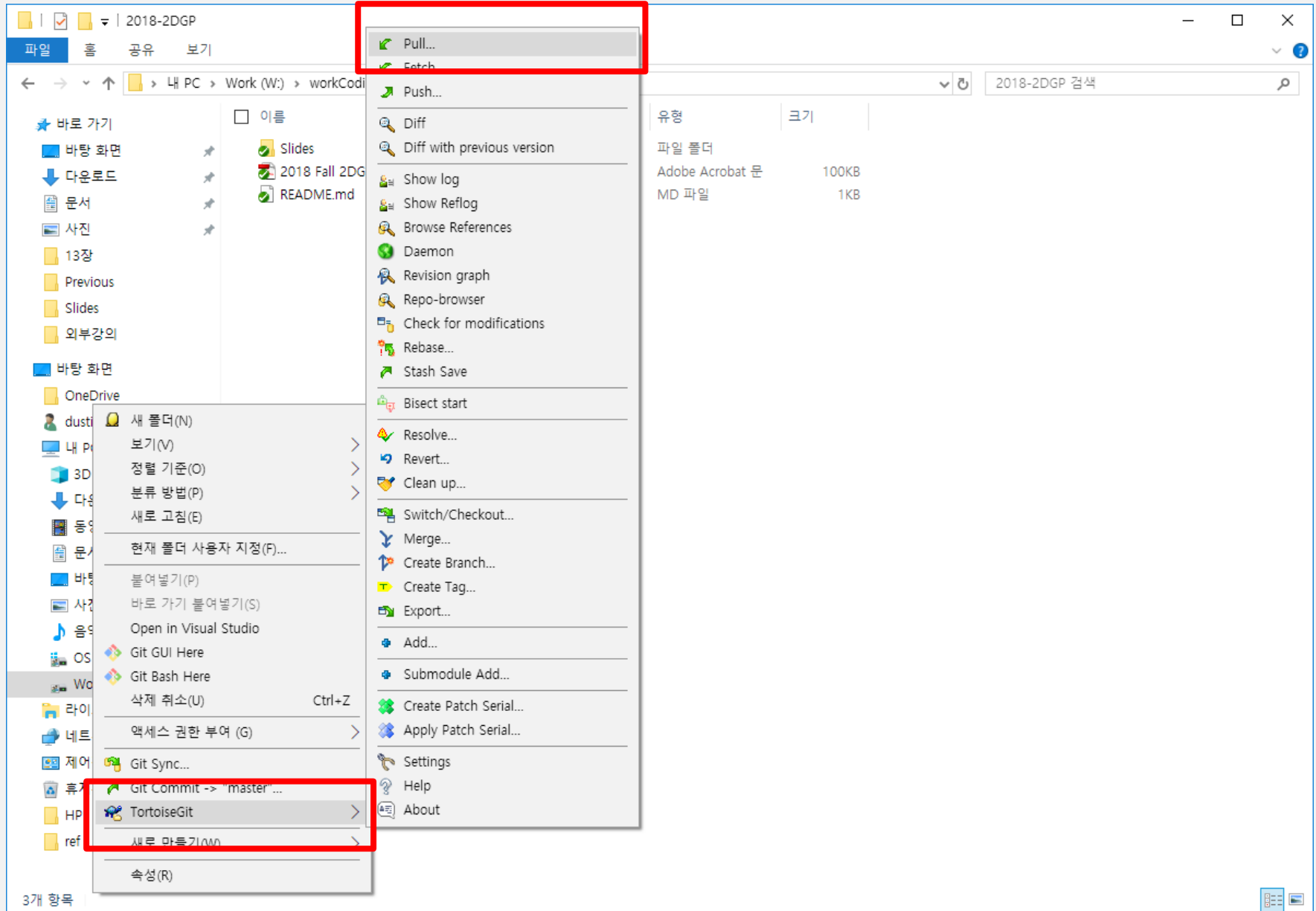
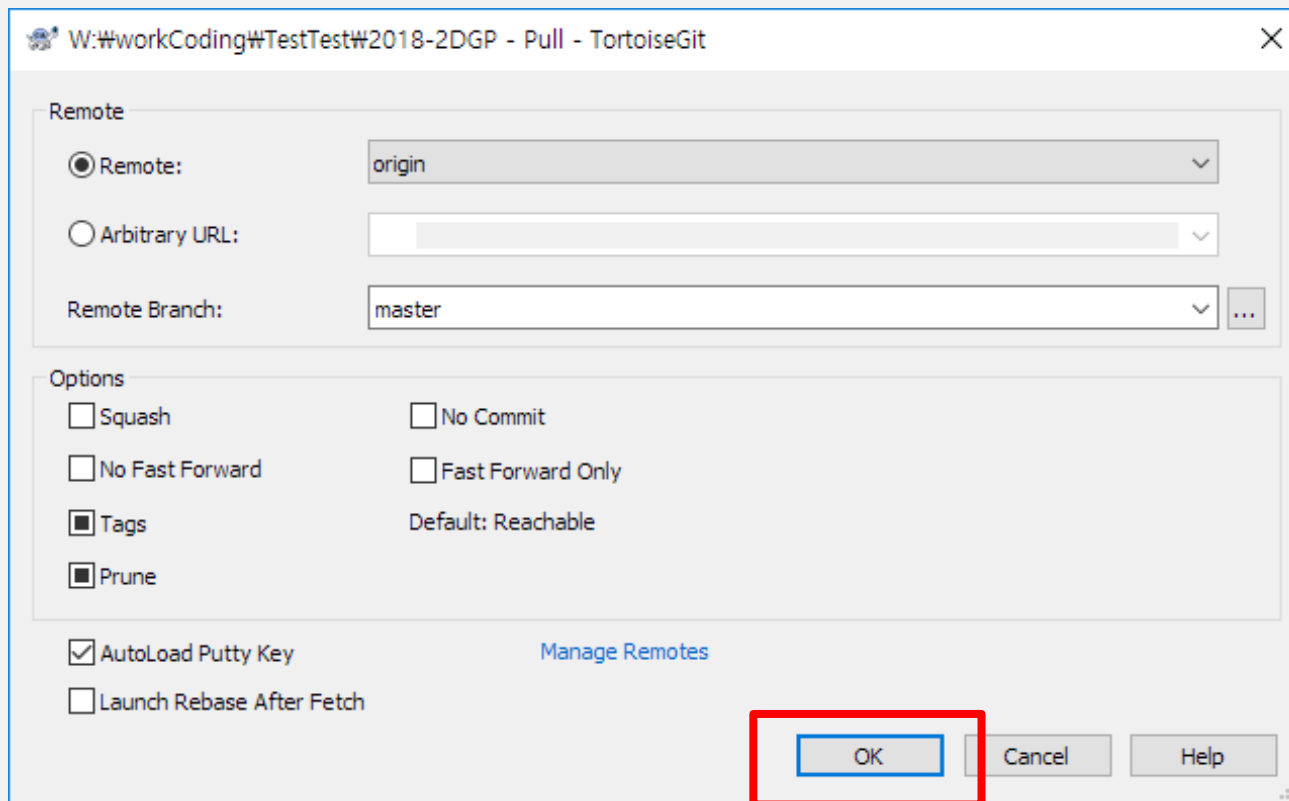


Lecture #1. 파이썬 기초 (2)

Git Pull – 서버에서 업데이트된 내용을 내려받을 때







remote: Compressing objects

```
git.exe pull --progress -v --no-rebase "origin"
```

```
POST git-upload-pack (437 bytes)
```

```
remote: Counting objects: 12, done.
```

```
remote: Compressing objects: 100% (10/10), done.
```

```
remote: Total 12 (delta 1), reused 12 (delta 1), pack-reused 0
```

```
From https://github.com/game-lecture/2018-2DGP
```

```
46e421a..bb5c854 master -> origin/master
```

```
Updating 46e421a..bb5c854
```

```
Fast-forward
```

Labs/Lab01/bus_fare.py	17 ++++++
Labs/Lab01/draw_circle.py	20 ++++++
Labs/Lab01/drunken_turtle.py	9 ++++++
Labs/Lab01/random_turtle.py	18 ++++++
Labs/Lab01/random_turtle_function.py	11 ++++++
Labs/Lab01/ten_lines.py	20 ++++++
Labs/Lab01/test_age.py	16 ++++++
Labs/Lab01/while_turtle.py	11 ++++++

```
8 files changed, 122 insertions(+)
```

```
create mode 100644 Labs/Lab01/bus_fare.py
```

```
create mode 100644 Labs/Lab01/draw_circle.py
```

```
create mode 100644 Labs/Lab01/drunken_turtle.py
```

```
create mode 100644 Labs/Lab01/random_turtle.py
```

```
create mode 100644 Labs/Lab01/random_turtle_function.py
```

```
create mode 100644 Labs/Lab01/ten_lines.py
```

```
create mode 100644 Labs/Lab01/test_age.py
```

```
create mode 100644 Labs/Lab01/while_turtle.py
```

```
Success (1187 ms @ 2018-09-03 오전 8:55:54)
```

Pulled Diff

Close

Abort

Turtle 모듈

- 펜을 가지고, 화면 위를 다니면서 그림을 그림.
- 전진, 후진, 회전, 원 그리기 등 다양하게 움직이면 그림을 그릴 수 있음.



펜을 물고 있는 거북이


모듈의 사용 문법

모듈을 사용하기 위해 수입(import)함.



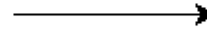
```
import turtle
```

```
turtle.forward(100)
```



turtle 이 갖고 있는 기능(함수, function)
을 이용하여, 그림을 그린다.

```
>>> import turtle  
>>> turtle.forward(100)
```

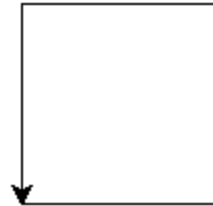


```
>>> turtle.reset()
```

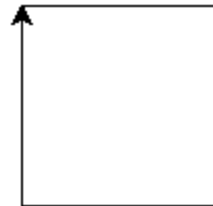


거북이의 기본 방향은 오른쪽임.

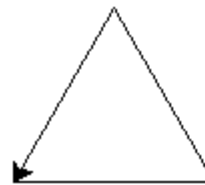

```
>>> turtle.forward(100)
>>> turtle.left(90)
>>> turtle.forward(100)
>>> turtle.left(90)
>>> turtle.forward(100)
>>> turtle.left(90)
>>> turtle.forward(100)
```



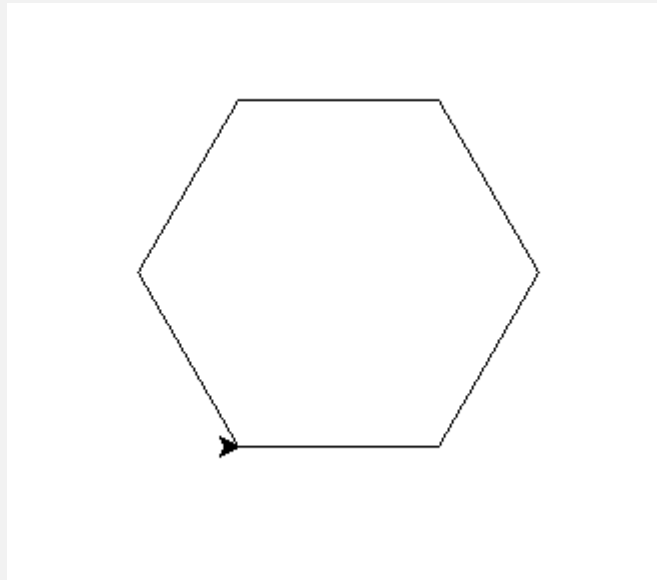
```
>>> turtle.reset()  
>>> turtle.forward(100)  
>>> turtle.right(90)  
>>> turtle.forward(100)  
>>> turtle.right(90)  
>>> turtle.forward(100)  
>>> turtle.right(90)  
>>> turtle.forward(100)
```



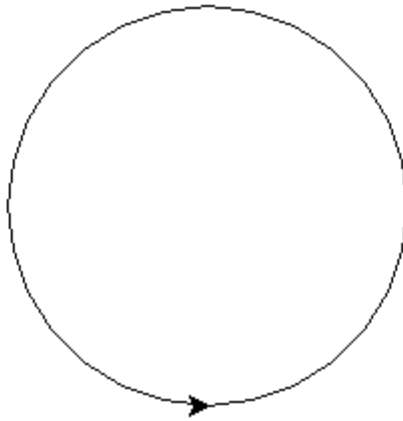
```
>>> turtle.forward(100)  
>>> turtle.left(120)  
>>> turtle.forward(100)  
>>> turtle.left(120)  
>>> turtle.forward(100)
```



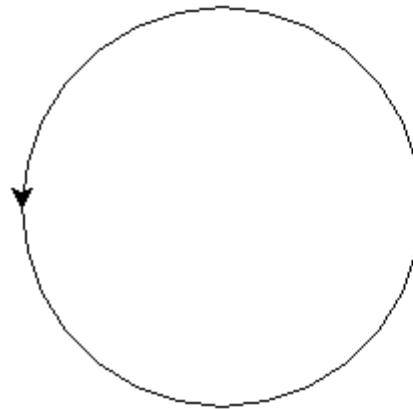
퀴즈 #1: 정육각형을 그려보자 !



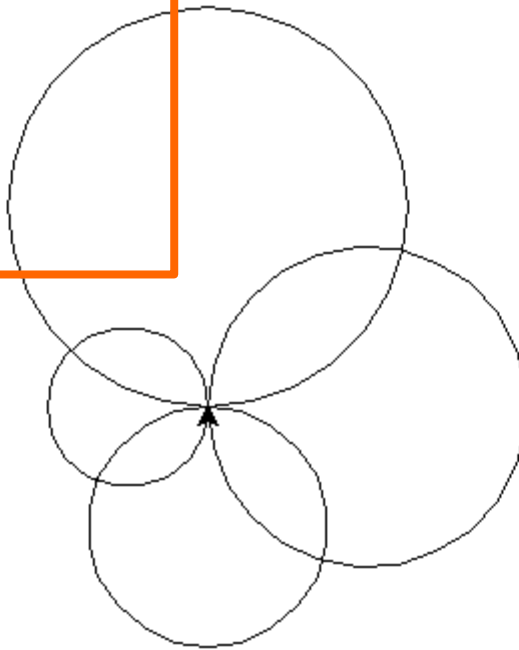
```
>>> turtle.circle(100)
```



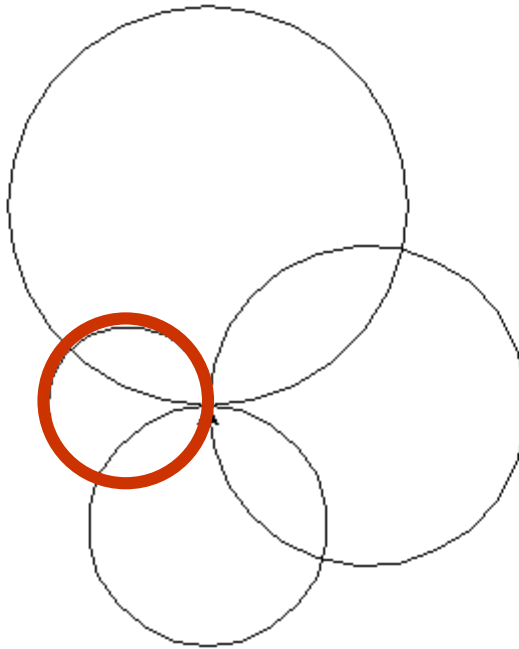
```
>>> turtle.right(90)  
>>> turtle.circle(100)
```



```
>>> turtle.circle(100)
>>> turtle.right(90)
>>> turtle.circle(80)
>>> turtle.right(90)
>>> turtle.circle(60)
>>> turtle.right(90)
>>> turtle.circle(40)
```



```
>>> turtle.updo()  
>>> turtle.undo()
```

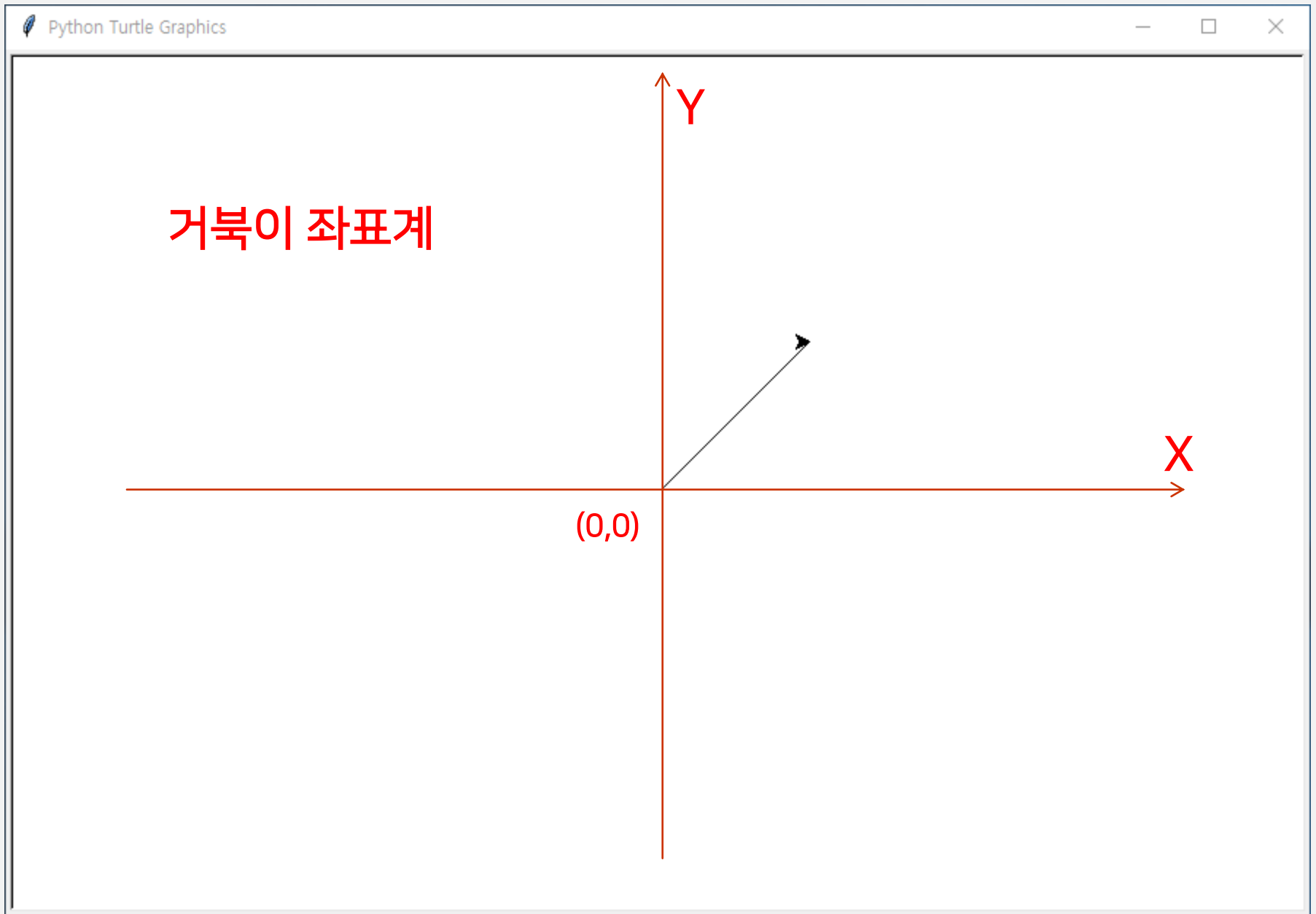


마지막에 그렸던 원이 없어짐.
이전 상태로 되돌아감.

