with the following Python code:

```
1  ## 함수 선언 부분 ##
2  def func1() :
3      result = 100
4      return result
5
6  def func2() :
7      print("반환값이 없는 함수 실행")
8
9  ## 전역 변수 선언 부분 ##
10 hap = 0
11
12 ## 메인 코드 부분 ##
13 hap = func1()
14 print("func1()에서 돌려준 값 ==> %d" % hap)
15 func2()
16
```

The terminal output shows the execution of the script:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
func1()에서 돌려준 값 ==> 100
반환값이 없는 함수 실행
PS C:\Users\rryan>
```

Selection 04. 함수의 반환값과 매개변수

1. 함수의 반환값

- (예시) 함수의 반환값이 여러 개 있는 경우의 예시



The screenshot shows a Visual Studio Code editor window with a Python file named 'Untitled-1.py'. The code defines a function 'multi' that takes two arguments, 'v1' and 'v2', and returns a list containing their sum and difference. Below the function definition, there is a main block of code that calls 'multi(100, 200)' and prints the result.

```
1  ## 함수 선언 부분 ##
2  def multi(v1, v2) :
3      retList=[]          # 반환할 리스트
4      res1 = v1 + v2
5      res2 = v1 - v2
6      retList.append(res1)
7      retList.append(res2)
8      return retList
9
10 ## 전역 변수 선언 부분 ##
11 myList = []
12 hap, sub = 0, 0
13
14 ## 메인 코드 부분 ##
15 myList = multi(100, 200)
16 hap = myList[0]
17 sub = myList[1]
18 print("multi()에서 돌려준 값 ==> %d, %d" % (hap, sub))
19
```

The terminal output at the bottom shows the command to run the script and the resulting output:

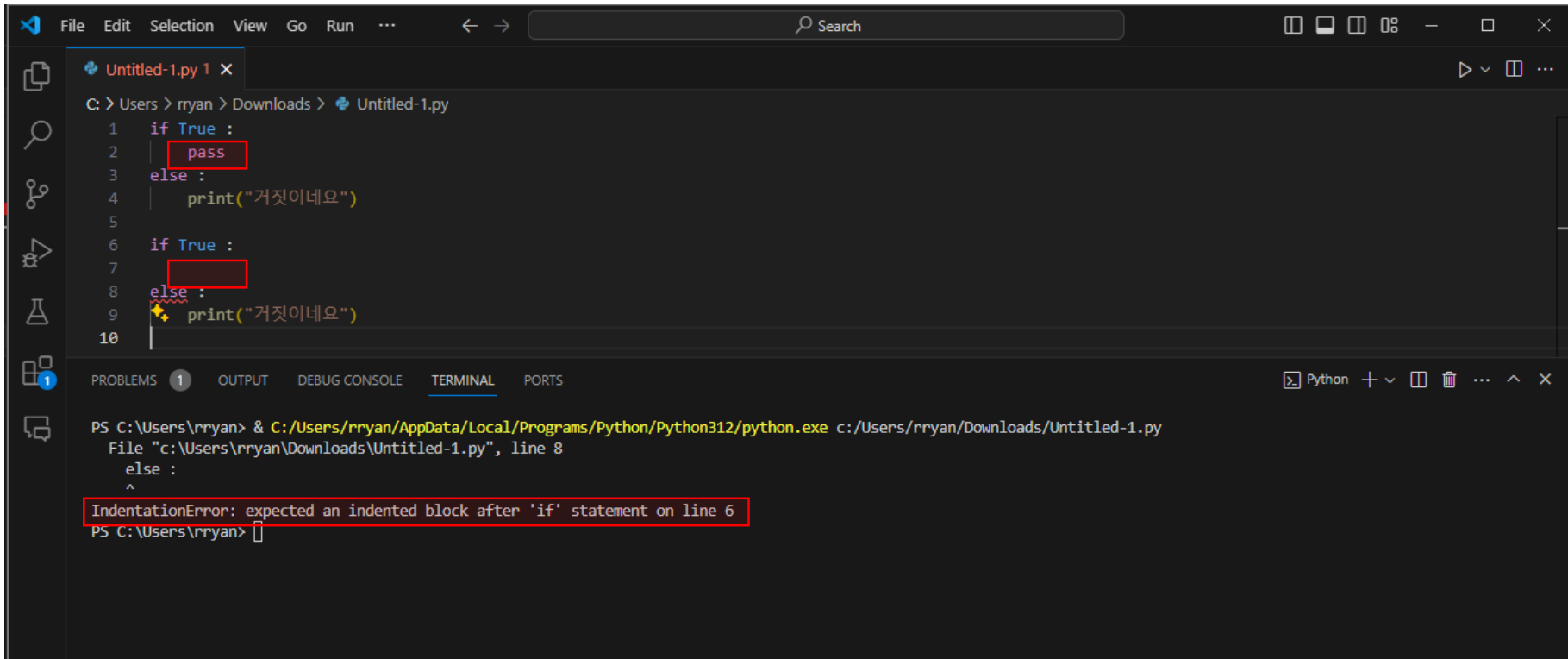
```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
multi()에서 돌려준 값 ==> 300, -100
PS C:\Users\rryan>
```

Selection 04. 함수의 반환값과 매개변수

1. 함수의 반환값

- Pass 예약어 : 함수를 구현할 때 선언을 해둔 뒤 구조 유지를 위해 내용을 비우는 경우

```
def myFunc() :  
    pass
```



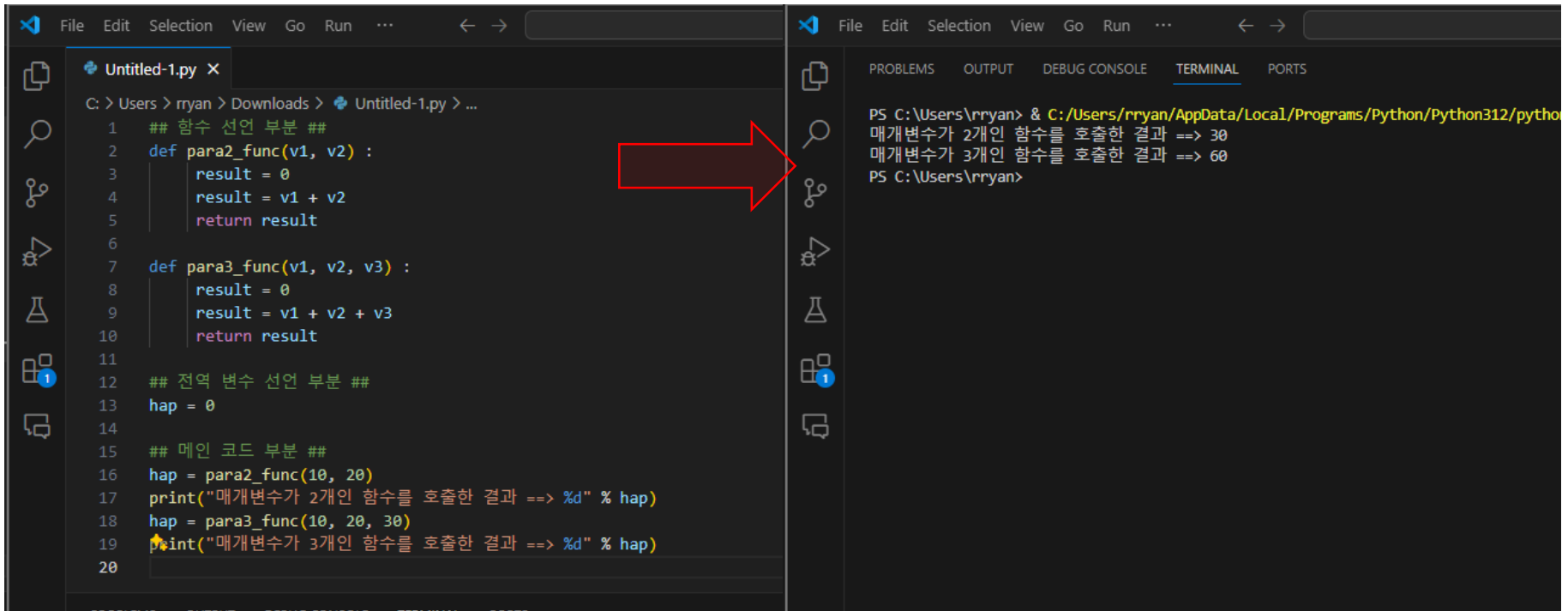
The screenshot shows a code editor with a file named 'Untitled-1.py'. The code contains two if-else blocks. The first block is correctly indented. The second block starts with 'if True :' on line 6, but the following 'else :' on line 8 is not indented, which causes an 'IndentationError'. The terminal at the bottom shows the command to run the file and the resulting error message: 'IndentationError: expected an indented block after 'if' statement on line 6'.