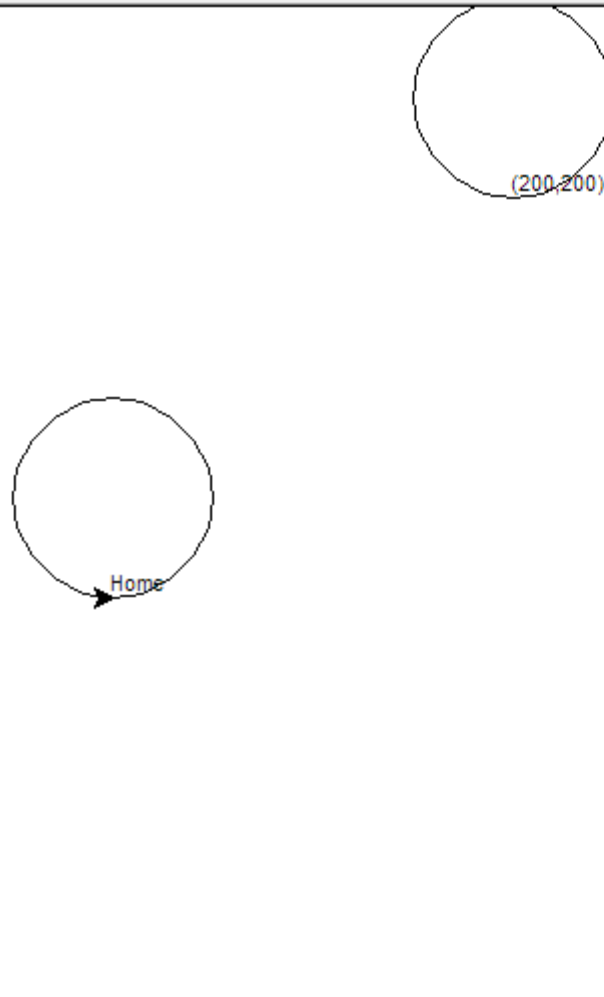

```
>>> turtle.reset()  
>>> turtle.goto(100, 100)
```



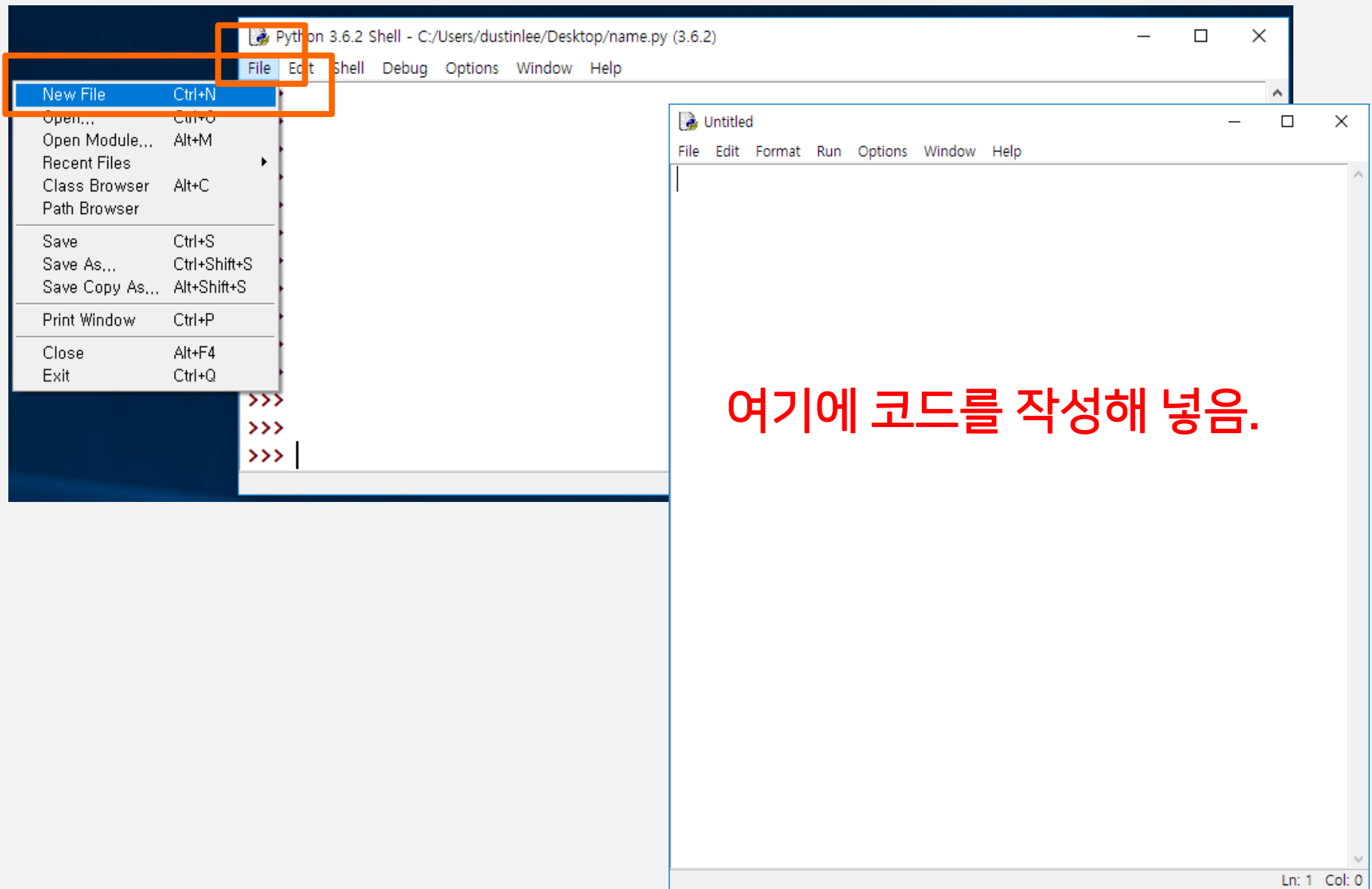
(100,100) 으로 이동한다.
거북이의 머리방향은 변함없이 여전히
오른쪽 방향.



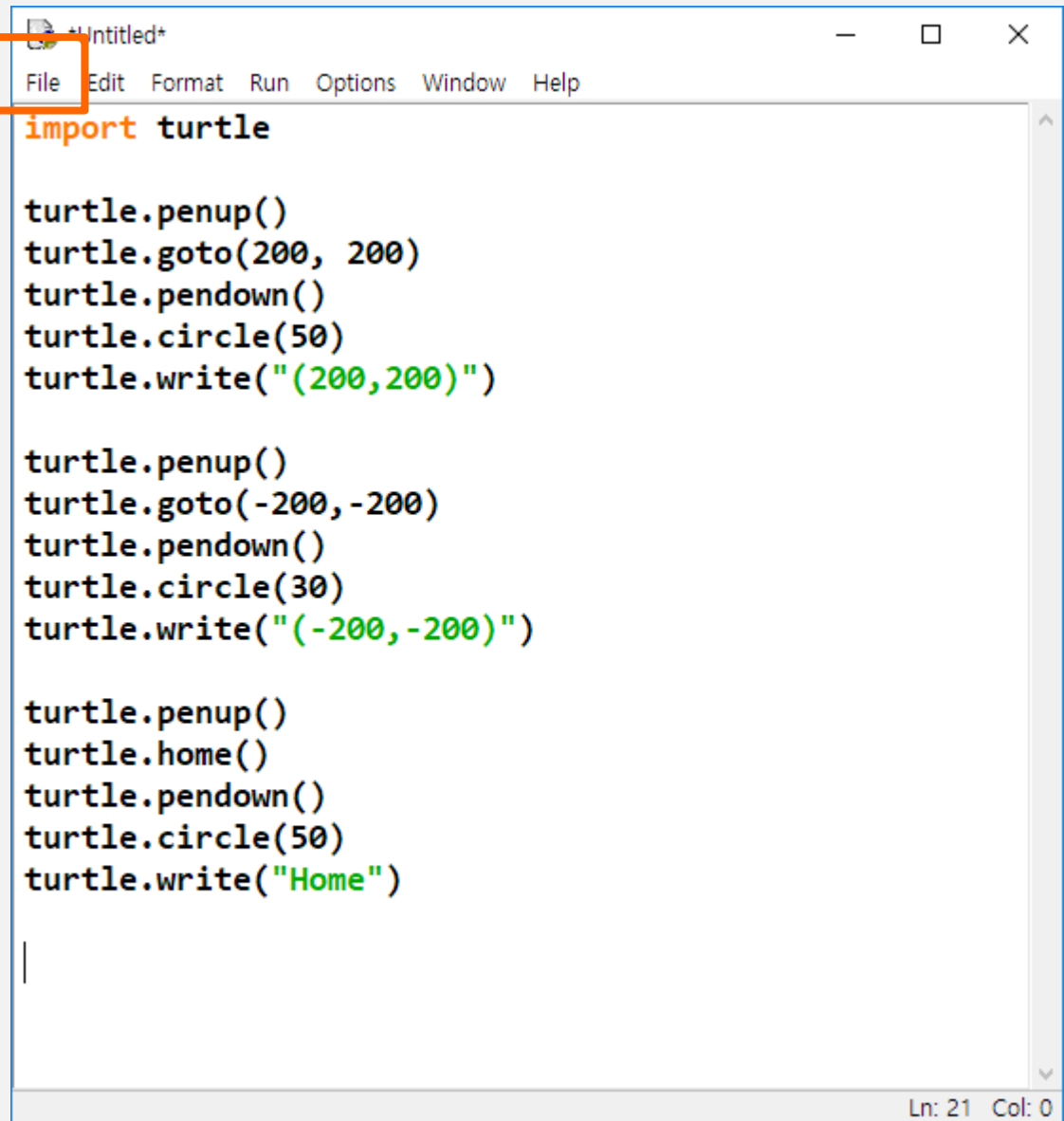
```
>>> turtle.penup()
>>> turtle.goto(200, 200)
>>> turtle.pendown()
>>> turtle.circle(50)
>>> turtle.write("(200,200)")
>>>
>>> turtle.penup()
>>> turtle.goto(-200, -200)
>>> turtle.pendown()
>>> turtle.circle(30)
>>> turtle.write("(-200,-200)")
>>>
>>> turtle.penup()
>>> turtle.home()
>>> turtle.pendown()
>>> turtle.circle(50)
>>> turtle.write("Home")
```



프로그램을 파일로 만들어서 저장



New File	Ctrl+N
Open...	Ctrl+O
Open Module...	Alt+M
Recent Files	
Class Browser	Alt+C
Path Browser	
Save	Ctrl+S
Save As...	Ctrl+Shift+S
Save Copy As...	Alt+Shift+S
Print Window	Ctrl+P
Close	Alt+F4
Exit	Ctrl+Q



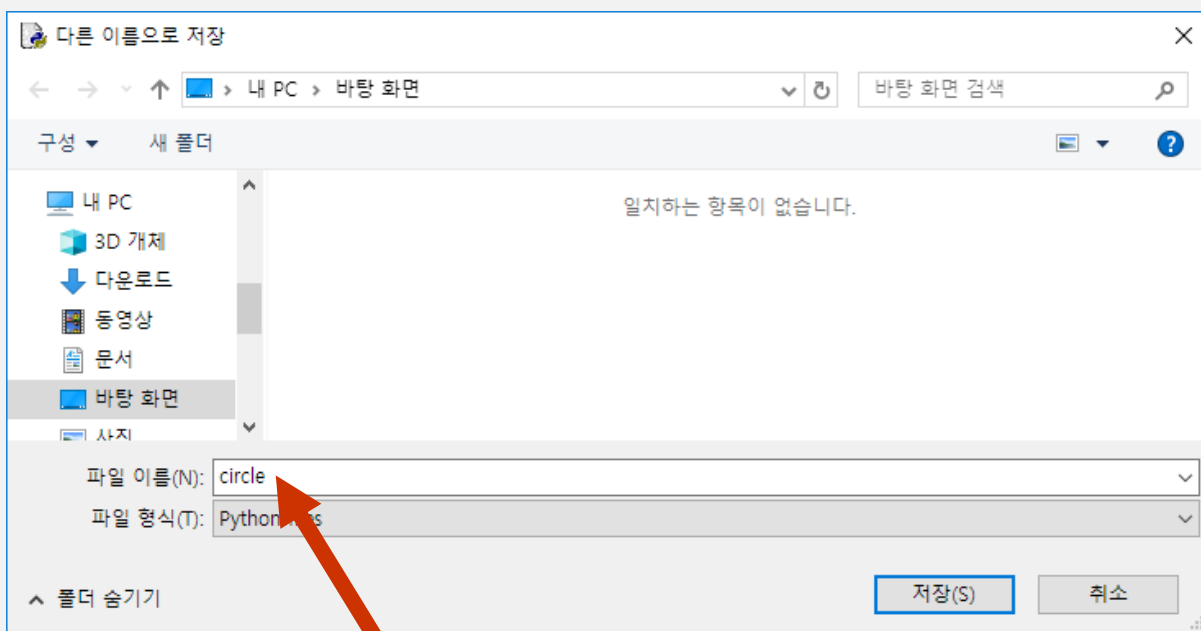
```
import turtle

turtle.penup()
turtle.goto(200, 200)
turtle.pendown()
turtle.circle(50)
turtle.write("(200,200)")

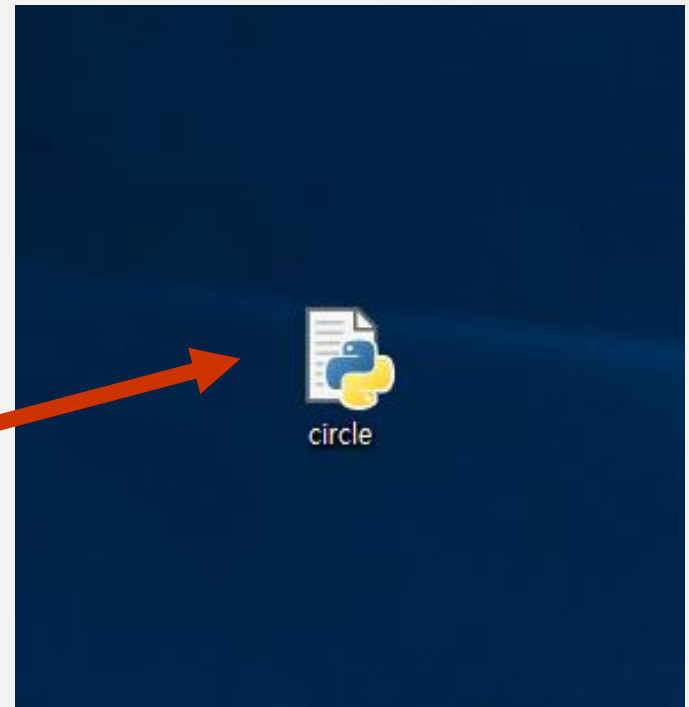
turtle.penup()
turtle.goto(-200,-200)
turtle.pendown()
turtle.circle(30)
turtle.write("(-200,-200)")

turtle.penup()
turtle.home()
turtle.pendown()
turtle.circle(50)
turtle.write("Home")
```

Ln: 21 Col: 0



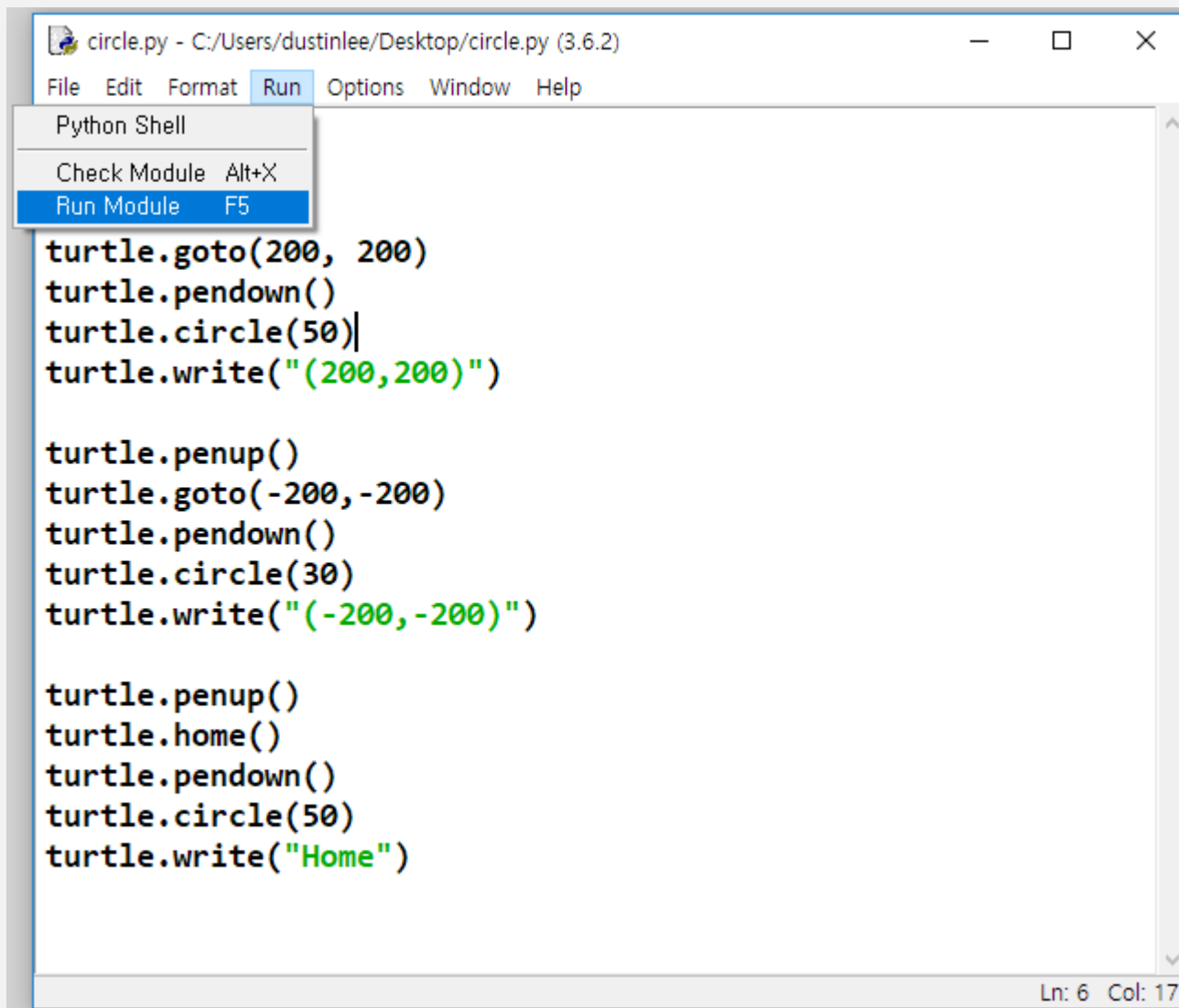
circle이라는 이름으로 바탕
화면에 저장.



바탕화면에 circle.py 라는
이름의 파일이 생성됨.

프로그램 실행 방법 #1

■ Run→Run Module 을 클릭 또는 단축기 F5



The screenshot shows a Python IDE window titled "circle.py - C:/Users/dustinlee/Desktop/circle.py (3.6.2)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The "Run" menu is open, displaying three options: "Python Shell", "Check Module Alt+X", and "Run Module F5". The "Run Module F5" option is highlighted. The script content in the editor is as follows:

```
turtle.goto(200, 200)
turtle.pendown()
turtle.circle(50)
turtle.write("(200,200)")

turtle.penup()
turtle.goto(-200,-200)
turtle.pendown()
turtle.circle(30)
turtle.write("(-200,-200)")

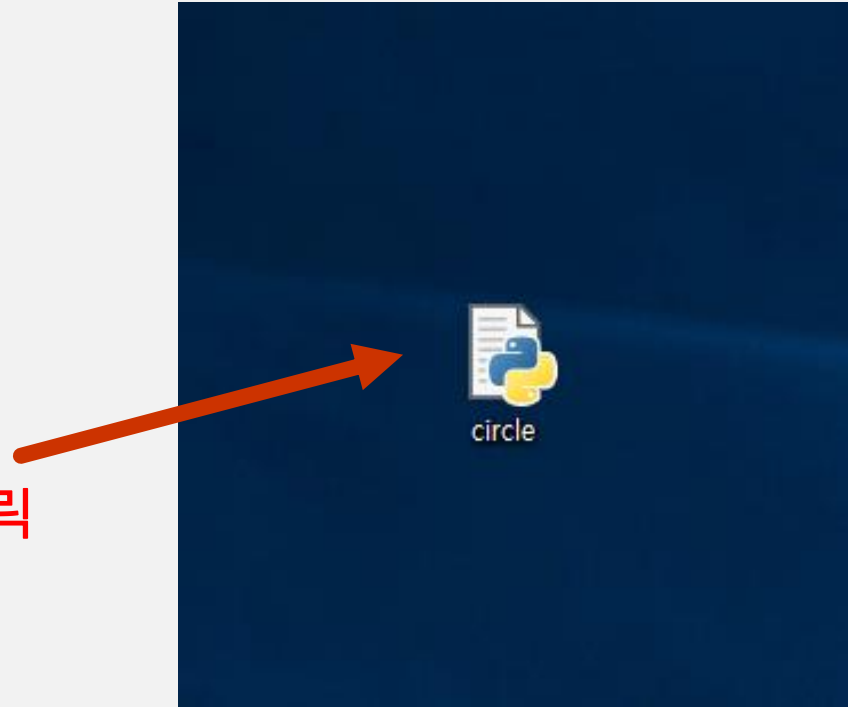
turtle.penup()
turtle.home()
turtle.pendown()
turtle.circle(50)
turtle.write("Home")
```

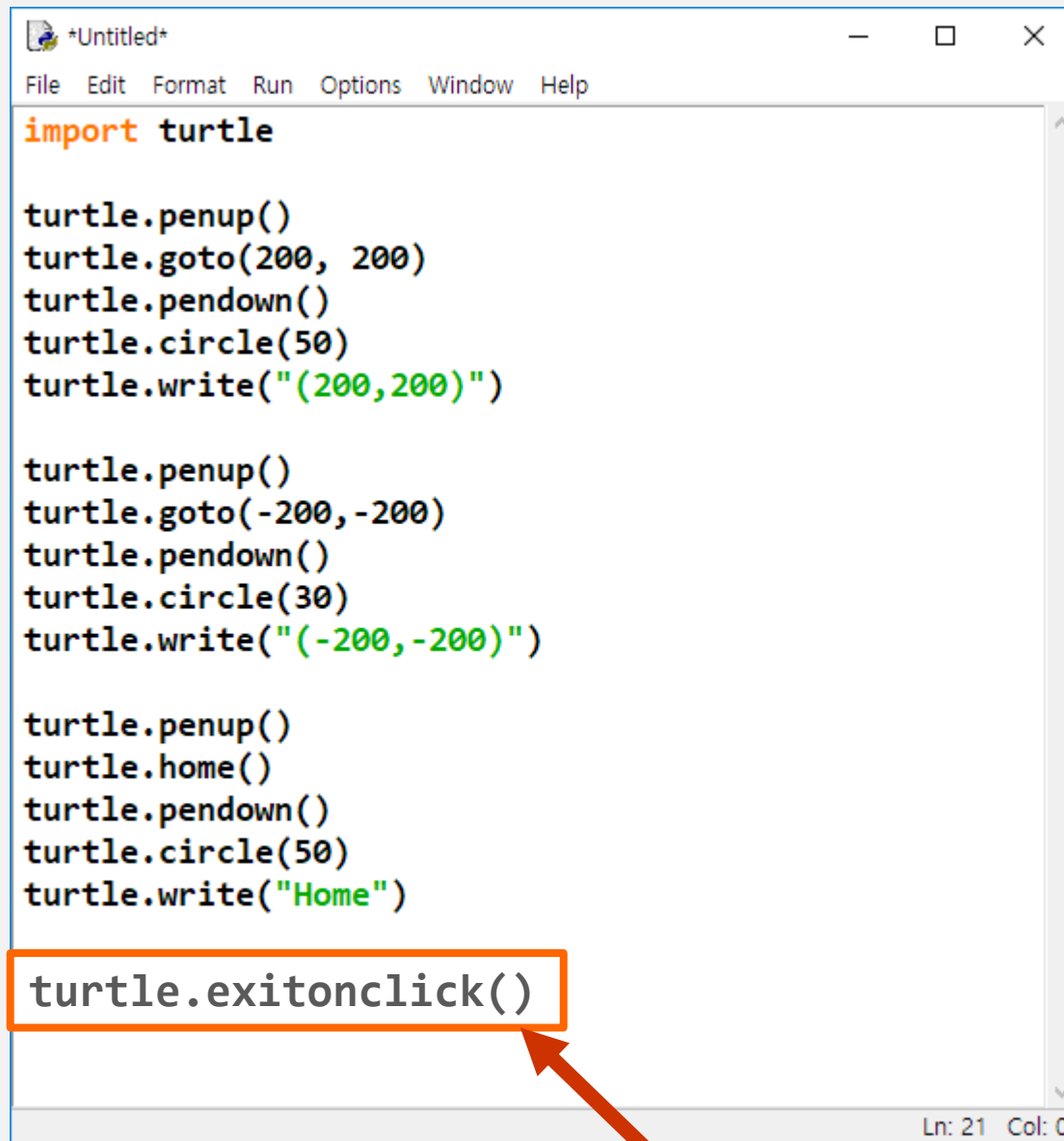
The status bar at the bottom right indicates "Ln: 6 Col: 17".

프로그램 실행 방법 #2

- 프로그램 파일을 더블 클릭하여 실행.
- 문제점은?

circle.py 를 더블 클릭





```
*Untitled*
File Edit Format Run Options Window Help

import turtle

turtle.penup()
turtle.goto(200, 200)
turtle.pendown()
turtle.circle(50)
turtle.write("(200,200)")

turtle.penup()
turtle.goto(-200,-200)
turtle.pendown()
turtle.circle(30)
turtle.write("(-200,-200)")

turtle.penup()
turtle.home()
turtle.pendown()
turtle.circle(50)
turtle.write("Home")

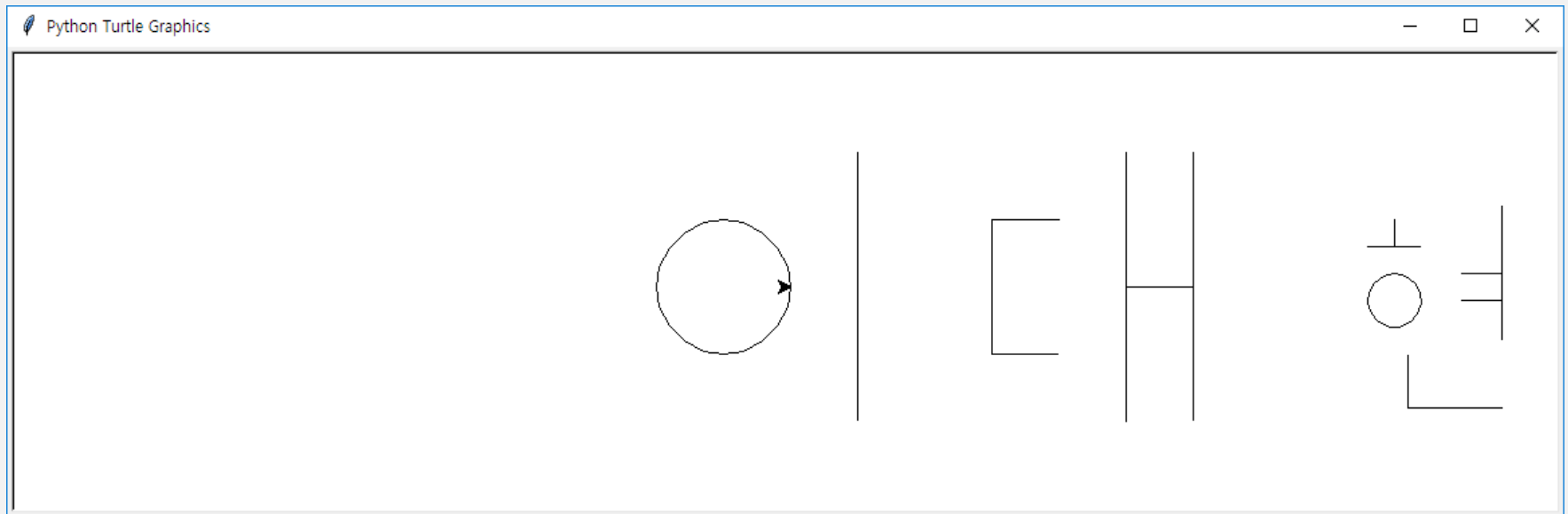
turtle.exitonclick()
```

Ln: 21 Col: 0

코드 마지막 부분에 `exitonclick()` 추가.

Quiz #1: 자기 이름 그리기

- 파일로 작성하여, 바탕화면에 `name.py` 로 저장하고, 더블클릭해서 실행!
- Tuple 을 이용할 것.



random 모듈

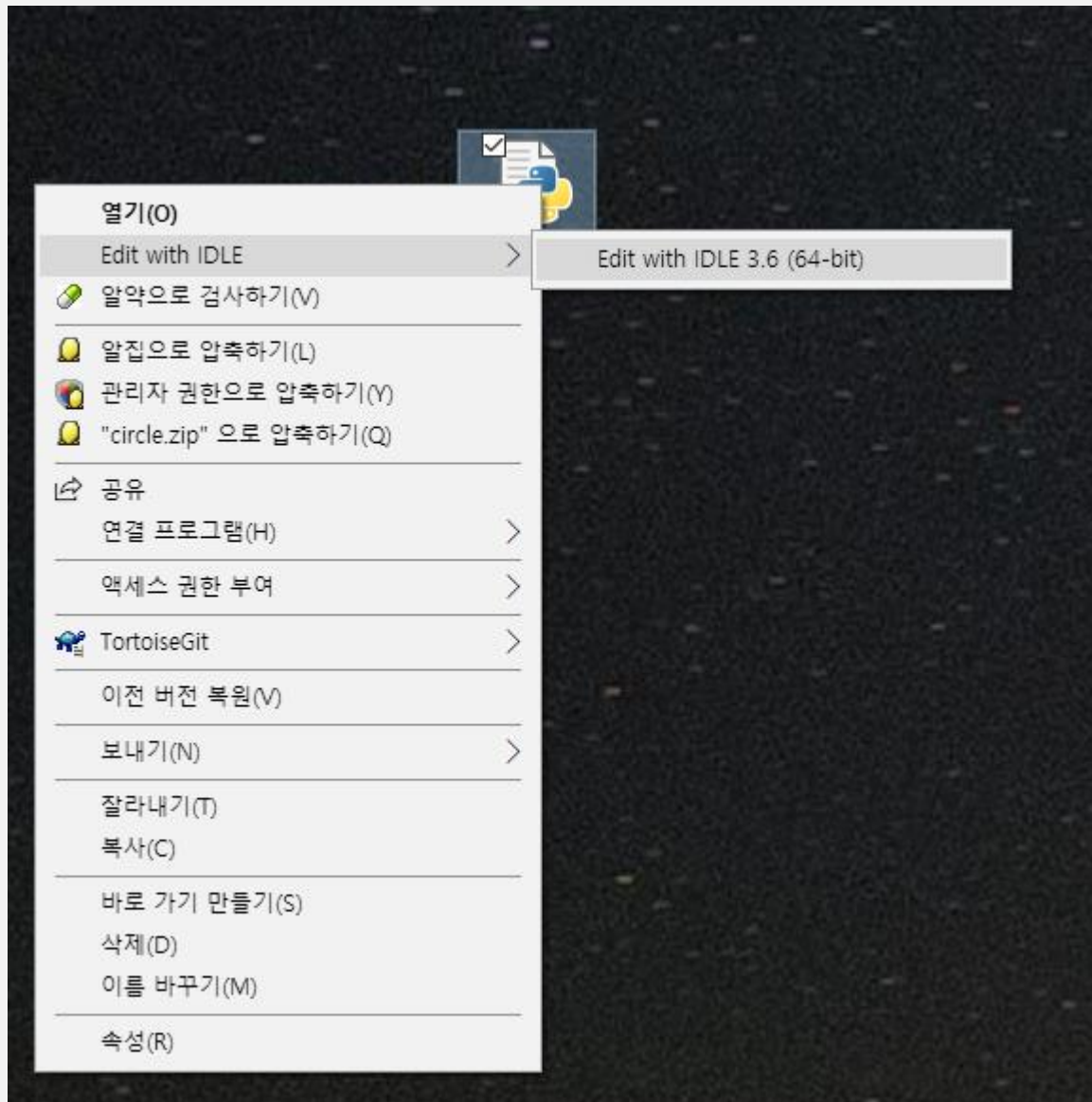
- 주사위를 던지면 어떤 수가 나올까? 무작위로 결정
- 무작위로 어떤 숫자를 뽑아내고자 할 때, random 모듈을 사용하면 된다.

A screenshot of a Python 3.6.2 Shell window. The window has a title bar with the text 'Python 3.6.2 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a series of Python commands and their outputs. The commands are: '>>>', '>>> import random', '>>> random.randint(1,6)', '>>> random.randint(1,6)', '>>> random.randint(1,6)', and '>>> random.randint(1,6)'. The outputs are: '3', '4', '2', and '4'. The prompt '>>>' is shown at the end of the last line. The status bar at the bottom right shows 'Ln: 519 Col: 0'.

```
>>>
>>> import random
>>> random.randint(1,6)
3
>>> random.randint(1,6)
4
>>> random.randint(1,6)
2
>>> random.randint(1,6)
4
>>> random.randint(1,6)
1
>>>
```

random.randint(시작, 끝)

마우스 오른쪽 버튼을 클릭하면, 소스코드를 직접 편집 가능.