```
File Edit Selection View Go Run ... Search  
Untitled-1.py 1 X  
C:\Users\rryan> Downloads > Untitled-1.py  
1 if True :  
2     pass  
3 else :  
4     print("거짓이네요")  
5  
6 if True :  
7  
8 else :  
9     print("거짓이네요")  
10  
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS  
Python + - ... ^ X  
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py  
File "c:\Users\rryan\Downloads\Untitled-1.py", line 8  
    else :  
    ^  
IndentationError: expected an indented block after 'if' statement on line 6  
PS C:\Users\rryan>
```

Selection 04. 함수의 반환값과 매개변수

2. 함수의 매개변수 전달

- 함수의 매개변수 전달 방법은 매개변수의 개수를 지정해 전달하는 방법과, 기본값을 설정해 놓고 전달하는 방법, 개수를 지정하지 않고 전달하는 방법으로 구분
- 매개변수의 개수를 지정해 전달하는 방법

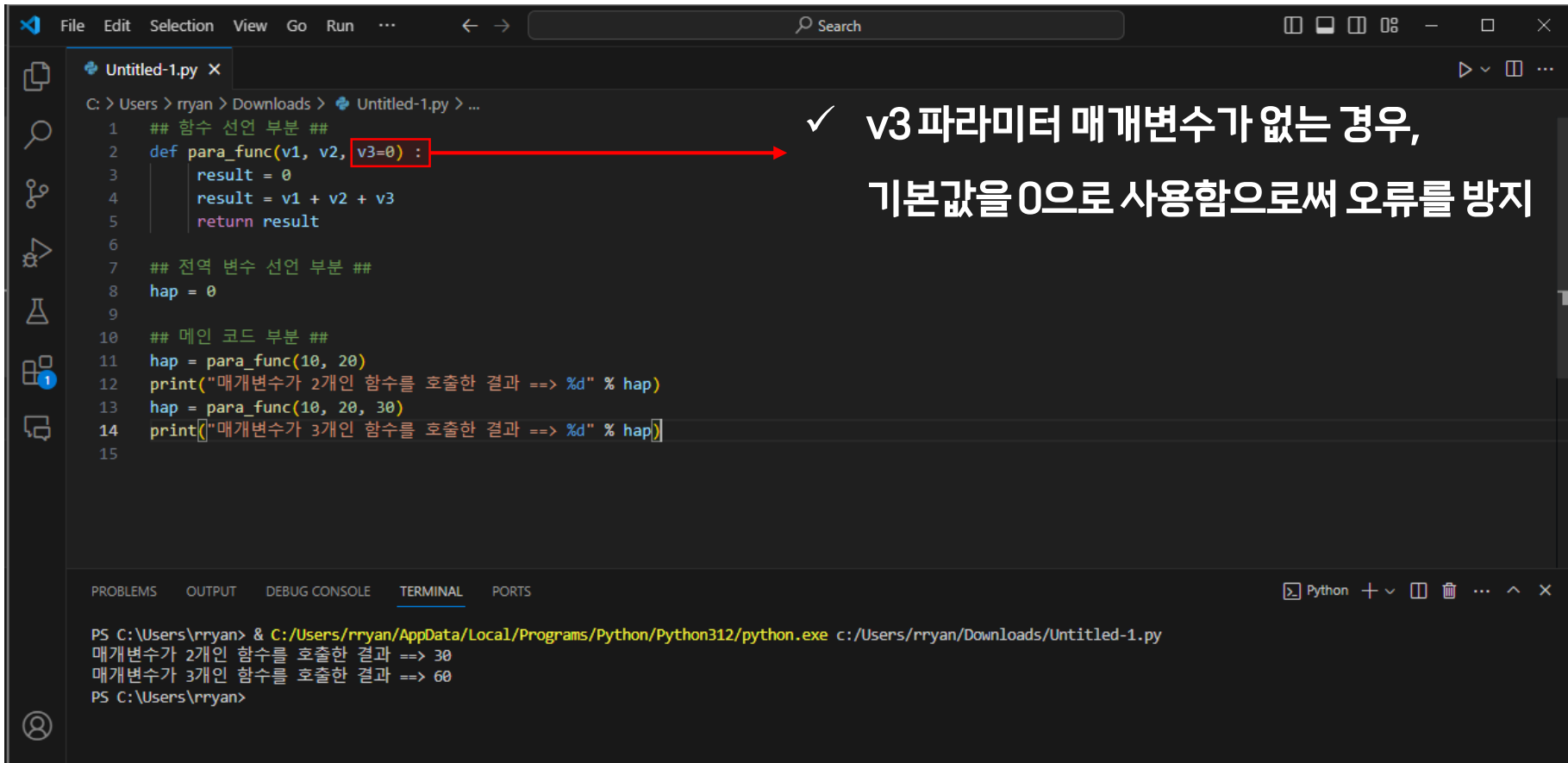


```
File Edit Selection View Go Run ... < ->
Untitled-1.py x
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  ## 함수 선언 부분 ##
2  def para2_func(v1, v2) :
3      result = 0
4      result = v1 + v2
5      return result
6
7  def para3_func(v1, v2, v3) :
8      result = 0
9      result = v1 + v2 + v3
10     return result
11
12     ## 전역 변수 선언 부분 ##
13     hap = 0
14
15     ## 메인 코드 부분 ##
16     hap = para2_func(10, 20)
17     print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
18     hap = para3_func(10, 20, 30)
19     print("매개변수가 3개인 함수를 호출한 결과 ==> %d" % hap)
20
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
매개변수가 2개인 함수를 호출한 결과 ==> 30
매개변수가 3개인 함수를 호출한 결과 ==> 60
PS C:\Users\rryan>
```

Selection 04. 함수의 반환값과 매개변수

2. 함수의 매개변수 전달

- 매개변수에 기본값을 설정해 놓고 전달하는 방법



The screenshot shows a Python IDE with a file named 'Untitled-1.py'. The code defines a function 'para_func' with three parameters: 'v1', 'v2', and 'v3'. The parameter 'v3' has a default value of 0, indicated by a red box and an arrow pointing to a checkmark. The function calculates the sum of 'v1', 'v2', and 'v3' and returns the result. Below the function definition, there are two calls to 'para_func'. The first call uses '10' and '20' for 'v1' and 'v2' respectively, and the second call uses '10', '20', and '30'. The terminal at the bottom shows the execution results: the first call returns 30, and the second call returns 60.