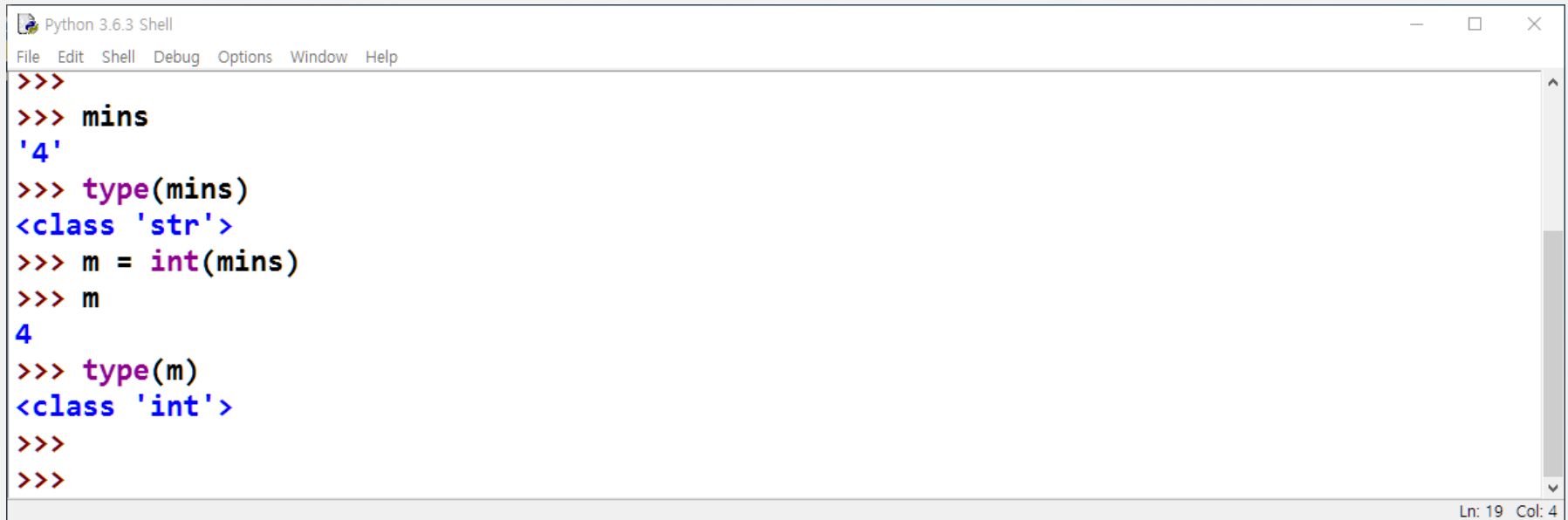
사용자로부터 입력 받기

- input 함수를 이용함.
- 사용자가 입력한 정보가 "문자열"로 되어 넘어옴.

[illegible]

자료형 변환

- mins의 값은 4가 아니고, '4'임. 즉, 정수가 아니고 문자열임.
- 이것을 정수로 바꾸기 위해서는 int() 라는 함수를 사용함.

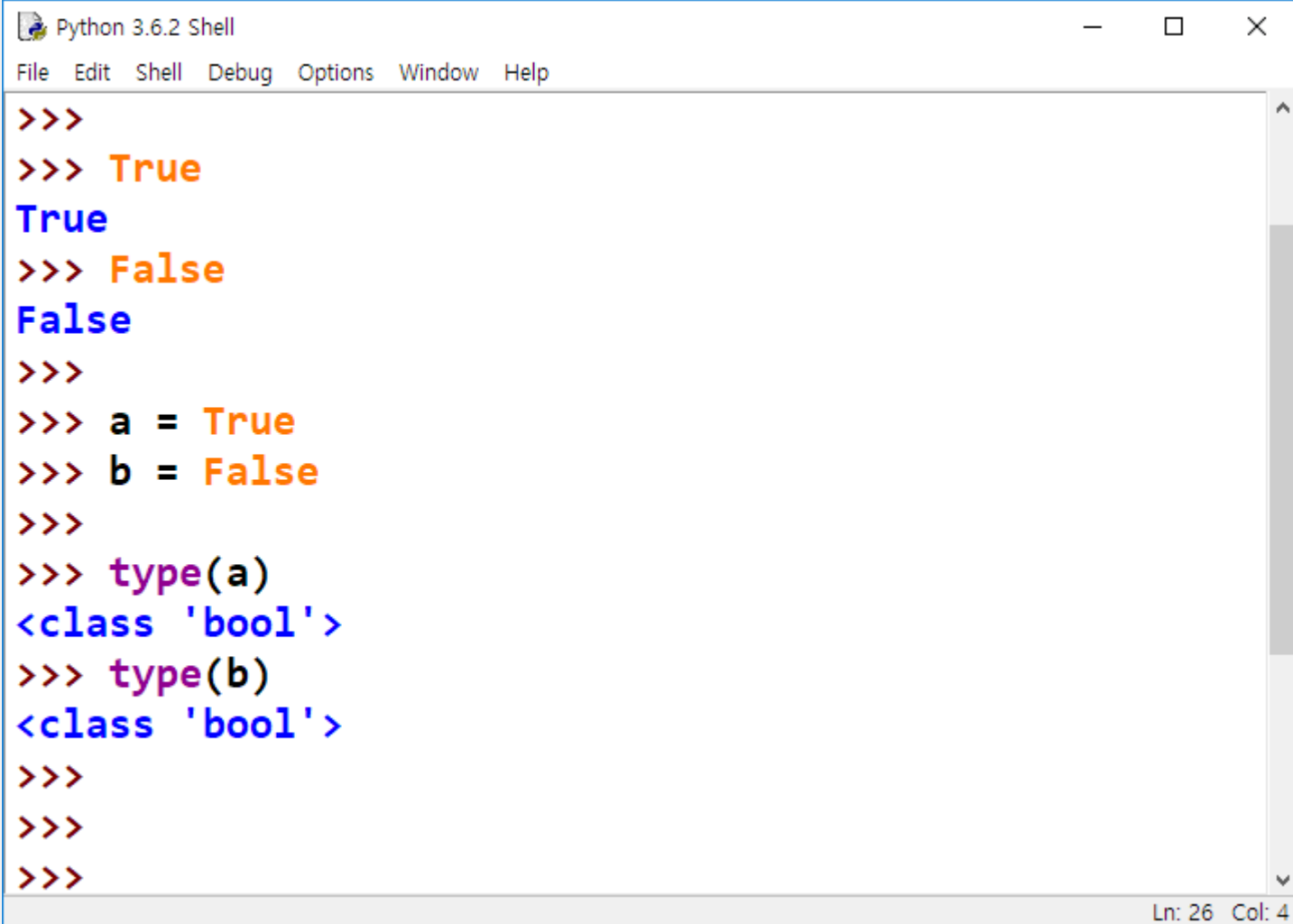


```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
>>>
>>> mins
'4'
>>> type(mins)
<class 'str'>
>>> m = int(mins)
>>> m
4
>>> type(m)
<class 'int'>
>>>
>>>
```

Ln: 19 Col: 4

자료형: bool

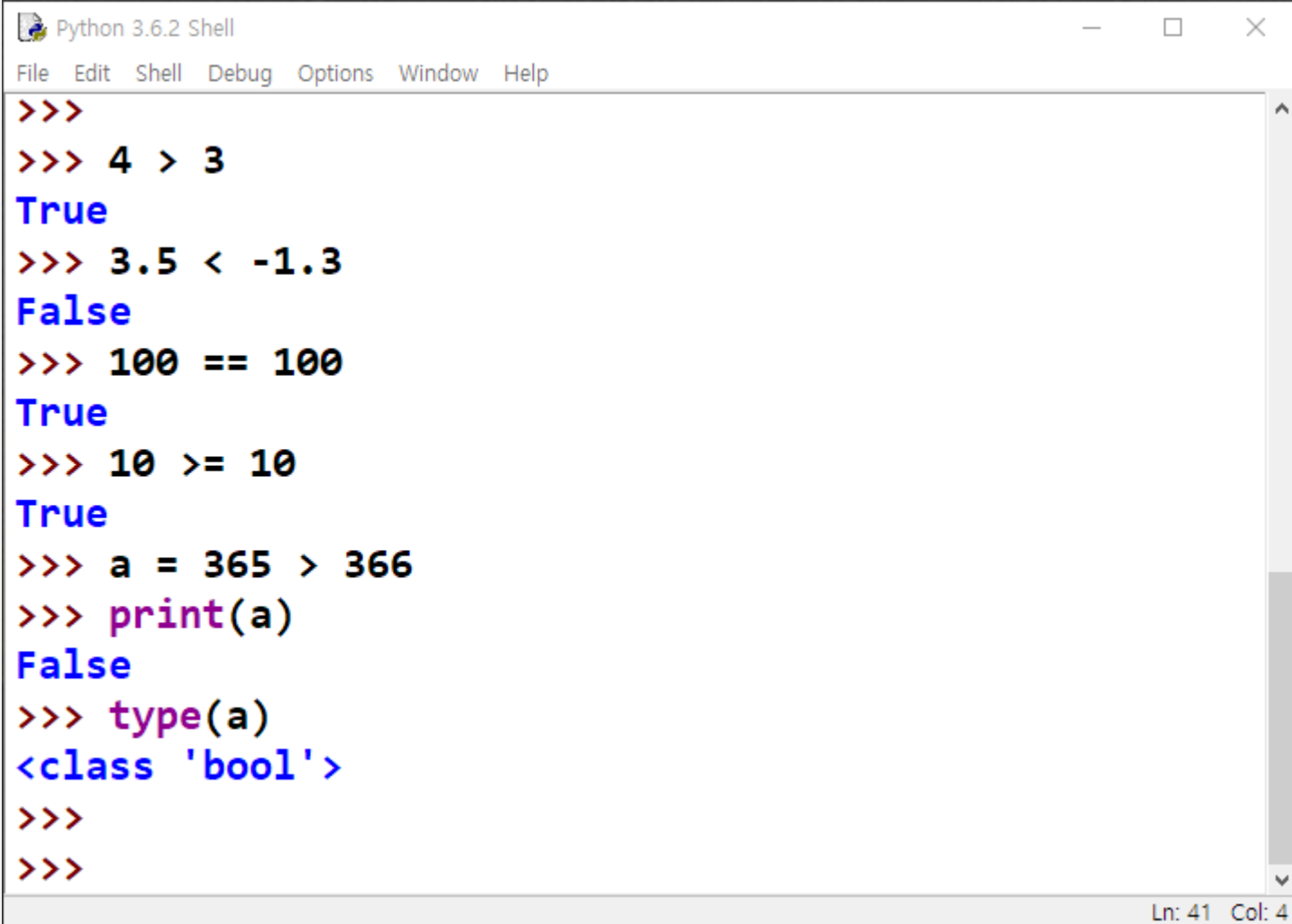
- 참(True), 또는 거짓(False)을 나타내는데 사용되는 자료형

A screenshot of a Python 3.6.2 Shell window. The window has a title bar with the text 'Python 3.6.2 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area is a text editor showing a series of Python commands and their outputs. The commands are: '>>>', '>>> True', '>>> False', '>>>', '>>> a = True', '>>> b = False', '>>>', '>>> type(a)', '>>> type(b)', and three more '>>>' prompts. The outputs are: 'True' (for the first True command), 'False' (for the first False command), '<class \'bool\'>' (for both type(a) and type(b)), and no output for the other commands. The text is color-coded: prompts are black, keywords like 'True' and 'False' are orange, and 'type' is purple. The status bar at the bottom right shows 'Ln: 26 Col: 4'.

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
>>>
>>> True
True
>>> False
False
>>>
>>> a = True
>>> b = False
>>>
>>> type(a)
<class 'bool'>
>>> type(b)
<class 'bool'>
>>>
>>>
>>>
Ln: 26 Col: 4
```

비교 연산(Comparison Operation)

- 두개의 값의 대소, 동일 등을 확인하는 계산.
- 결과는 참(True) 또는 거짓(False)임.

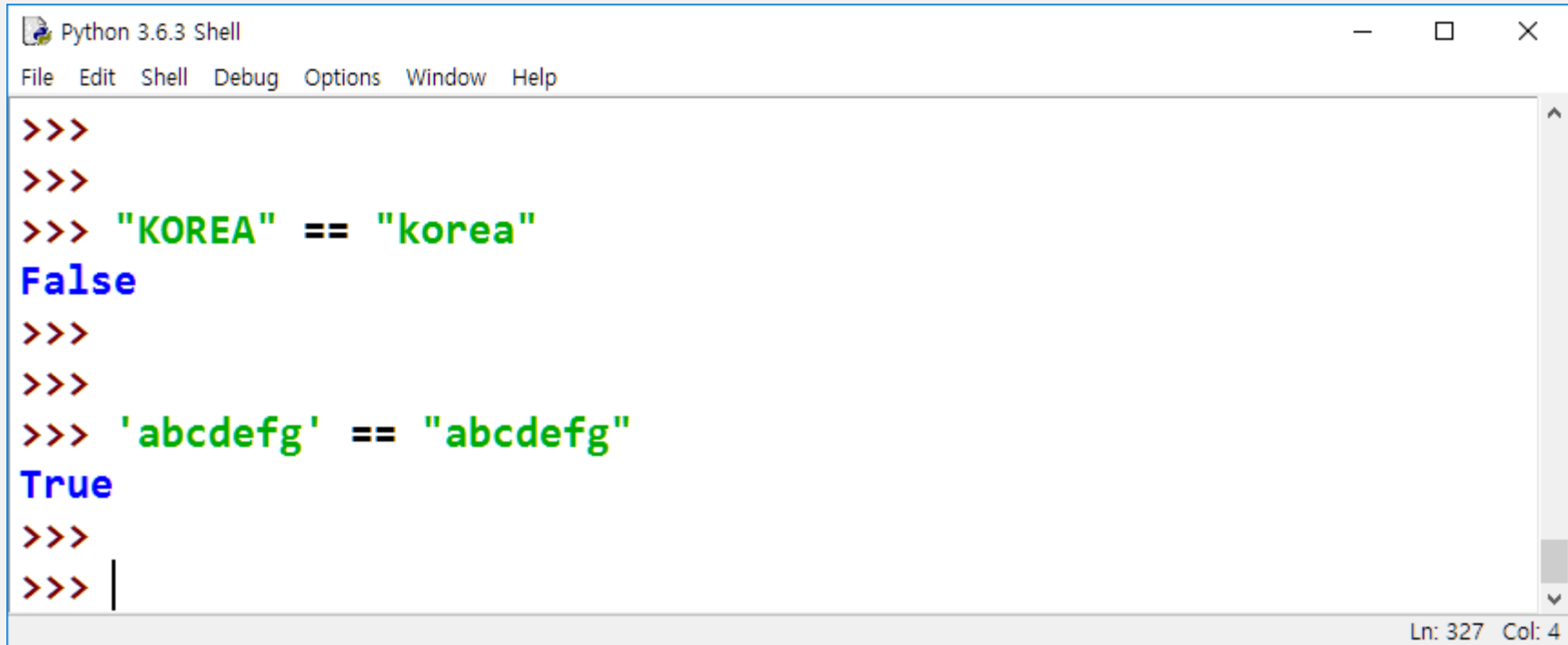
A screenshot of a Python 3.6.2 Shell window. The window has a title bar with the text 'Python 3.6.2 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main area of the window contains a series of Python commands and their outputs. The commands are: '>>>', '>>> 4 > 3', '>>> 3.5 < -1.3', '>>> 100 == 100', '>>> 10 >= 10', '>>> a = 365 > 366', '>>> print(a)', and '>>> type(a)'. The outputs are: 'True', 'False', 'True', 'False', and '<class \'bool\'>'. The window also has a status bar at the bottom right showing 'Ln: 41 Col: 4'.

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
>>>
>>> 4 > 3
True
>>> 3.5 < -1.3
False
>>> 100 == 100
True
>>> 10 >= 10
True
>>> a = 365 > 366
>>> print(a)
False
>>> type(a)
<class 'bool'>
>>>
>>>
Ln: 41 Col: 4
```


비교 연산 기호

기호	뜻
<	작다
<=	작거나 같다
==	같다
>=	크거나 같다
>	크다
!=	다르다

문자열의 비교

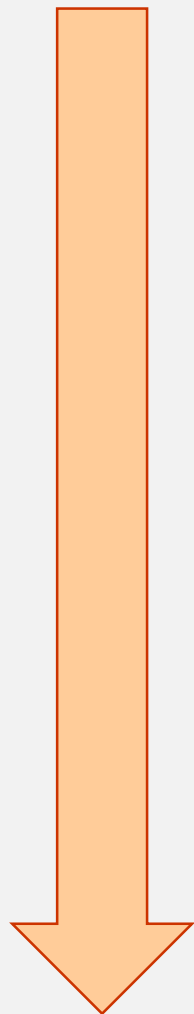
A screenshot of a Python 3.6.3 Shell window. The window has a title bar with the text "Python 3.6.3 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a Python interactive prompt with the following text:

```
>>>  
>>>  
>>> "KOREA" == "korea"  
False  
>>>  
>>>  
>>> 'abcdefg' == "abcdefg"  
True  
>>>  
>>> |
```

The text is color-coded: the prompt characters are red, the strings are green, and the boolean results are blue. A vertical scrollbar is visible on the right side of the text area. At the bottom right of the window, the status bar shows "Ln: 327 Col: 4".

```
Python 3.6.3 Shell  
File Edit Shell Debug Options Window Help  
>>>  
>>>  
>>> "KOREA" == "korea"  
False  
>>>  
>>>  
>>> 'abcdefg' == "abcdefg"  
True  
>>>  
>>> |  
Ln: 327 Col: 4
```

파이썬 문장은 위에서부터 아래로 차례로 실행



```
circle.py - C:\Users\dustinlee\Desktop\circle.py (3.6.2)
File Edit Format Run Options Window Help

import turtle

turtle.penup()
turtle.goto(200, 200)
turtle.pendown()
turtle.circle(50)
turtle.write("(200,200)")

turtle.penup()
turtle.goto(-200,-200)
turtle.pendown()
turtle.circle(30)
turtle.write("(-200,-200)")

turtle.penup()
turtle.home()
turtle.pendown()
turtle.circle(50)
turtle.write("Home")

turtle.exitonclick()

Ln: 1 Col: 0
```

문법: 조건문 (Conditional Statement)

- 조건을 검사하여, 그 결과에 따라 처리를 하는 문장

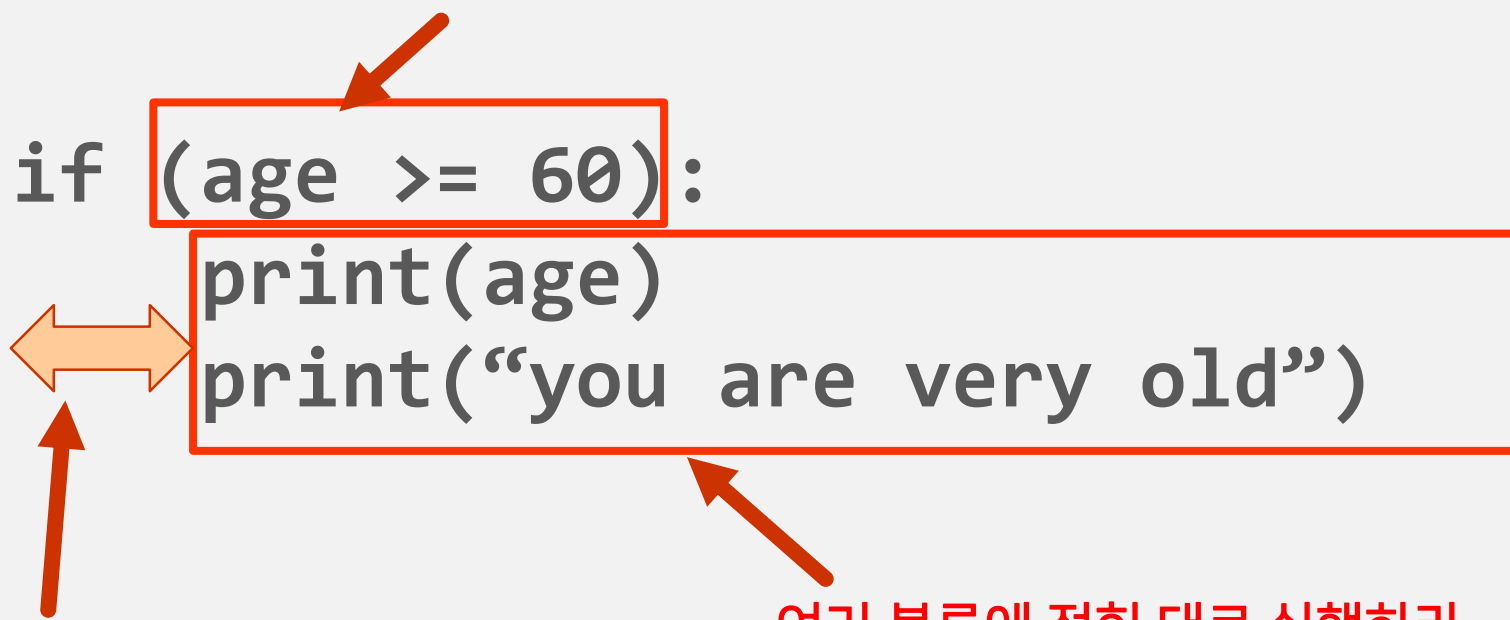
```
if (age >= 60):  
    print(age)  
    print("you are very old")
```



만약 age가 60 이상이면, age의 값을 출력하고, 이어서 "you are very old"라는 문자열을 출력하라.
age 가 60보다 작으면? 아무것도 하지 않음.

이 조건이 참(True)이면,

```
if (age >= 60):  
    print(age)  
    print("you are very old")
```



여기 블록에 적힌 대로 실행하라.

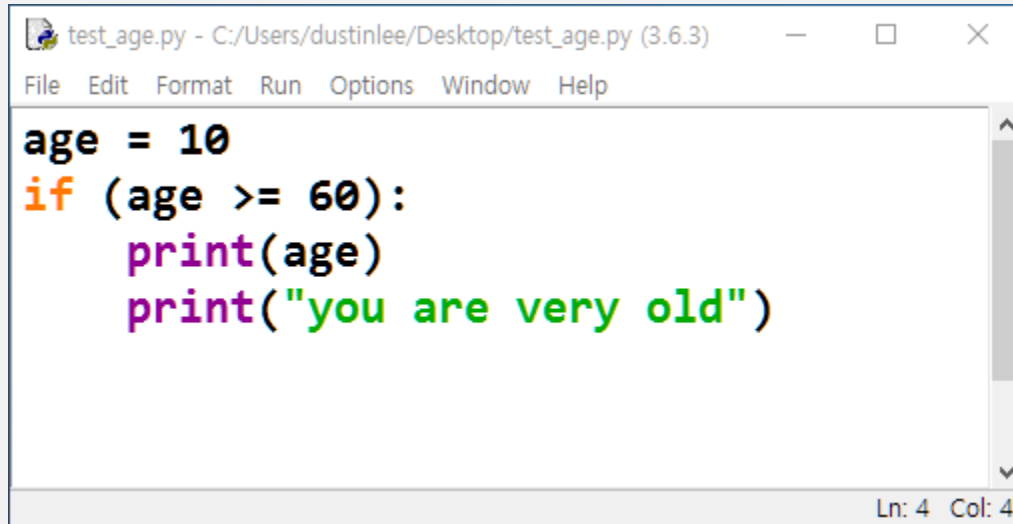
들여쓰기(indentation)

*** 매우 중요 ***

일반적으로 공백4개씩

조건이 참이면, 들여쓰기된 블록을 실행함.

test_age.py



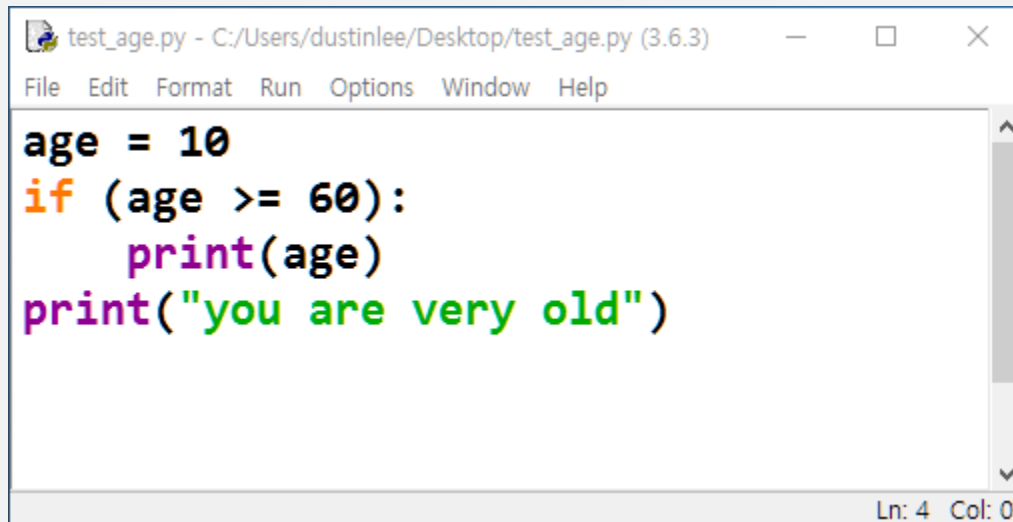
A screenshot of a Python IDE window titled "test_age.py - C:/Users/dustinlee/Desktop/test_age.py (3.6.3)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

```
age = 10
if (age >= 60):
    print(age)
    print("you are very old")
```

The status bar at the bottom right indicates "Ln: 4 Col: 4".



test_age.py



A screenshot of a Python IDE window titled "test_age.py - C:/Users/dustinlee/Desktop/test_age.py (3.6.3)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the same Python code as the first screenshot:

```
age = 10
if (age >= 60):
    print(age)
    print("you are very old")
```

The status bar at the bottom right indicates "Ln: 4 Col: 0".

문법: 조건문 (Conditional Statement) 확장형

```
if (age >= 60):  
    print(age)  
    print("you are very old")  
else:  
    print(age)  
    print("you are young")
```

문법: 조건문 (Conditional Statement) 확장형

```
if (age >= 60):  
    print(age)  
    print("you are very old")  
elif (age <= 20):  
    print(age)  
    print("you are very young")  
else:  
    print(age)  
    print("you are young")
```


버스 요금 계산기를 만들어 보자.

■ 버스 요금

- 미취학 만 7살 미만 : 공짜
- 초등학생 : 450원
- 중고등학생: 720원
- 성인: 1200원
- 65세 이상: 공짜

bus_fare.py

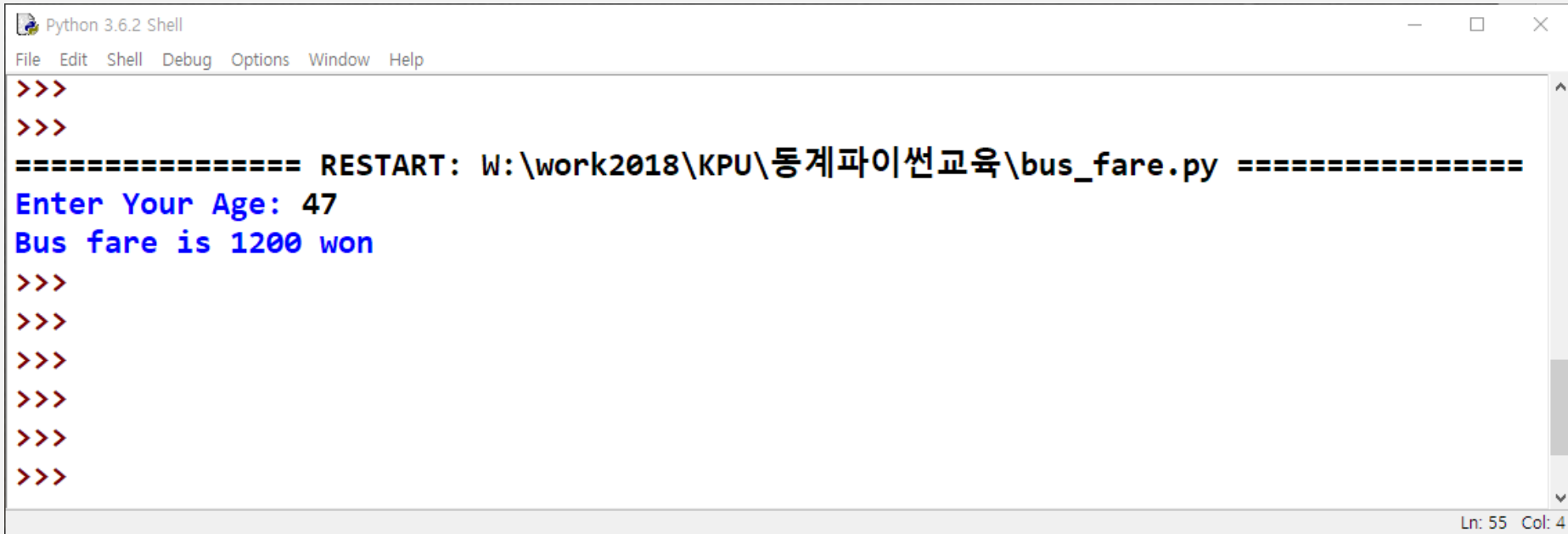
```
bus_fare.py - C:/Users/dustinlee/Desktop/bus_fare.py (3.6.3)
File Edit Format Run Options Window Help

age_str = input("Enter Your Age: ")
age = int(age_str)
if age <= 6:
    print("Bus is free")
elif age <= 12:
    print("Bus fare is 450 won")
elif age <= 18:
    print("Bus fare is 720 won")
elif age <= 64:
    print("Bus fare is 1200 won")
else:
    print("Bus is free")

Ln: 15 Col: 0
```

에디터 화면에서, F5를 눌러서 실행하면,

- IDLE 화면에서, RESTART 가 출력되고, 프로그램 실행이 시작됨.
- 불편한 점은?



The screenshot shows a Python 3.6.2 Shell window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The command prompt shows two empty lines, followed by a restart message: "===== RESTART: W:\work2018\KPU\통계파이썬교육\bus_fare.py =====". Below this, the program prompts "Enter Your Age: 47" and outputs "Bus fare is 1200 won". There are several more empty lines in the prompt. The status bar at the bottom right indicates "Ln: 55 Col: 4".

```
>>>  
>>>  
===== RESTART: W:\work2018\KPU\통계파이썬교육\bus_fare.py =====  
Enter Your Age: 47  
Bus fare is 1200 won  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
Ln: 55 Col: 4
```

문법: while 반복문 (Iteration Statement)

- 어떤 조건을 만족하는 동안, 계속해서 반복적으로 실행하는 문장.

```
while <조건문>:  
    <수행할 문장1>  
    <수행할 문장2>  
    <수행할 문장3>  
    ...
```

```
import turtle
```

```
count = 10
```

```
while (count > 0):  
    turtle.forward(100)  
    turtle.left(30)  
    count = count - 1
```



count가 0 보다 크면 계속해서 반복한다. 뭘? (turtle을 앞으로 100 이동,
그리고 왼쪽으로 30도 회전, 그리고 count 값 하나 감소)

```
import turtle
```

```
count = 10
```

```
while (count > 0):
```

```
    turtle.forward(100)
```

```
    turtle.left(30)
```

```
    count = count - 1
```