```
1  ## 함수 선언 부분 ##
2  def para_func(v1, v2, v3=0) :
3      result = 0
4      result = v1 + v2 + v3
5      return result
6
7  ## 전역 변수 선언 부분 ##
8  hap = 0
9
10 ## 메인 코드 부분 ##
11 hap = para_func(10, 20)
12 print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
13 hap = para_func(10, 20, 30)
14 print("매개변수가 3개인 함수를 호출한 결과 ==> %d" % hap)
15
```

✓ v3 파라미터 매개변수가 없는 경우,
기본값을 0으로 사용함으로써 오류를 방지

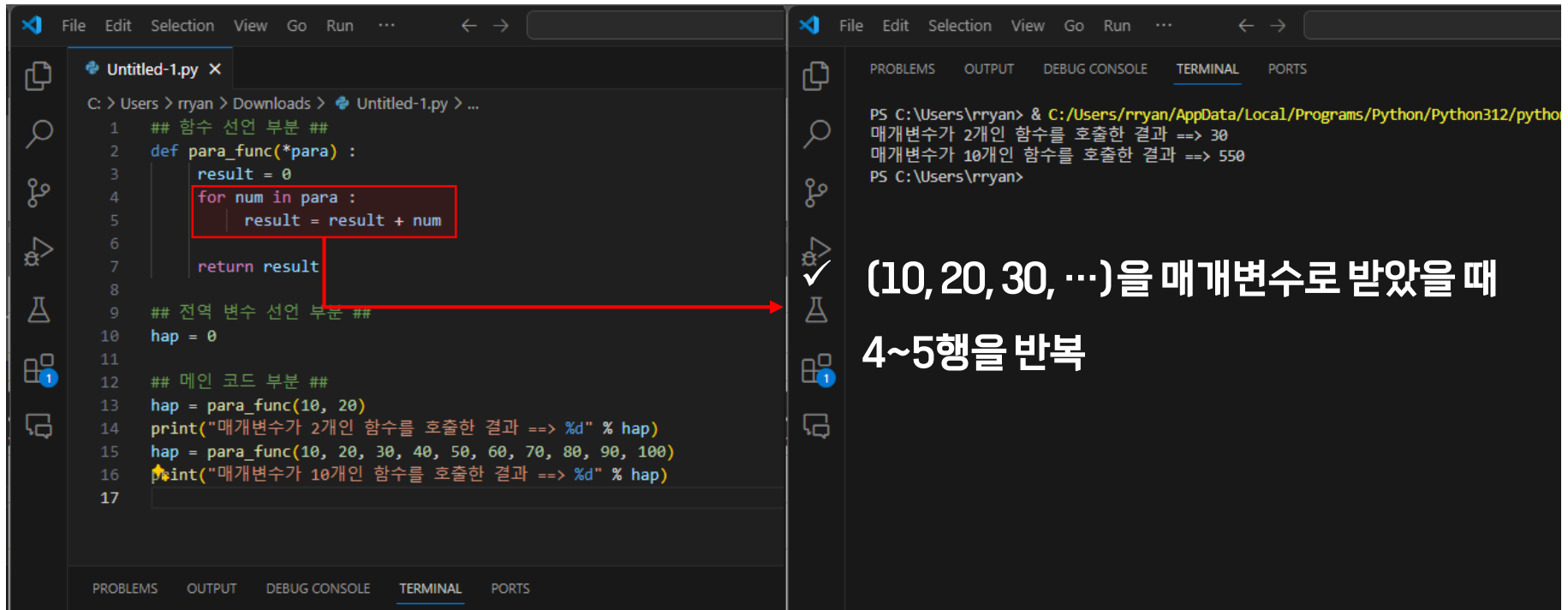
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
매개변수가 2개인 함수를 호출한 결과 ==> 30
매개변수가 3개인 함수를 호출한 결과 ==> 60
PS C:\Users\rryan>

Selection 04. 함수의 반환값과 매개변수

2. 함수의 매개변수 전달

- 매개변수의 개수를 지정하지 않고 전달하기 위해 가변 매개변수 방식을 사용
- 함수의 매개변수명 앞에 *를 붙이면 매개변수가 튜플/리스트 형식으로 처리하고, **을 붙이면 매개변수가 딕셔너리 형식으로 전달받아 처리



```
1  ## 함수 선언 부분 ##
2  def para_func(*para) :
3      result = 0
4      for num in para :
5          result = result + num
6
7      return result
8
9  ## 전역 변수 선언 부분 ##
10 hap = 0
11
12 ## 메인 코드 부분 ##
13 hap = para_func(10, 20)
14 print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
15 hap = para_func(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
16 print("매개변수가 10개인 함수를 호출한 결과 ==> %d" % hap)
17
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

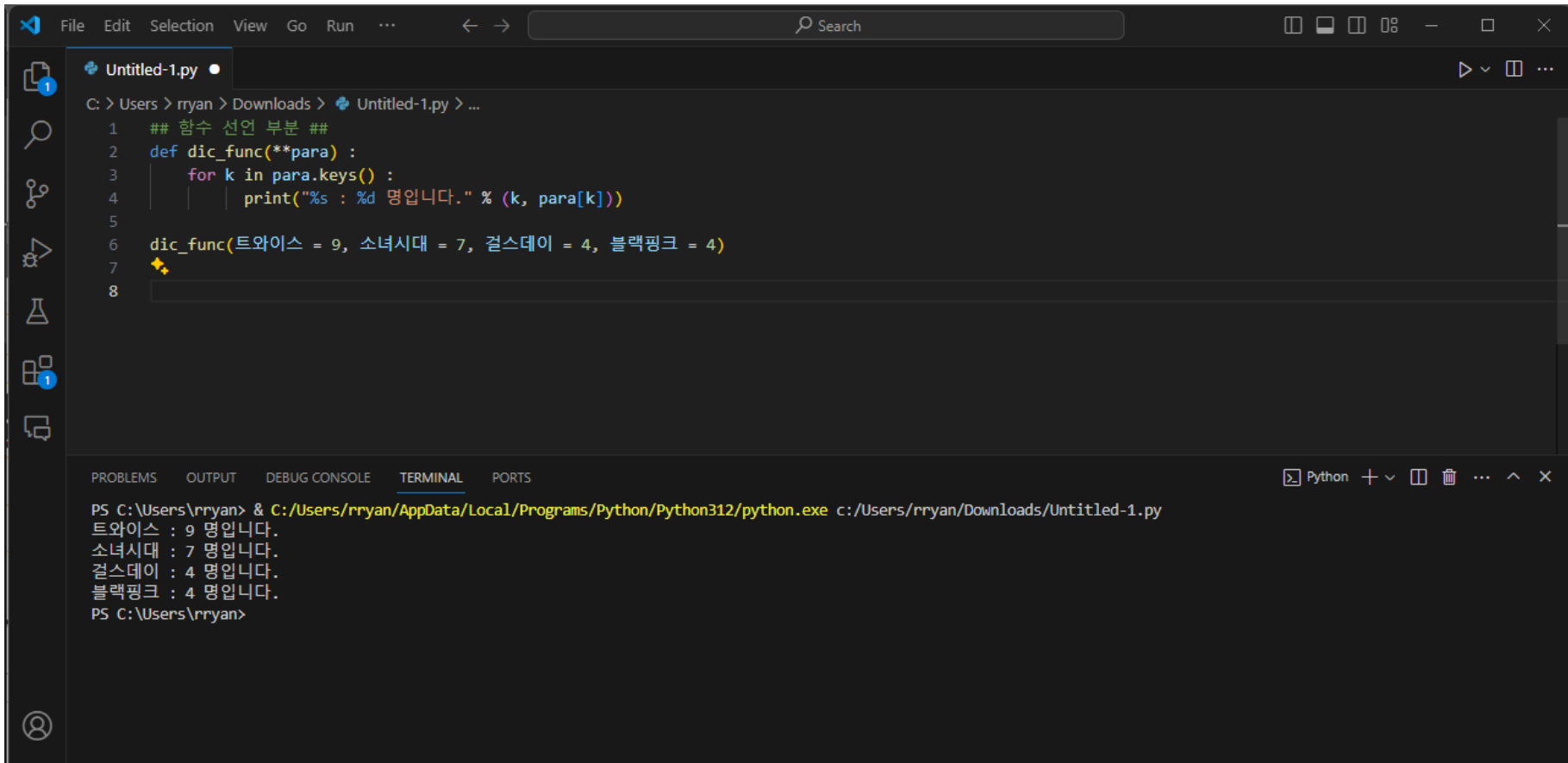
```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
매개변수가 2개인 함수를 호출한 결과 ==> 30
매개변수가 10개인 함수를 호출한 결과 ==> 550
PS C:\Users\rryan>
```

(10, 20, 30, ...)을 매개변수로 받았을 때
4~5행을 반복

Selection 04. 함수의 반환값과 매개변수

2. 함수의 매개변수 전달

- **을 붙인 딕셔너리 형식의 매개변수는 키=값 형식으로 처리



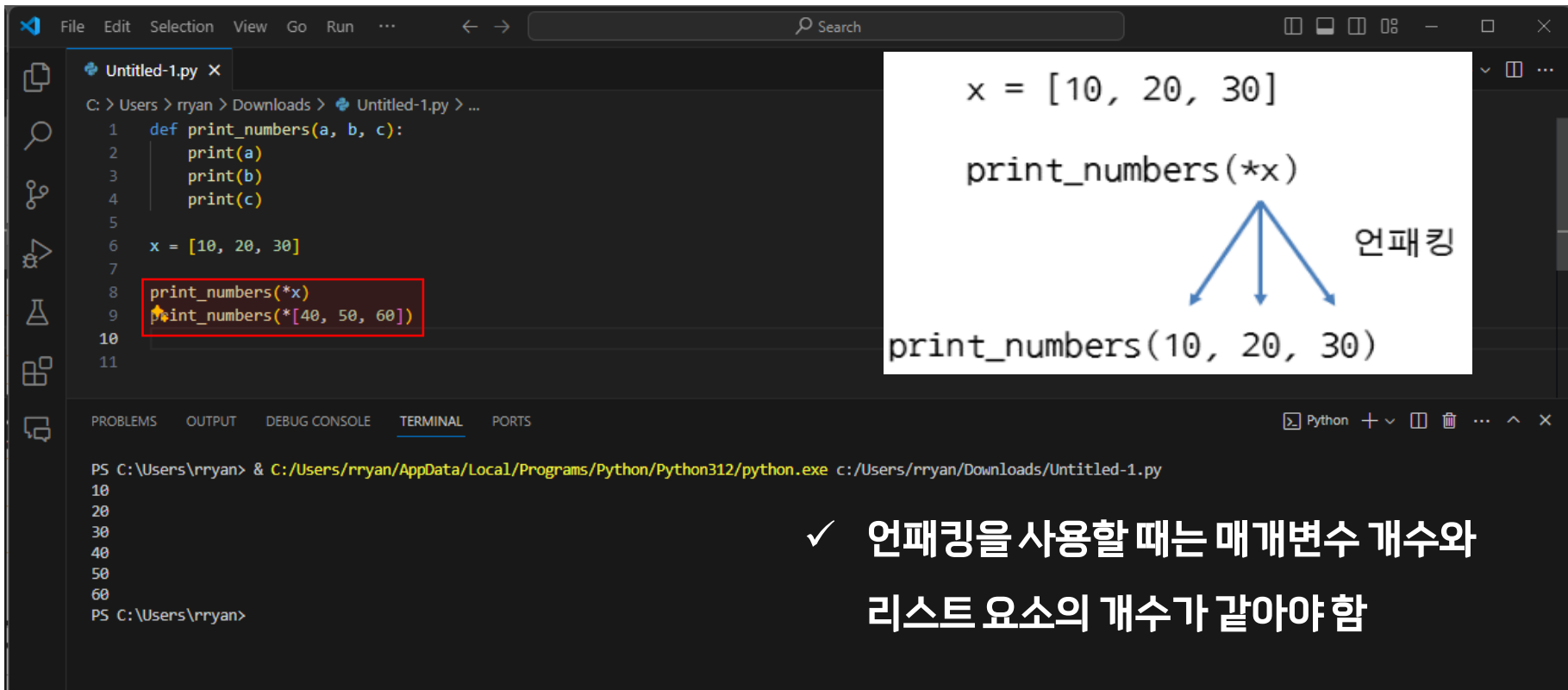
The screenshot shows a Visual Studio Code editor window with a file named 'Untitled-1.py'. The code defines a function `dic_func` that takes a dictionary `para` as an argument. It iterates over the keys of `para` and prints each key-value pair. The function is then called with a dictionary containing four items: '트와이스' (9), '소녀시대' (7), '걸스데이' (4), and '블랙핑크' (4). The terminal at the bottom shows the command to run the script and the resulting output, which matches the values in the dictionary.

```
File Edit Selection View Go Run ... Search
Untitled-1.py
C: > Users > rryan > Downloads > Untitled-1.py > ...
1  ## 함수 선언 부분 ##
2  def dic_func(**para) :
3      for k in para.keys() :
4          print("%s : %d 명입니다." % (k, para[k]))
5
6  dic_func(트와이스 = 9, 소녀시대 = 7, 걸스데이 = 4, 블랙핑크 = 4)
7  ✨
8
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - [icon] [icon] [icon] [icon] [icon] [icon]
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
트와이스 : 9 명입니다.
소녀시대 : 7 명입니다.
걸스데이 : 4 명입니다.
블랙핑크 : 4 명입니다.
PS C:\Users\rryan>
```

Selection 04. 함수의 반환값과 매개변수

2. 함수의 매개변수 전달

- 튜플/리스트 형식 앞에 *를 붙여 함수에 넣어주면 언패킹(unpacking)이 되고, 딕셔너리 형식은 앞에 **를 붙여 언패킹을 사용할 수 있음



```
def print_numbers(a, b, c):  
    print(a)  
    print(b)  
    print(c)  
  
x = [10, 20, 30]  
  
print_numbers(*x)  
print_numbers(*[40, 50, 60])
```

Diagram illustrating unpacking:

```
x = [10, 20, 30]  
print_numbers(*x)  
print_numbers(10, 20, 30)
```

언패킹

Terminal Output:

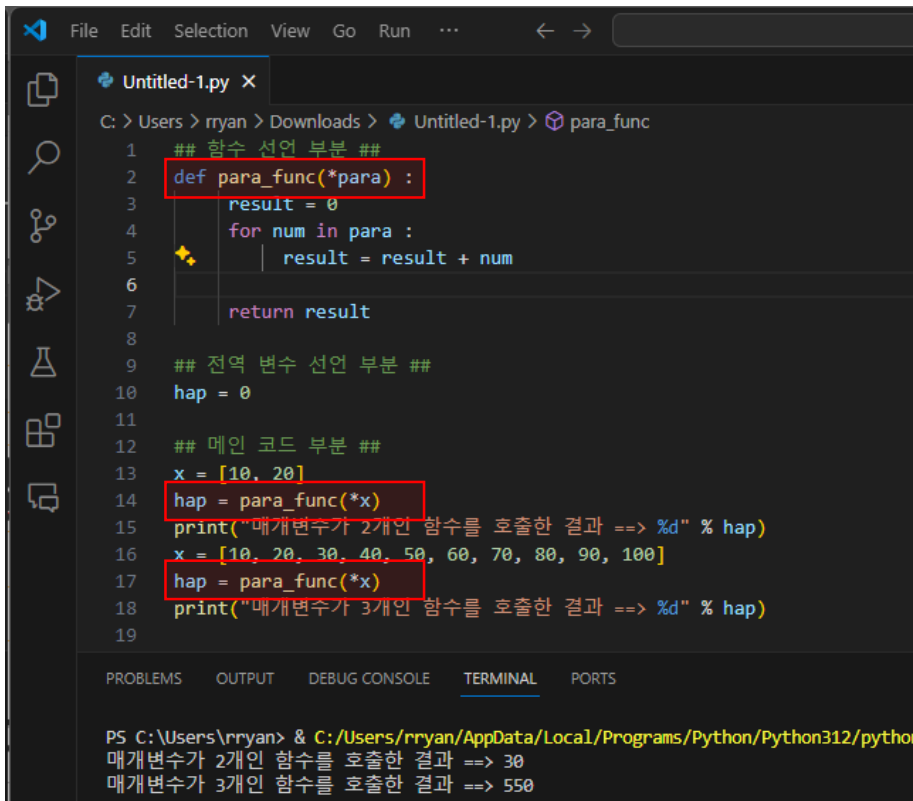
```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py  
10  
20  
30  
40  
50  
60  
PS C:\Users\rryan>
```

- ✓ 언패킹을 사용할 때는 매개변수 개수와 리스트 요소의 개수가 같아야 함

Selection 04. 함수의 반환값과 매개변수

2. 함수의 매개변수 전달

- (예시) 가변 매개변수 방식을 갖는 함수에 언패킹 한 인수를 넣는 예시
- 함수: `def para_func(*para)`, 인수: `para_func(*x)`