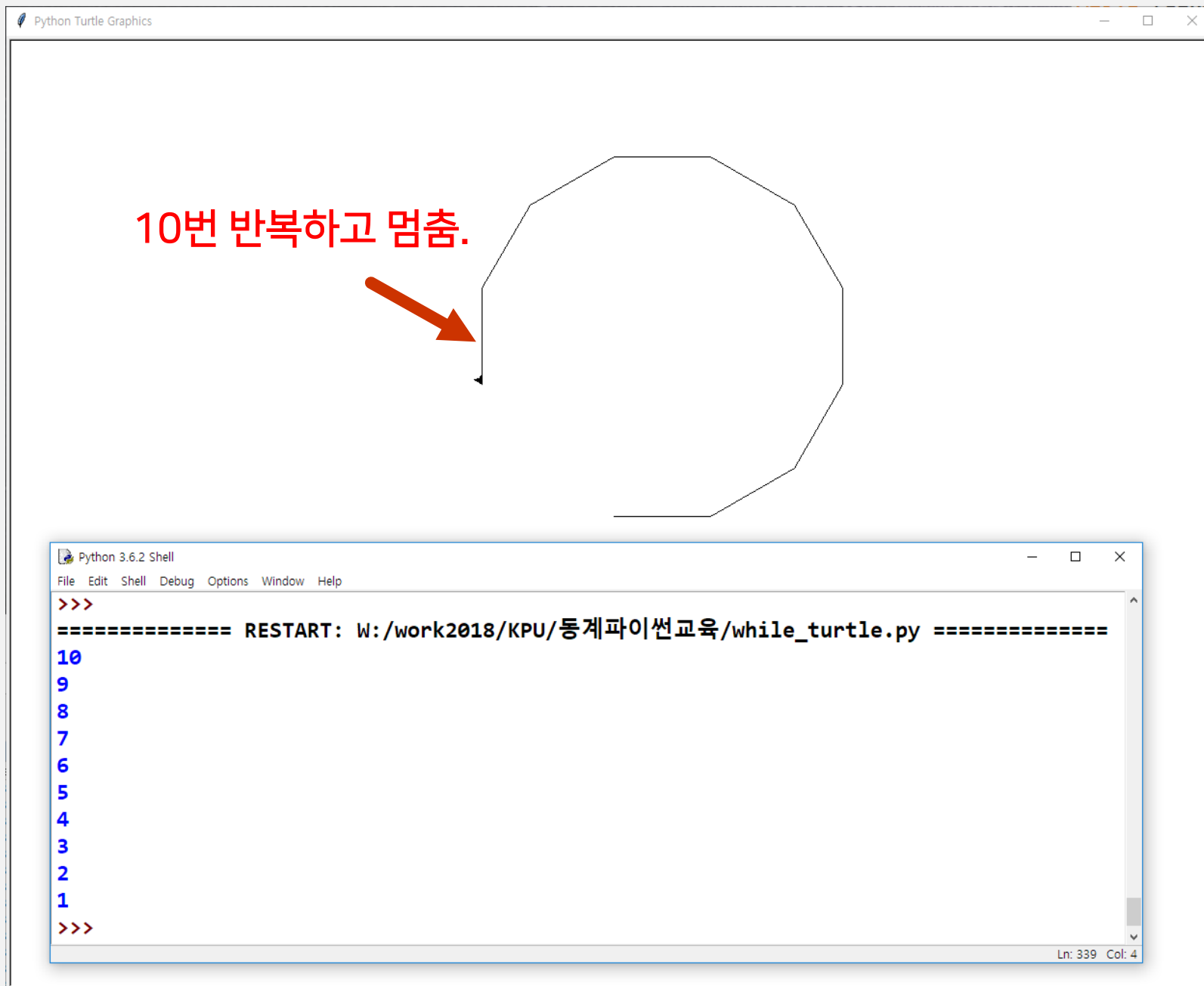
이 조건이 참(True)인 동안



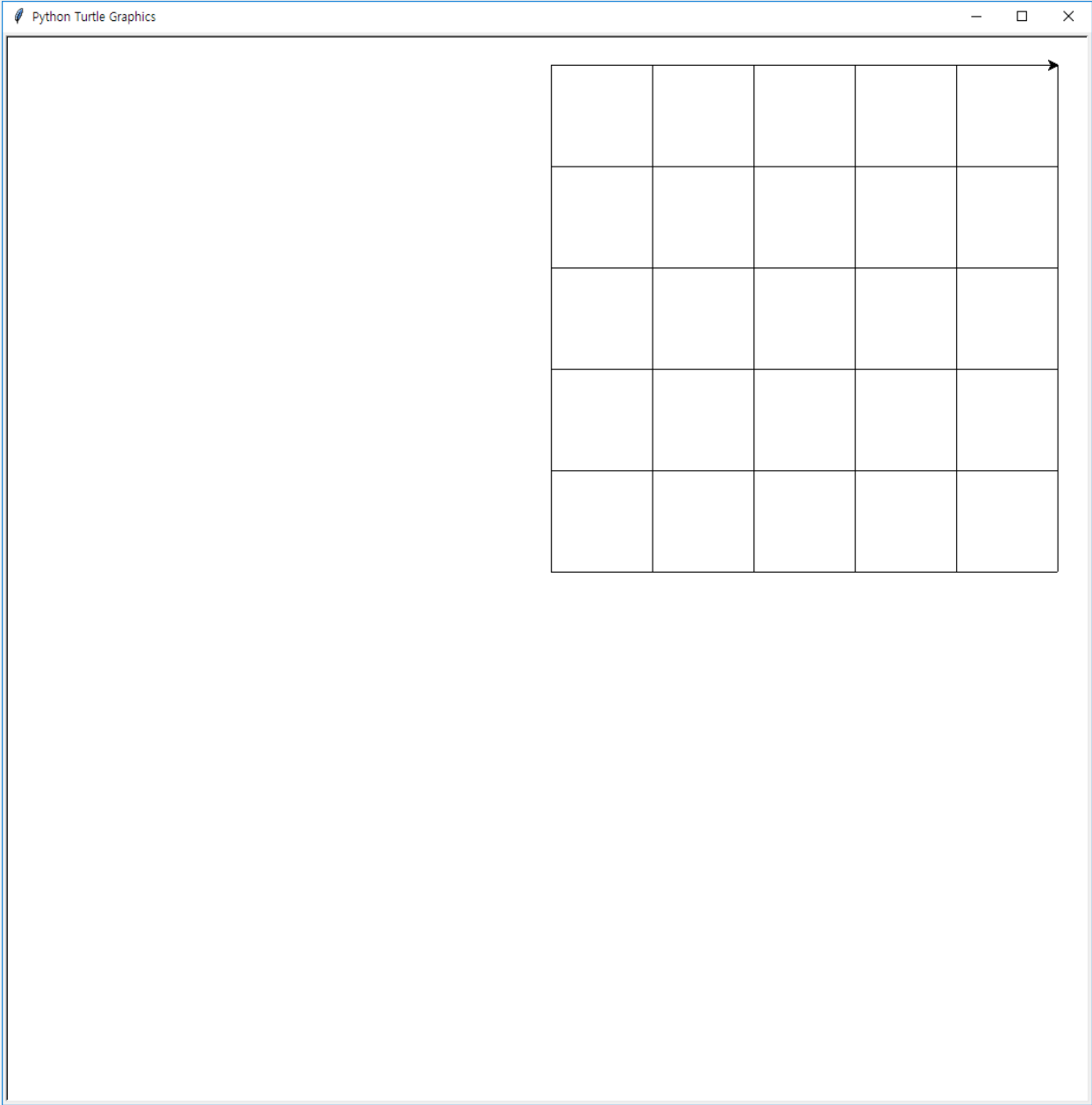
들여쓰기(indentation)

*** 매우 중요 ***

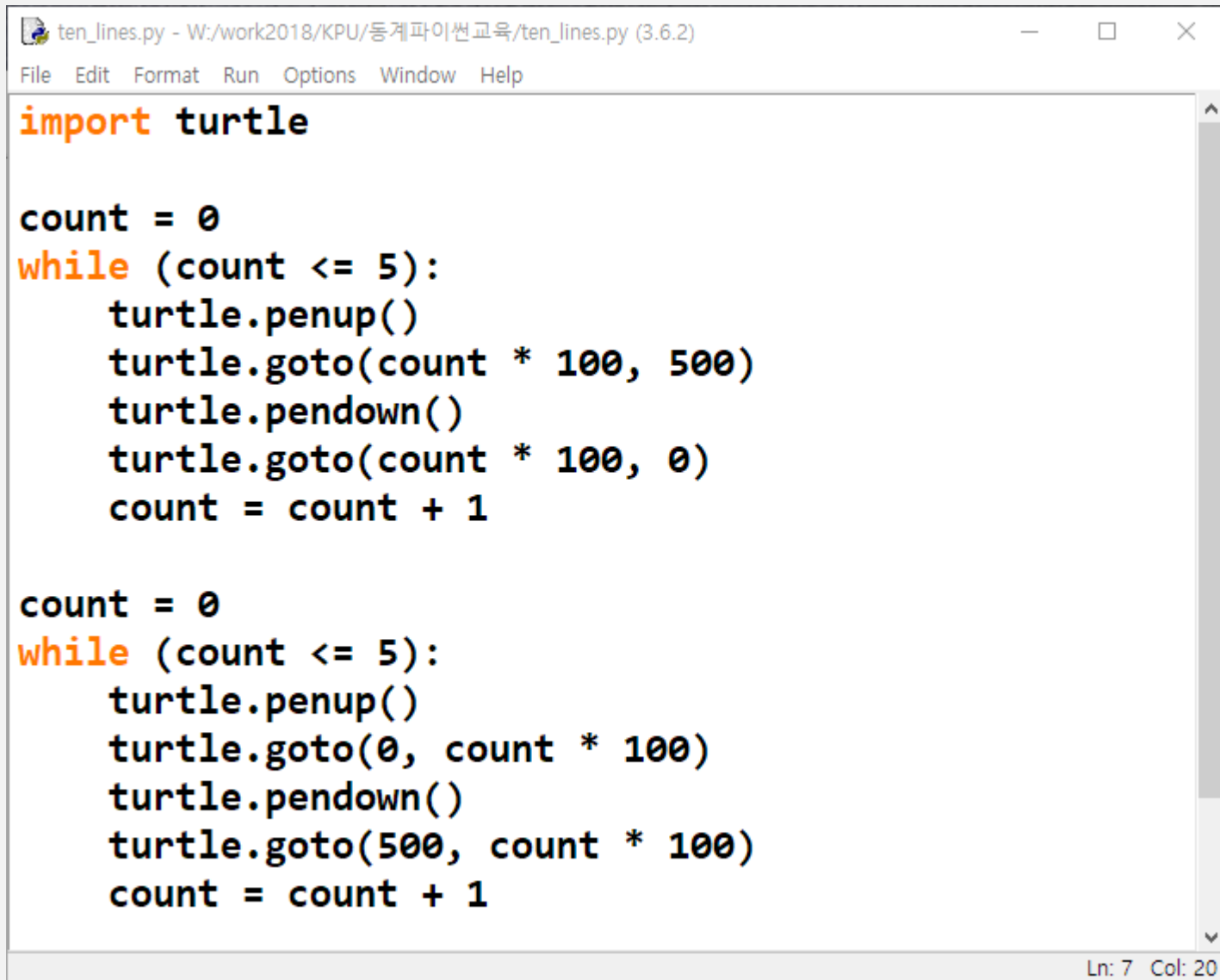
여기 블록을 반복적으로 실행한다.



퀴즈 #2: 모눈 그리기(길이 500, 간격 100)



하나의 답안



```
ten_lines.py - W:/work2018/KPU/등계파이썬교육/ten_lines.py (3.6.2)
File Edit Format Run Options Window Help

import turtle

count = 0
while (count <= 5):
    turtle.penup()
    turtle.goto(count * 100, 500)
    turtle.pendown()
    turtle.goto(count * 100, 0)
    count = count + 1

count = 0
while (count <= 5):
    turtle.penup()
    turtle.goto(0, count * 100)
    turtle.pendown()
    turtle.goto(500, count * 100)
    count = count + 1

Ln: 7 Col: 20
```


거북이의 방향 설정

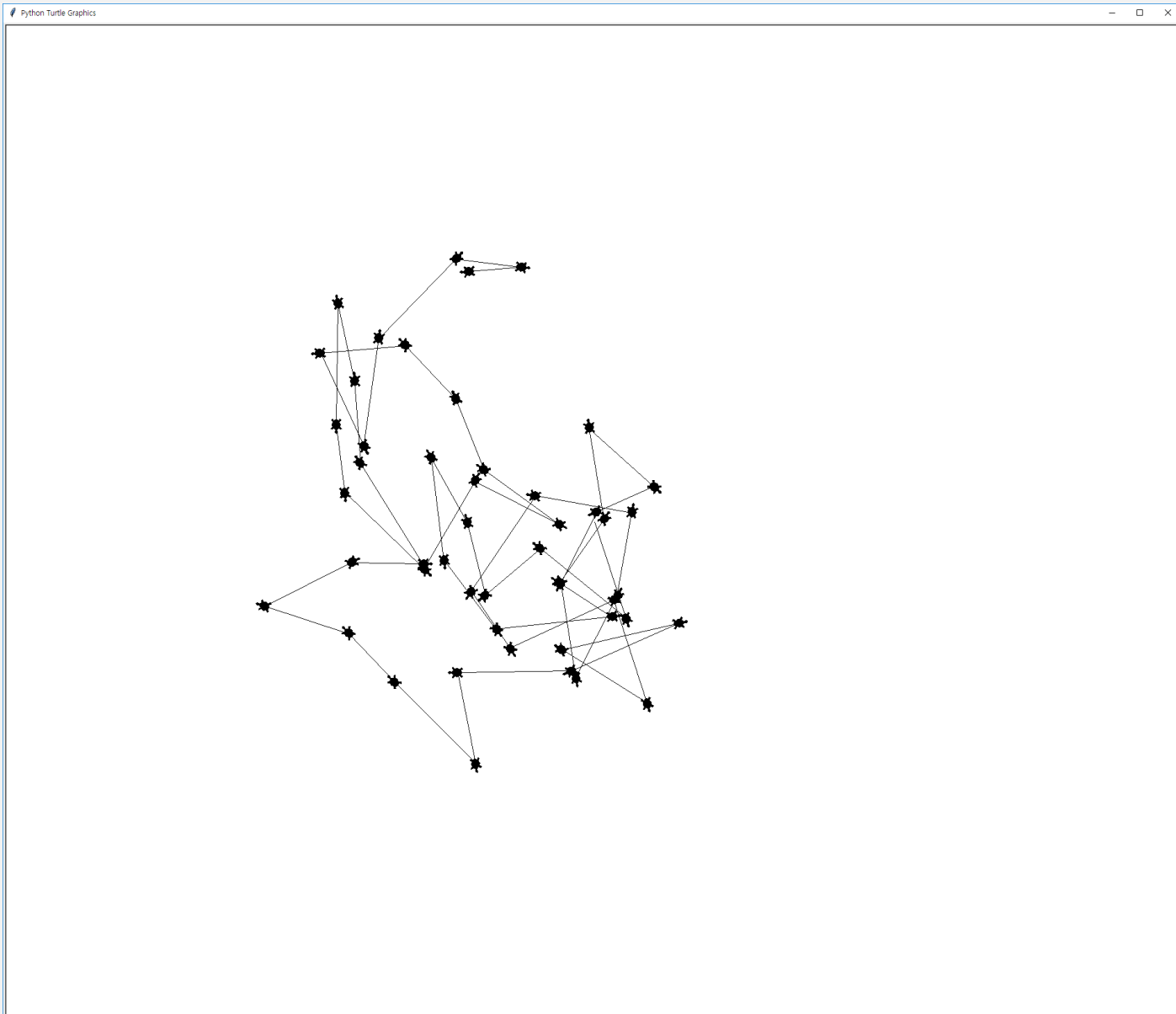
turtle.setheading(각도)

The image shows two windows from a Python 3.6.2 environment. The left window is the 'Python 3.6.2 Shell' with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). It contains the following code and annotations:

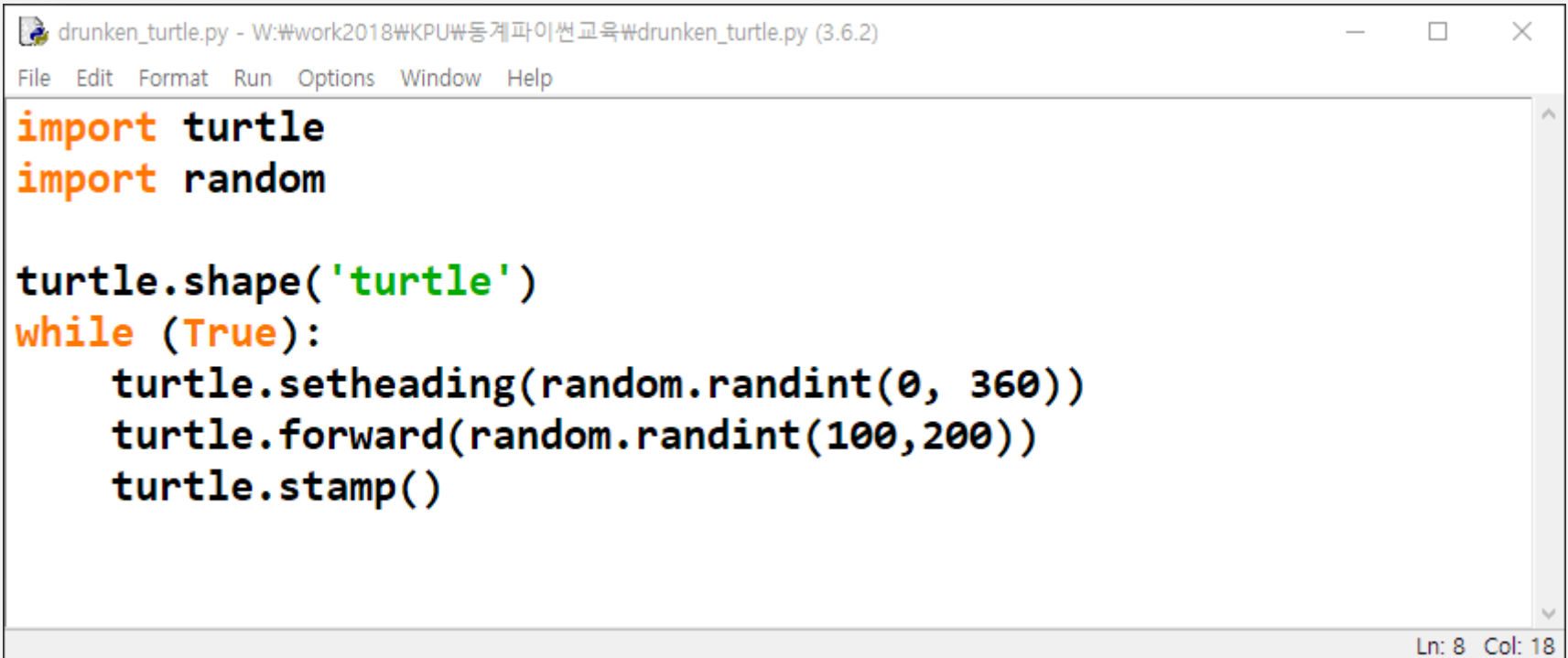
```
>>>
>>>
>>> import turtle
>>> turtle.setheading(90)   윗쪽
>>> turtle.setheading(0)   오른쪽
>>> turtle.setheading(180)  왼쪽
>>> turtle.setheading(-90)  아래쪽
>>> turtle.setheading(360)  오른쪽
>>>
>>>
>>>
>>>
>>>
```

The right window is titled 'Python Turtle Graphics' and displays a white canvas with a small black arrow (the turtle) pointing to the right, representing a heading of 0 degrees.

술취한 거북이?



drunken_turtle.py

A screenshot of a Python IDE window titled "drunken_turtle.py - W:\work2018\KPU\등계파이썬교육\drunken_turtle.py (3.6.2)". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
import turtle
import random

turtle.shape('turtle')
while (True):
    turtle.setheading(random.randint(0, 360))
    turtle.forward(random.randint(100, 200))
    turtle.stamp()
```

The status bar at the bottom right shows "Ln: 8 Col: 18".