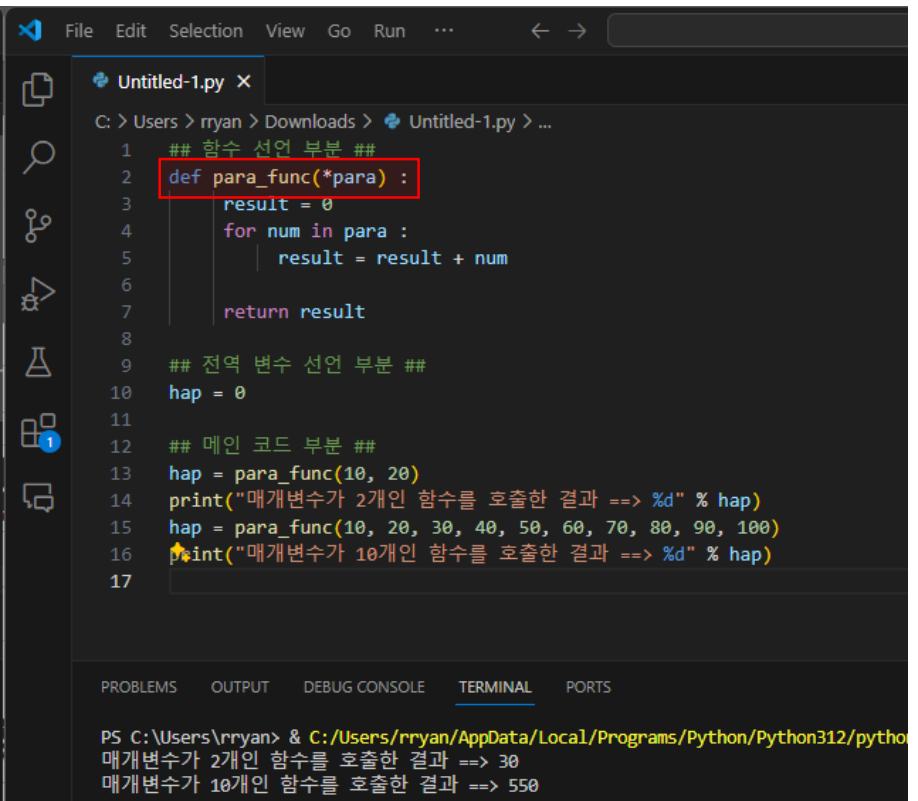


```
1  ## 함수 선언 부분 ##
2  def para_func(*para) :
3      result = 0
4      for num in para :
5          result = result + num
6
7      return result
8
9  ## 전역 변수 선언 부분 ##
10 hap = 0
11
12 ## 메인 코드 부분 ##
13 x = [10, 20]
14 hap = para_func(*x)
15 print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
16 x = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
17 hap = para_func(*x)
18 print("매개변수가 3개인 함수를 호출한 결과 ==> %d" % hap)
19
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe C:\Users\rryan\Downloads\Untitled-1.py

매개변수가 2개인 함수를 호출한 결과 ==> 30
매개변수가 3개인 함수를 호출한 결과 ==> 550



```
1  ## 함수 선언 부분 ##
2  def para_func(*para) :
3      result = 0
4      for num in para :
5          result = result + num
6
7      return result
8
9  ## 전역 변수 선언 부분 ##
10 hap = 0
11
12 ## 메인 코드 부분 ##
13 hap = para_func(10, 20)
14 print("매개변수가 2개인 함수를 호출한 결과 ==> %d" % hap)
15 hap = para_func(10, 20, 30, 40, 50, 60, 70, 80, 90, 100)
16 print("매개변수가 10개인 함수를 호출한 결과 ==> %d" % hap)
17
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe C:\Users\rryan\Downloads\Untitled-1.py

매개변수가 2개인 함수를 호출한 결과 ==> 30
매개변수가 10개인 함수를 호출한 결과 ==> 550

Selection 05. 모듈

1. 모듈의 생성과 사용

- 모듈 (Module)은 함수의 집합으로, 구현한 함수를 다른 프로그램에서 사용하고자 할 때
 임포트 함으로써 사용이 가능

The screenshot displays the Visual Studio Code interface with two Python files, `Module1.py` and `A.py`, and a terminal window. `Module1.py` contains three functions: `func1()`, `func2()`, and `func3()`, each printing a message. `A.py` imports `Module1` and calls `Module1.func1()`, `Module1.func2()`, and `Module1.func3()`. The terminal shows the output of these functions being executed. A diagram on the right illustrates the module relationship: `Module1.py` (containing function declarations) is imported by `A.py` and `B.py` (both containing function calls).

```
Module1.py
1 def func1():
2     print("Module1.py의 func1() 함수가 호출됨")
3
4 def func2():
5     print("Module1.py의 func2() 함수가 호출됨")
6
7 def func3():
8     print("Module1.py의 func3() 함수가 호출됨")
9

A.py
1 import Module1
2
3 ## 메인 코드 부분 ##
4
5 Module1.func1()
6 Module1.func2()
7 Module1.func3()
```

Diagram illustrating module usage:

- Module1.py** (Module):
 - func1() 함수 선언
 - func2() 함수 선언
 - func3() 함수 선언
- A.py** (User Program):
 - import Module1
 - func1() 함수 호출
 - func2() 함수 호출
 - func3() 함수 호출
- B.py** (User Program):
 - import Module1
 - func1() 함수 호출
 - func2() 함수 호출
 - func3() 함수 호출

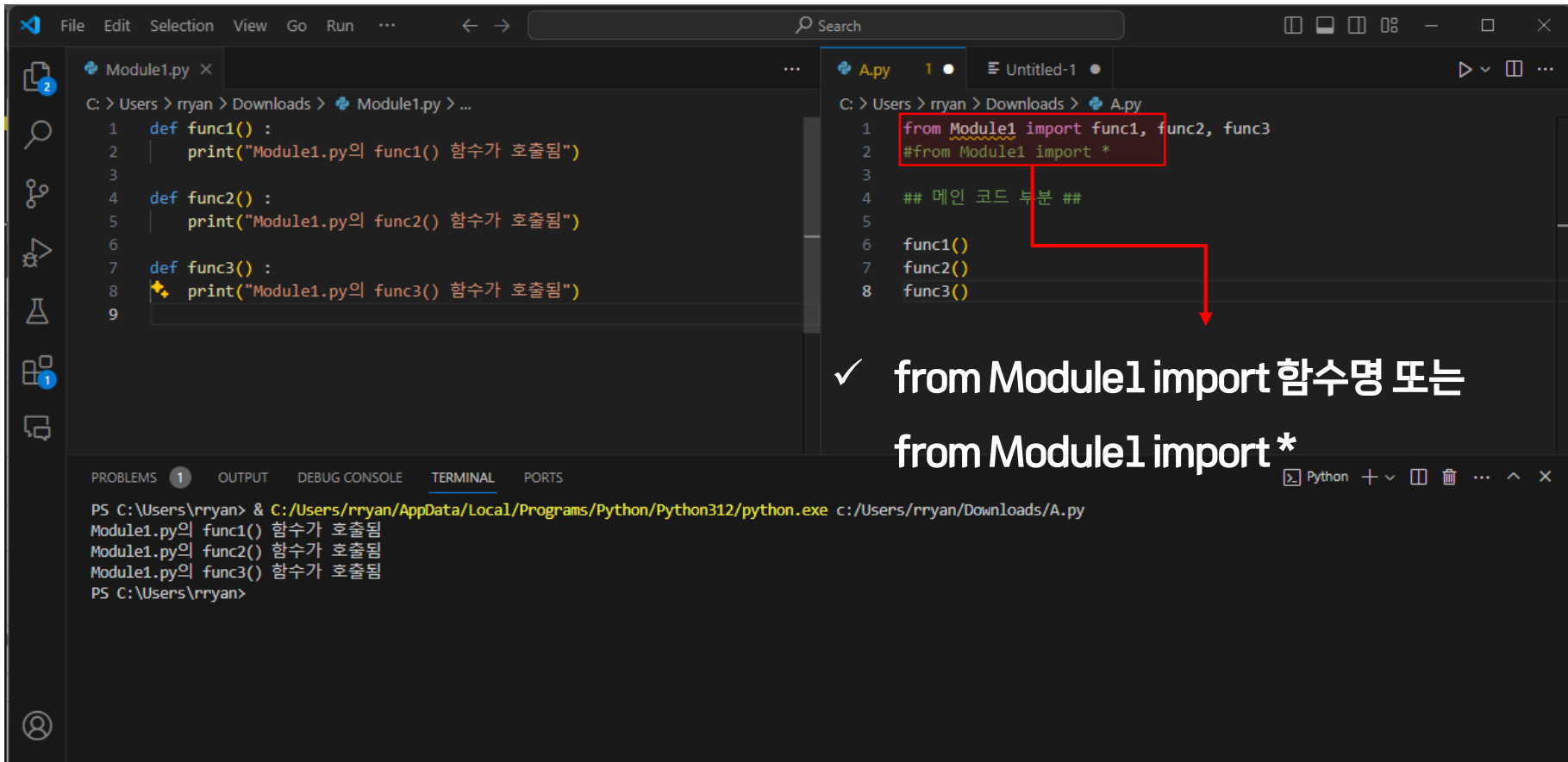
Terminal Output:

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:
Module1.py의 func1() 함수가 호출됨
Module1.py의 func2() 함수가 호출됨
Module1.py의 func3() 함수가 호출됨
PS C:\Users\rryan>
```

Selection 05. 모듈

1. 모듈의 생성과 사용

- Module1.함수명() 의 Module 형식을 생략하기 위해선 1행의 형식변경이 필요



```
Module1.py
1 def func1():
2     print("Module1.py의 func1() 함수가 호출됨")
3
4 def func2():
5     print("Module1.py의 func2() 함수가 호출됨")
6
7 def func3():
8     print("Module1.py의 func3() 함수가 호출됨")
9

A.py
1 from Module1 import func1, func2, func3
2 #from Module1 import *
3
4 ## 메인 코드 부분 ##
5
6 func1()
7 func2()
8 func3()
```

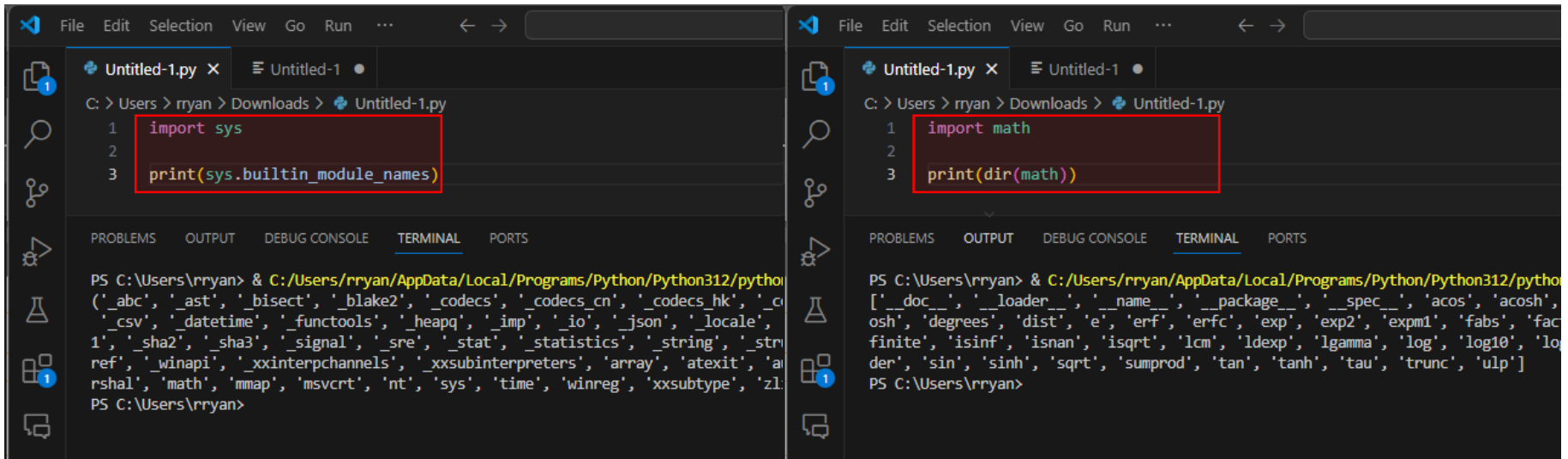
✓ from Module1 import 함수명 또는
from Module1 import *

```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/A.py
Module1.py의 func1() 함수가 호출됨
Module1.py의 func2() 함수가 호출됨
Module1.py의 func3() 함수가 호출됨
PS C:\Users\rryan>
```

Selection 05. 모듈

2. 모듈의 종류

- 표준 모듈, 사용자 정의 모듈, 서드 파티(Third Party) 모듈로 구분
 - 표준 모듈 : 파이썬에서 제공하는 모듈
 - 사용자 정의 모듈 : 직접 만들어서 사용하는 모듈
 - 서드 파티 모듈 : 파이썬이 아닌 외부 회사나 단체에서 제공하는 모듈
- ✓ (예시) 게임 개발 기능이 있는 pyGame, 윈도우 창을 제공하는 PyGTK 등



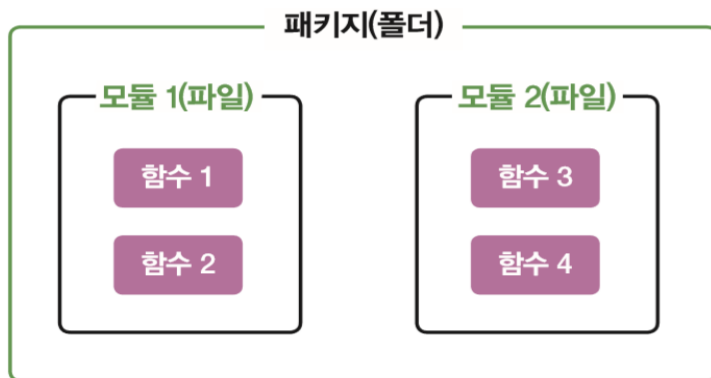
```
File Edit Selection View Go Run ... < ->
Untitled-1.py x
C: > Users > rryan > Downloads > Untitled-1.py
1 import sys
2
3 print(sys.builtin_module_names)
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
['_abc', '_ast', '_bisect', '_blake2', '_codecs', '_codecs_cn', '_codecs_hk', '_c
'_csv', '_datetime', '_functools', '_heapq', '_imp', '_io', '_json', '_locale',
1, '_sha2', '_sha3', '_signal', '_sre', '_stat', '_statistics', '_string', '_str
ref', '_winapi', '_xxinterchannels', '_xxsubinterpreters', 'array', 'atexit', 'a
rshal', 'math', 'mmap', 'msvcrt', 'nt', 'sys', 'time', 'winreg', 'xxsubtype', 'zl
PS C:\Users\rryan>
```

```
File Edit Selection View Go Run ... < ->
Untitled-1.py x
C: > Users > rryan > Downloads > Untitled-1.py
1 import math
2
3 print(dir(math))
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/pytho
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh',
osh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'fac
finite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'lo
der', 'sin', 'sinh', 'sqrt', 'sumprod', 'tan', 'tanh', 'tau', 'trunc', 'ulp']
PS C:\Users\rryan>
```

Selection 06. 함수의 심화 내용

1. 패키지

- 모듈이 하나의 *.py 파일 안에 함수가 여러 개 들어있는 것이라면, 패키지 (Package)는 여러 모듈을 모아 놓은 것으로 폴더의 형태로 나타내며 주제별로 분리할 때 주로 사용됨



- **임포트 형식**

```
from 패키지명.모듈명 import 함수명
```

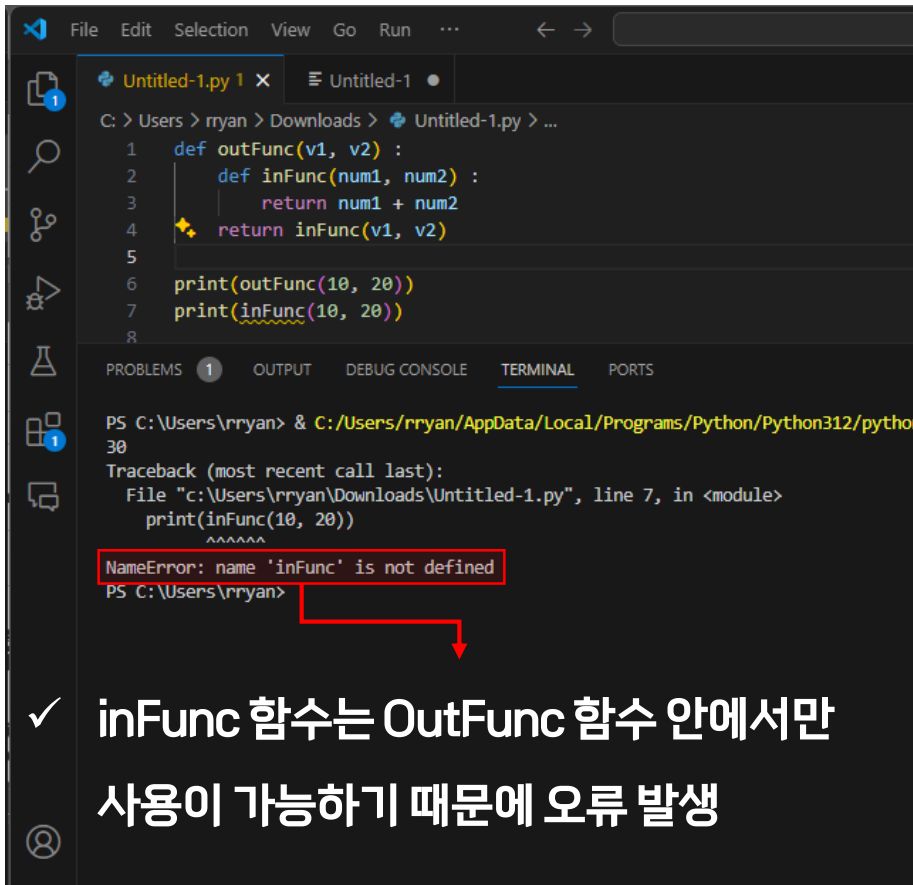
- **사용 예제**