```
drunken_turtle.py - W:\work2018\KPU\등계파이썬교육\drunken_turtle.py (3.6.2)
File Edit Format Run Options Window Help

import turtle
import random

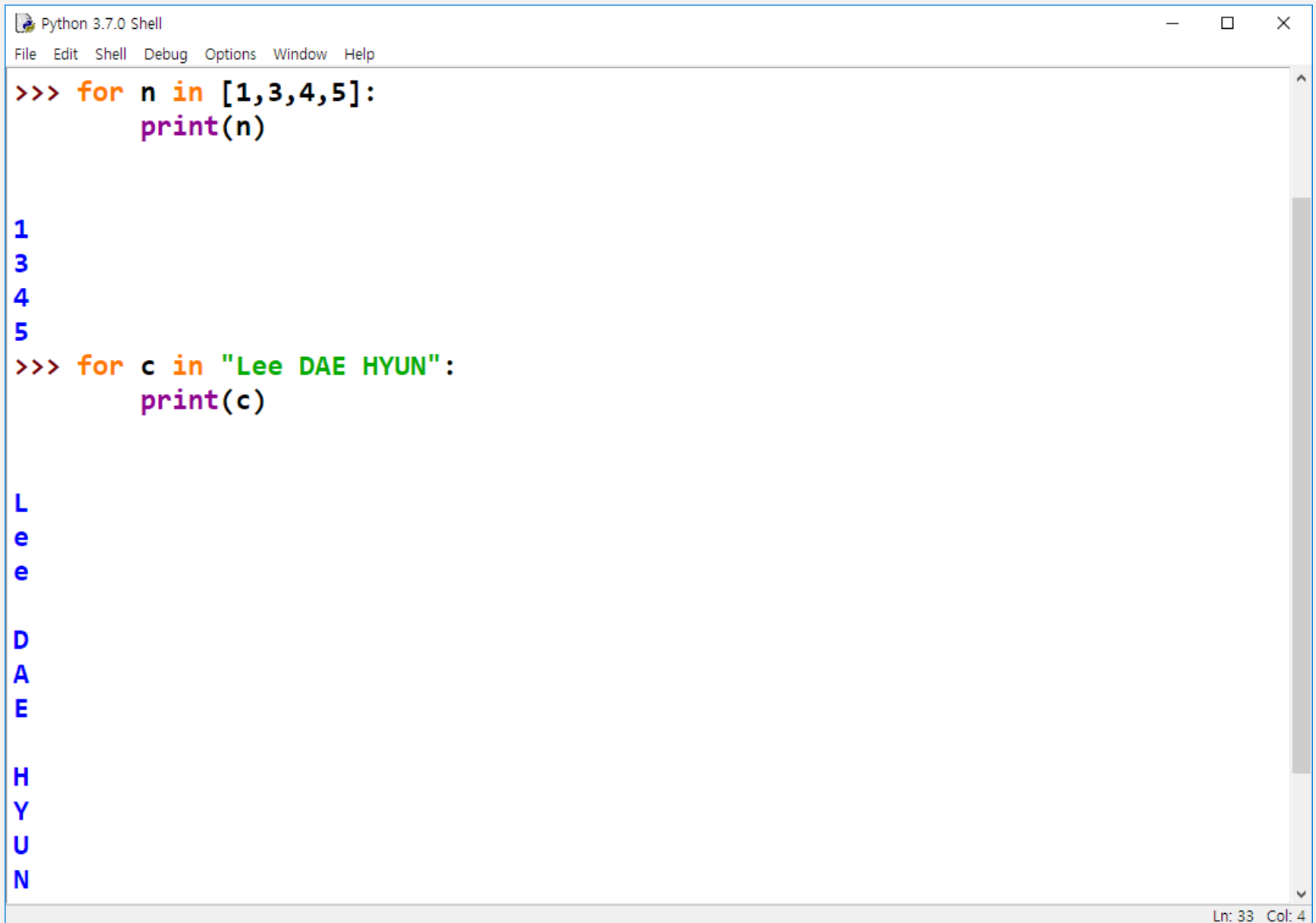
turtle.shape('turtle')
while (True):
    turtle.setheading(random.randint(0, 360))
    turtle.forward(random.randint(100, 200))
    turtle.stamp()

Ln: 8 Col: 18
```

문법: for 반복문

- 집합적 데이터의 각 요소를 하나씩 꺼내서 반복적으로 처리

```
for 변수 in 리스트(또는 튜플, 문자열):  
    수행할 문장1  
    수행할 문장2  
    ...
```



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

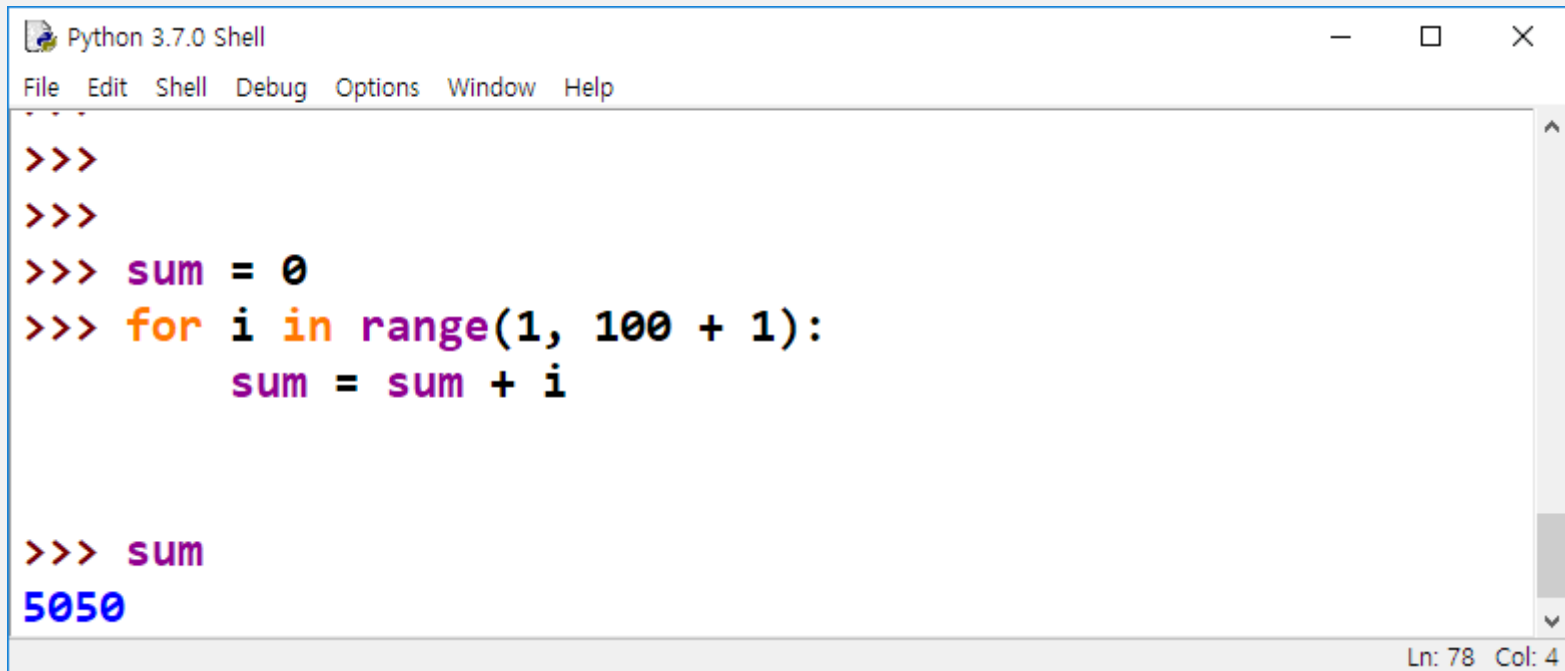
>>> for n in [1,3,4,5]:
    print(n)

1
3
4
5

>>> for c in "Lee DAE HYUN":
    print(c)

L
e
e
D
A
E
H
Y
U
N

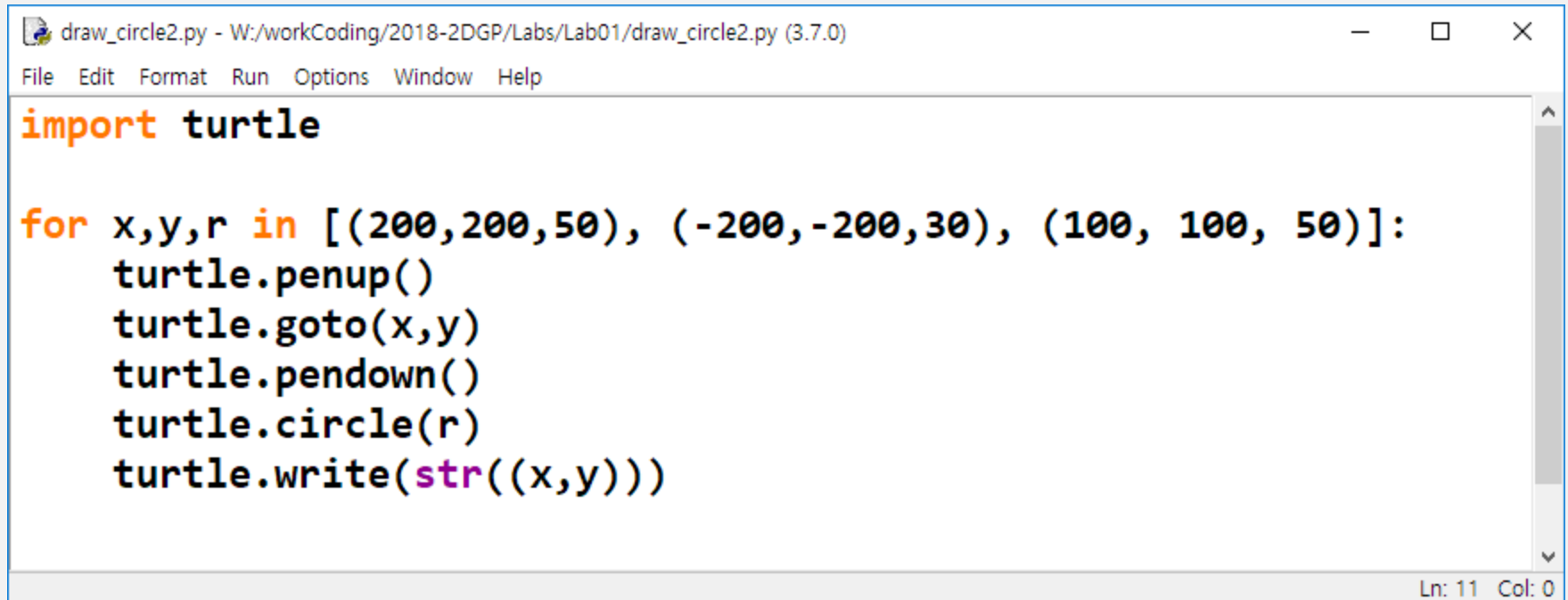
Ln: 33 Col: 4
```



A screenshot of a Python 3.7.0 Shell window. The window has a title bar with the text "Python 3.7.0 Shell" and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a Python script. The script starts with two empty lines, followed by the initialization of a variable 'sum' to 0. Then, a 'for' loop is defined with 'i' in 'range(1, 100 + 1):'. Inside the loop, the statement 'sum = sum + i' is indented. After the loop, the variable 'sum' is printed. The output of the script is '5050'. The status bar at the bottom right of the window shows 'Ln: 78 Col: 4'.

```
>>>  
>>>  
>>> sum = 0  
>>> for i in range(1, 100 + 1):  
    sum = sum + i  
  
>>> sum  
5050
```

Ln: 78 Col: 4

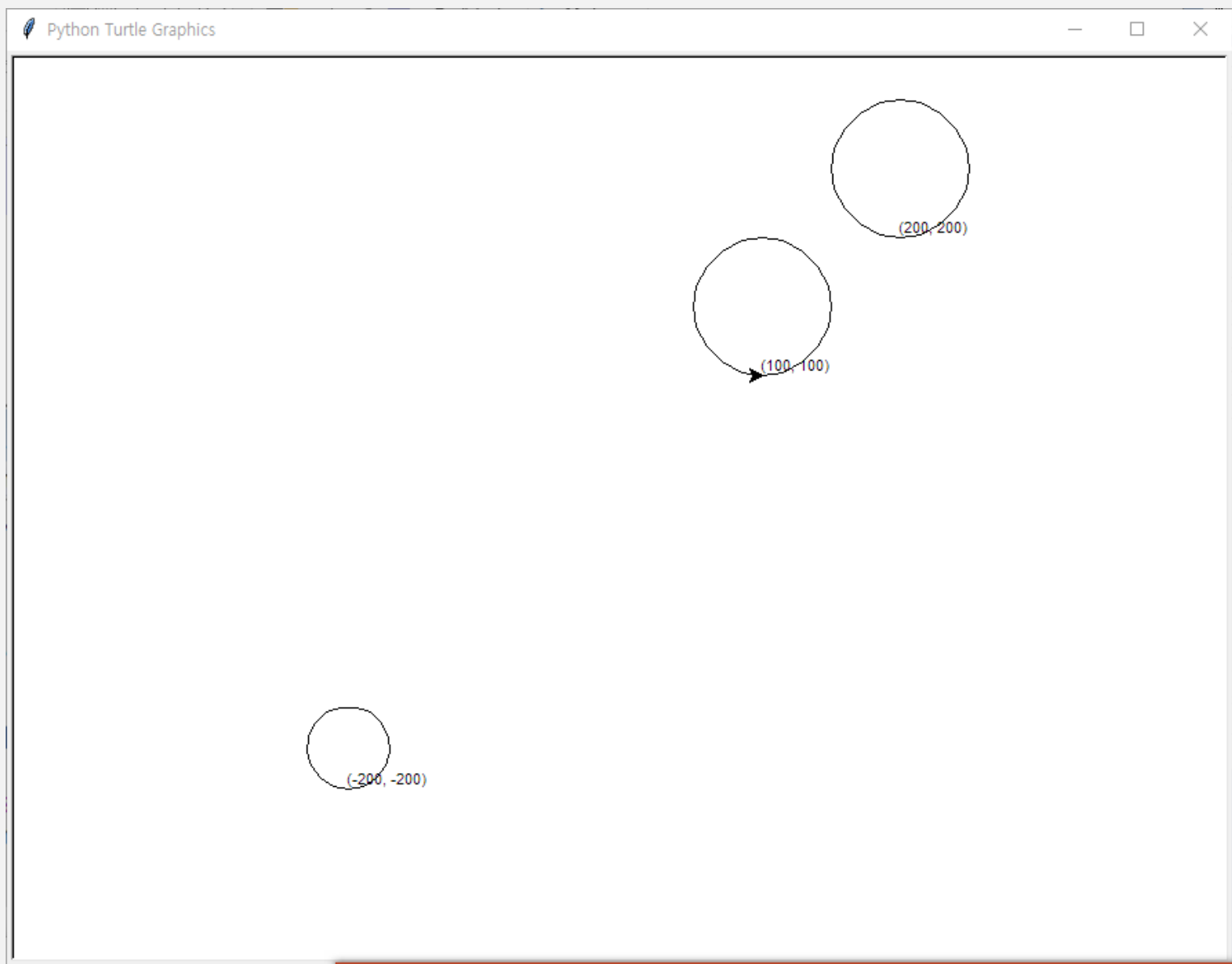


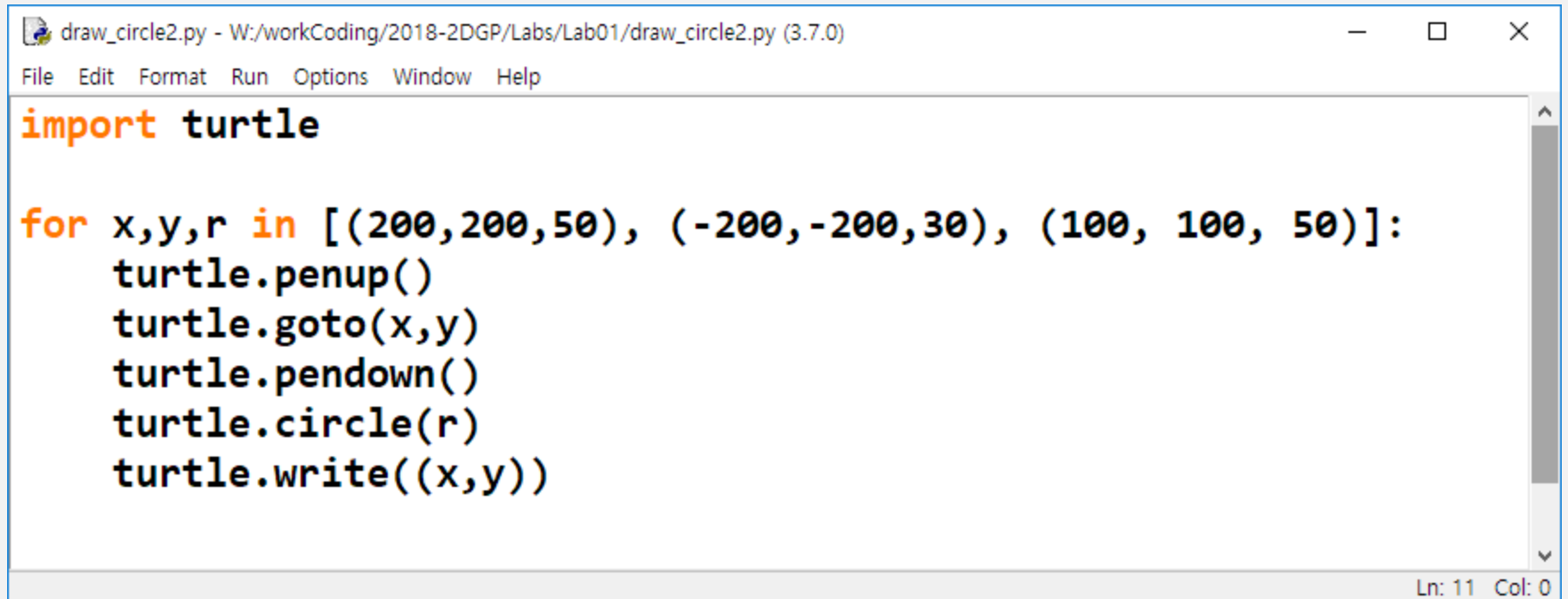
The image shows a screenshot of a Python IDE window. The title bar reads "draw_circle2.py - W:/workCoding/2018-2DGP/Labs/Lab01/draw_circle2.py (3.7.0)". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The main text area contains the following Python code:

```
import turtle

for x,y,r in [(200,200,50), (-200,-200,30), (100, 100, 50)]:
    turtle.penup()
    turtle.goto(x,y)
    turtle.pendown()
    turtle.circle(r)
    turtle.write(str((x,y)))
```

The status bar at the bottom right indicates "Ln: 11 Col: 0".





The image shows a screenshot of a Python IDE window. The title bar reads "draw_circle2.py - W:/workCoding/2018-2DGP/Labs/Lab01/draw_circle2.py (3.7.0)". The menu bar includes "File", "Edit", "Format", "Run", "Options", "Window", and "Help". The code editor contains the following Python code:

```
import turtle

for x,y,r in [(200,200,50), (-200,-200,30), (100, 100, 50)]:
    turtle.penup()
    turtle.goto(x,y)
    turtle.pendown()
    turtle.circle(r)
    turtle.write((x,y))
```

The status bar at the bottom right indicates "Ln: 11 Col: 0".

함수(function)

- 수학에서 함수는, 어떤 수식을 정의한 것.

$$f(a, b) = a + b$$

$$f(3,4)=?$$

$$f(100,10)=?$$

프로그래밍에서 함수(function)란?