```
from package.Module1 import *
```

Selection 06. 함수의 심화 내용

2. 내부 함수, 람다 함수 (lambda), 맵 함수 (map)

- 내부 함수 : 함수 안에 함수가 있는 형태
- 람다 함수 : 'lamda 매개변수 : 계산식'

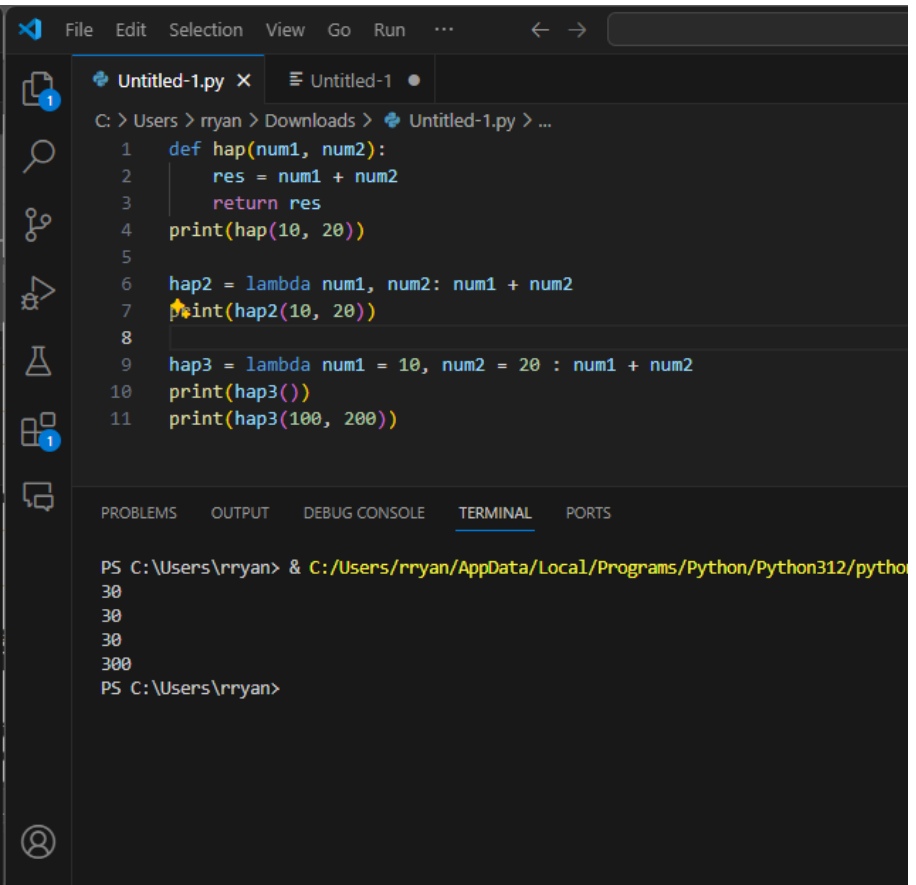


```
File Edit Selection View Go Run ...
Untitled-1.py x
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 def outFunc(v1, v2) :
2     def inFunc(num1, num2) :
3         return num1 + num2
4     return inFunc(v1, v2)
5
6 print(outFunc(10, 20))
7 print(inFunc(10, 20))
8

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
30
Traceback (most recent call last):
  File "c:\Users\rryan\Downloads\Untitled-1.py", line 7, in <module>
    print(inFunc(10, 20))
          ^^^^^
NameError: name 'inFunc' is not defined
PS C:\Users\rryan>
```

✓ inFunc 함수는 OutFunc 함수 안에서만
사용이 가능하기 때문에 오류 발생



```
File Edit Selection View Go Run ...
Untitled-1.py x
C: > Users > rryan > Downloads > Untitled-1.py > ...
1 def hap(num1, num2):
2     res = num1 + num2
3     return res
4 print(hap(10, 20))
5
6 hap2 = lambda num1, num2: num1 + num2
7 print(hap2(10, 20))
8
9 hap3 = lambda num1 = 10, num2 = 20 : num1 + num2
10 print(hap3())
11 print(hap3(100, 200))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python
30
30
30
300
PS C:\Users\rryan>
```

Selection 06. 함수의 심화 내용

2. 내부 함수, 람다 함수 (lambda), 맵 함수 (map)

- 람다 함수는 단독으로도 사용되지만, 맵 함수와 함께 더 많이 사용
- `map(함수명, 리스트)`는 리스트의 모든 내용을 하나씩 함수에 적용
- (예시) 리스트에 모두 10을 더하는 코드의 예시

```
File Edit Selection View Go Run ... Search
Untitled-1.py
C:\Users\rryan> Downloads > Untitled-1.py > ...
1 myList = [1, 2, 3, 4, 5]
2 def add10(num) :
3     return num + 10
4
5 for i in range(len(myList)) :
6     myList[i] = add10(myList[i])
7 print(myList)

Untitled-2.py
C:\Users\rryan> Downloads > Untitled-2.py > ...
1 myList = [1, 2, 3, 4, 5]
2 add10 = lambda num : num + 10
3 myList = list(map(add10, myList))
4 print(myList)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python + - ... ^ x

PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
[11, 12, 13, 14, 15]
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-2.py
[11, 12, 13, 14, 15]
PS C:\Users\rryan>
```

Selection 06. 함수의 심화 내용

3. 재귀 함수

- 재귀 함수 (Recursion Function)은 자신이 자신을 호출하는 함수

The screenshot shows a Python IDE with a file named 'Untitled-1.py'. The code defines a function `hello()` that prints 'Hello World' and calls itself. The terminal output shows the function being called multiple times, followed by a `Traceback (most recent call last):` error message: `RecursionError: maximum recursion depth exceeded`. A red box highlights the error message, and a red arrow points from it to the text '자기자신(hello())을 반복적으로 호출하다 최대 깊이를 초과하면 오류가 발생'.

def hello():
 print('Hello, world!')
 hello()
 ↙ hello()
 ↙ hello()
 ↙ hello()
 ↙ hello()
 ↙ hello()
 ↙ hello()
 ↙ ... ← 최대 재귀 깊이를 초과하면 RecursionError가 발생함

재귀 깊이가 깊어짐

✓ 자기자신(hello())을 반복적으로 호출하다
최대 깊이를 초과하면 오류가 발생

Selection 06. 함수의 심화 내용

3. 재귀 함수

- 재귀 함수 (Recursion Function)은 자신이 자신을 호출하는 함수

The image shows a Python IDE window with a file named 'Untitled-1.py'. The code defines a recursive factorial function and prints its values for 1 through 10. The terminal output shows the results: 1, 2, 6, 24, 120, 3628800. To the right of the code, a diagram illustrates the recursive process. It shows a series of function calls: `factorial(5)` calls `factorial(4)`, which calls `factorial(3)`, which calls `factorial(2)`, which calls `factorial(1)`. The base case is reached when `n == 1`, returning 1. The return values are then propagated back up the call stack: `factorial(2)` returns 2, `factorial(3)` returns 6, `factorial(4)` returns 24, and `factorial(5)` returns 120. Two annotations are present: a blue circle with '2' and the text 'n과 반환값을 곱한 뒤 다시 반환' (After multiplying n and the return value, return again), and a blue circle with '1' and the text '재귀호출' (Recursive call).

```
def factorial(num):  
    if num <= 1:  
        return num  
    else:  
        return num * factorial(num-1)  
  
print(factorial(1))  
print(factorial(2))  
print(factorial(3))  
print(factorial(4))  
print(factorial(5))  
print(factorial(10))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

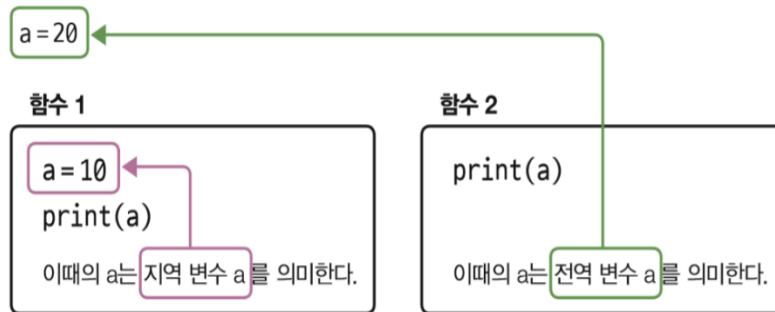
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe
1
2
6
24
120
3628800
PS C:\Users\rryan>

120
5 * 24
return n * factorial(n - 1);
4 * 6
return n * factorial(n - 1);
3 * 2
return n * factorial(n - 1);
2 * 1
return n * factorial(n - 1);
1
if (n == 1)
return 1;

2 n과 반환값을 곱한 뒤 다시 반환
1 재귀호출

Chapter 9. 요약

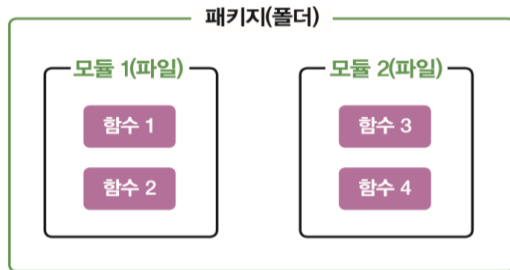
- ✓ 함수(Function)는 값을 넣으면 결과 값을 반환하는 기능
- ✓ 함수는 매개변수(Parameter)를 입력받아 매개변수를 처리한 후 값을 반환
- ✓ 지역 변수는 한정된 지역에서만 사용되는 변수이고, 전역 변수는 프로그램 전체에서 사용
- ✓ 지역 변수와 전역 변수가 공존할 때 접근 가능 범위는 아래와 같음



- ✓ 함수에서 어떤 계산이나 작동을 한 후 반환할 값이 있으면 return 반환 값 형식으로 표현
- ✓ 함수를 실행한 반환 값이 없을 때는 return 문을 생략할 수 있음

Chapter 9. 요약

- ✓ 함수의 매개변수를 전달하는 방법은 개수를 정확히 지정해서 전달하는 방법, 기본값을 설정해 놓고 전달하는 방법, 개수를 지정하지 않고 전달하는 방법으로 구분
- ✓ 가변 매개변수 방식을 지원하며 매개변수 앞에 *(튜플/리스트), **(딕셔너리)을 추가하여 사용
- ✓ 모듈은 함수의 집합과 같으며 별도의 파일에 함수들을 모아 놓은 것
- ✓ 모듈을 사용하려면 import 모듈명을, 함수를 호출할 때는 모듈명.함수명() 형식으로 사용하거나 from 모듈명 import 함수명 사용을 사용해서 함수명() 으로 호출
- ✓ 패키지는 여러 모듈을 모아 놓은 것으로 실제로는 폴더의 형태로 나타남



- ✓ 람다 함수는 함수를 한 줄로 간략하게 만들어 주며, 람다 함수와 map() 함수를 사용해 리스트에 함수 수식을 모두 한꺼번에 적용 가능

111001100110

감사합니다.

우창우

Dr.woo@chungbuk.ac.kr

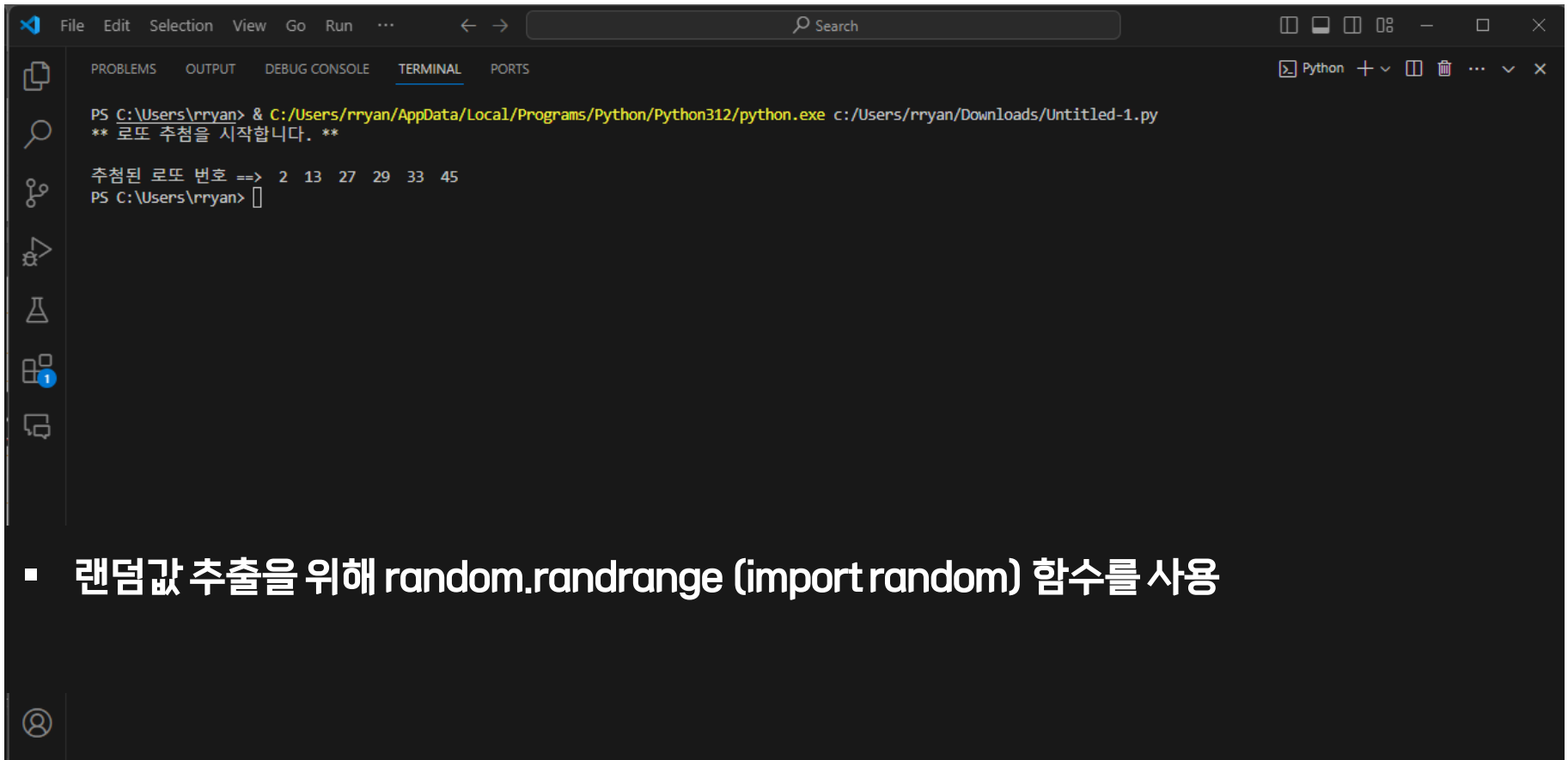
팀프로젝트를 위한 조원 명단

조 번호	프로젝트 주제	조장	조원
1조		조형준	고태경, 김다민
2조		김민혁	전영우, 김정민
3조	(게임) 텍스트 방탈출 게임	홍성진	김태영, 정세연
4조		이규민	우태현, 전수혁
5조		배정민	박상인, 서범교, 송설희
6조	(웹) 교내 학생 간 개인 중고 거래 사이트	박조현	김건우, 오다영
7조	(모바일_앱) 가계부 캘린더 앱	박주현	권정욱, 정현준
8조		김규현	김준후, 조윤정
9조	(모바일_앱) 학교 주변 맛집, 제휴 업체 등 위치를 표시해주는 지도 앱	신종환	신승우, 한강민
11조	(모바일_앱) 학교 Q&A 챗봇	한준영	고태영, 이관학, 육광민
12조	(앱) 수업 참여도 분석 프로그램	윤시훈	전준석, 김민경
13조		김준호	황지연, 이용희
14조	(게임) 게임개발	배수환	이한결, 신혜원
15조	(모바일_앱) 택시 함께 탈 사람 매칭 앱	박성범	이태정, 김민석

실습&과제 : (제출처) Dr.woo@chungbuk.ac.kr, (기한) 4.6(토) 까지

1. 로또 번호 추천

- 1~45의 숫자 중에서 6개를 뽑는 프로그램



```
PS C:\Users\rryan> & C:/Users/rryan/AppData/Local/Programs/Python/Python312/python.exe c:/Users/rryan/Downloads/Untitled-1.py
** 로또 추첨을 시작합니다. **

추첨된 로또 번호 ==> 2 13 27 29 33 45
PS C:\Users\rryan>
```

- 랜덤값 추출을 위해 `random.randrange (import random)` 함수를 사용