- 어떤 특정한 일을 처리하는 기능을 모아놓은 것, 수학적인 함수도 구현 가능.
- 일반적으로 라이브러리, 모듈은 여러 개의 함수들로 구성됨.
- 프로그래머는 자기만의 함수를 만들 수 있음.
- 함수의 이름은 그 함수의 기능을 정확히 나타내는 것이 좋음.

```
turtle.forward(100)  
turtle.right(90)  
turtle.undo()
```

문법: 함수 정의 - 함수를 만들기

```
def 함수명(매개변수):  
    <수행할 문장1>  
    <수행할 문장2>  
    ...
```

add 함수 만들기

a와 b 두개의 값을 받아서,

```
def add(a, b):  
    sum = a + b  
    return sum
```

a와 b를 더해, sum을 계산

sum의 값을 되돌려줌(return)

함수 정의임을 뜻함.

함수 이름

인수: 외부에서 전달되는 값

```
def add(a, b):  
    sum = a + b  
    return sum
```

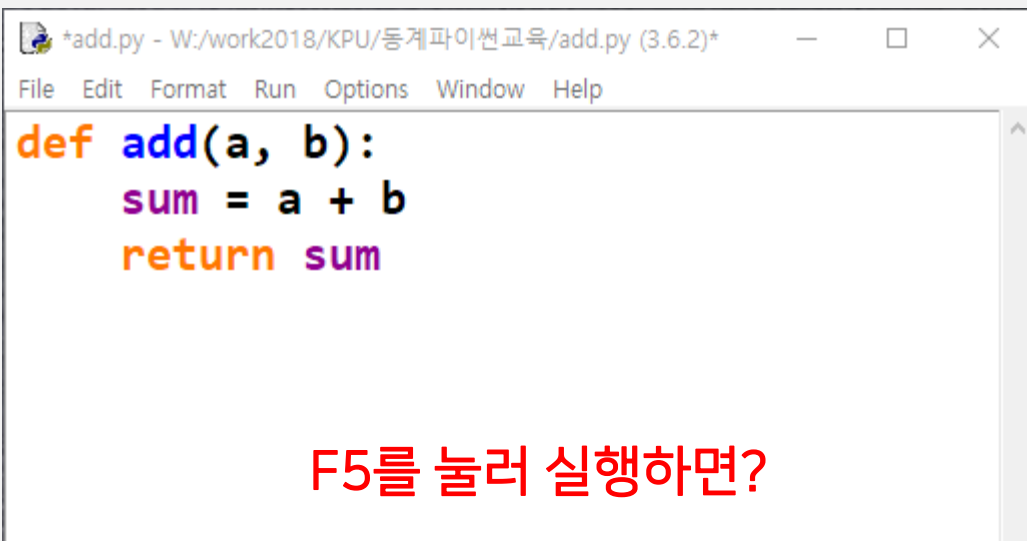
들여쓰기(indentation)

*** 매우 중요 ***

함수가 호출될 때 실행됨.

함수 정의, 그 자체로는 실행되지 않음.

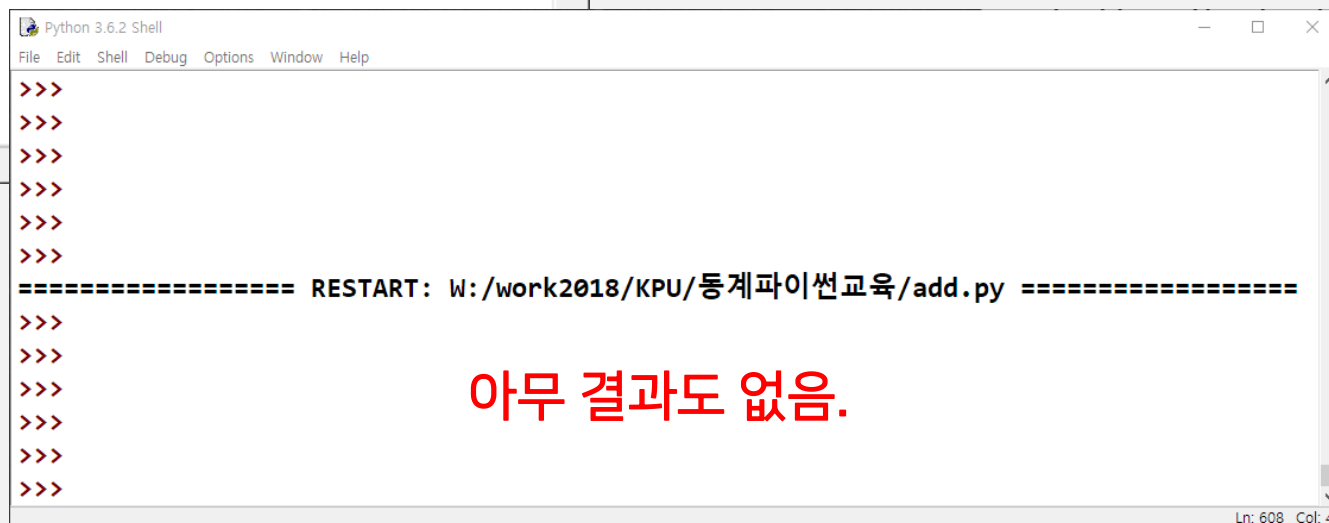
add.py



A screenshot of a text editor window titled '*add.py - W:/work2018/KPU/등계파이썬교육/add.py (3.6.2)*'. The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code content is as follows:

```
def add(a, b):  
    sum = a + b  
    return sum
```

F5를 눌러 실행하면?



A screenshot of a Python 3.6.2 Shell window titled 'Python 3.6.2 Shell'. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell shows several empty prompt lines (">>>>") followed by a restart message:

```
==== RESTART: W:/work2018/KPU/등계파이썬교육/add.py =====  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>
```

아무 결과도 없음.

함수를 실행하려면, 함수 호출을 해야 함.

add.py

```
*add.py - W:/work2018/KPU/동계파이썬교육/add.py (3.6.2)*
File Edit Format Run Options Window Help

def add(a, b):
    sum = a + b
    return sum

result = add(100, 10)
print(result)
```

함수 호출

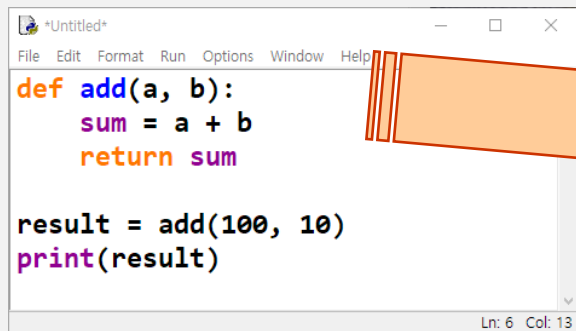
F5를 눌러 실행하면?

```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help

>>>
>>>
>>>
>>>
>>>
>>>
>>>
===== RESTART: W:/work2018/KPU/동계파이썬교육/add.py =====
110
>>>
>>>
>>>
>>>
>>>
```

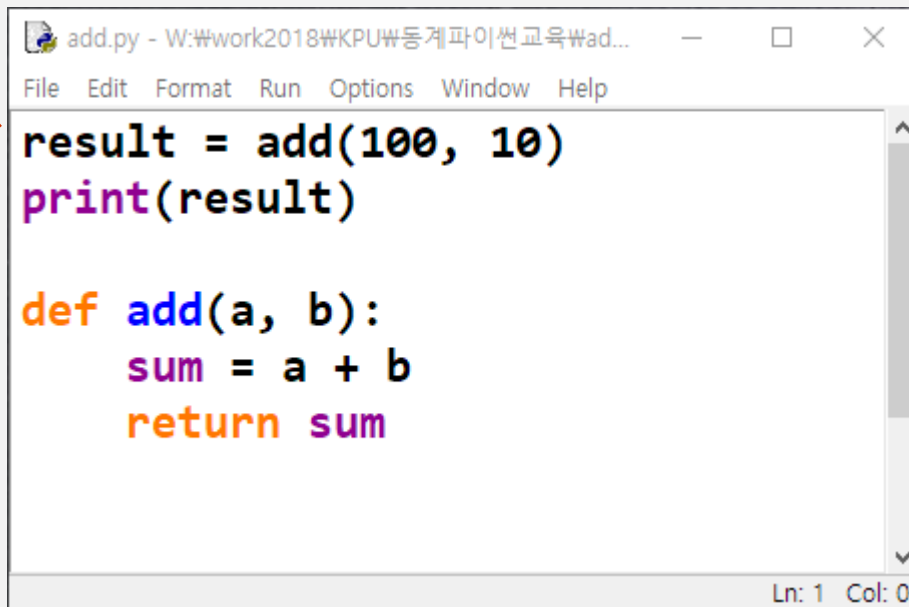
결과가 출력됨.

함수를 호출하려면, 함수 정의가 먼저 되어 있어야 함.



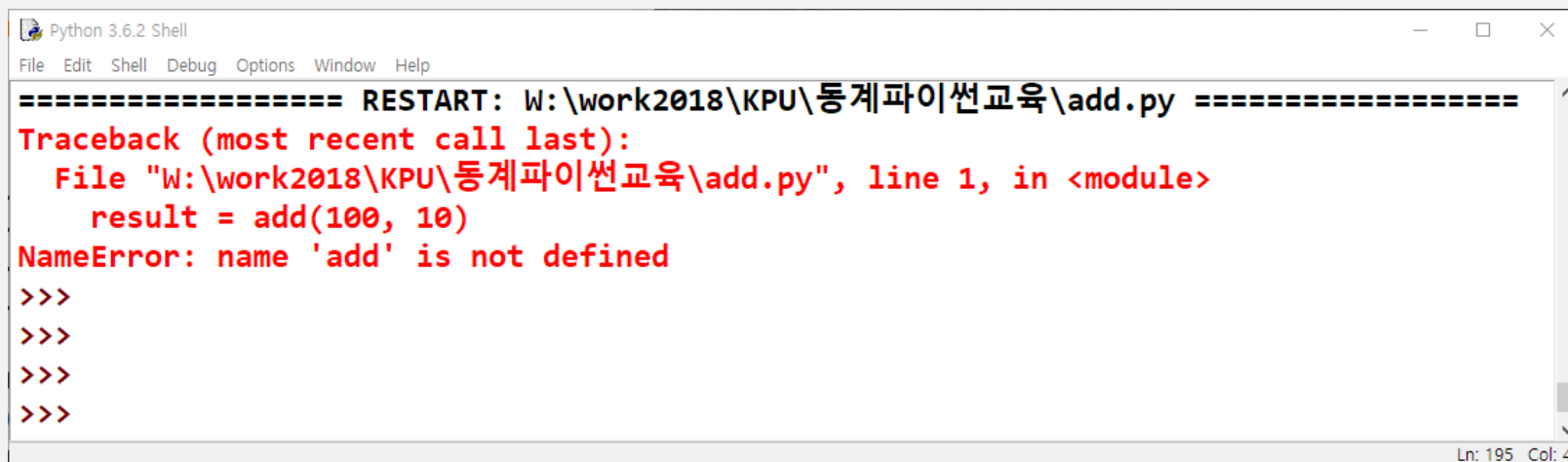
```
def add(a, b):  
    sum = a + b  
    return sum  
  
result = add(100, 10)  
print(result)
```

Ln: 6 Col: 13



```
result = add(100, 10)  
print(result)  
  
def add(a, b):  
    sum = a + b  
    return sum
```

Ln: 1 Col: 0



```
Python 3.6.2 Shell  
File Edit Shell Debug Options Window Help  
===== RESTART: W:\work2018\KPU\통계파이썬교육\add.py =====  
Traceback (most recent call last):  
  File "W:\work2018\KPU\통계파이썬교육\add.py", line 1, in <module>  
    result = add(100, 10)  
NameError: name 'add' is not defined  
>>>  
>>>  
>>>  
>>>
```

Ln: 195 Col: 4

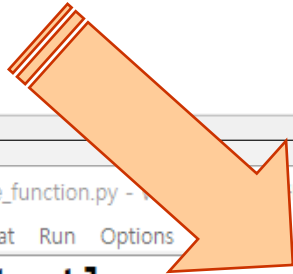
함수는 여러 작업을 모아서 하나로 처리할 수 있게 해 줘.

```
drunken_turtle.py - W:\work2018\KPU\등계\파이썬교육\drunken_turtle.py (3.6.2)
File Edit Format Run Options Window Help

import turtle
import random

turtle.shape('turtle')
while (True):
    turtle.setheading(random.randint(0, 360))
    turtle.forward(random.randint(100,200))
    turtle.stamp()
```

Ln: 8 Col: 18



```
random_turtle_function.py - W:\work2018\KPU\등계\파이썬교육\random_turtle_function.py (3.6.2)
File Edit Format Run Options Help

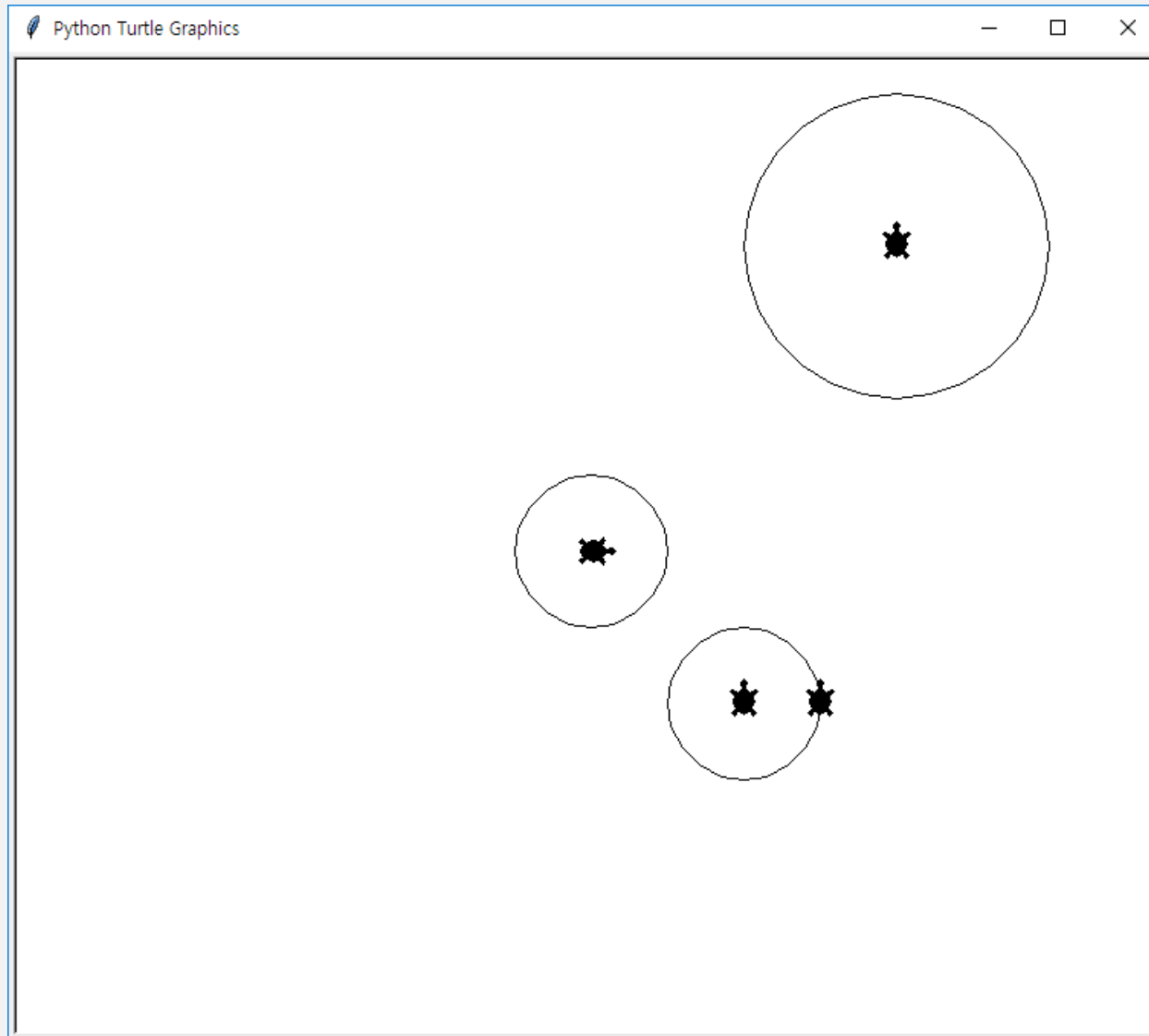
import turtle
import random

def drunken_move():
    turtle.setheading(random.randint(0, 360))
    turtle.forward(random.randint(100,200))
    turtle.stamp()

turtle.shape('turtle')
while (True):
    drunken_move()
```

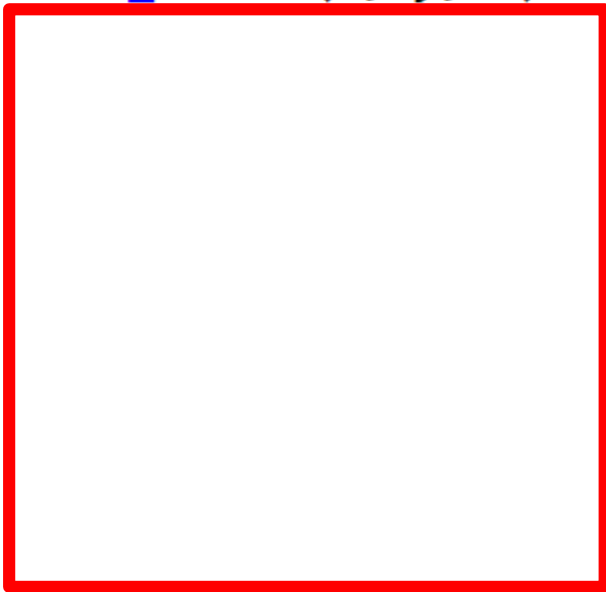
Ln: 7 Col: 18

Quiz #3: 지정해주는 위치를 중심으로 원을 그리는 함수 만들기



```
*draw_circle.py - W:/work2018/KPU/등계파이썬교육/draw_circle.py ...
File Edit Format Run Options Window Help

import turtle

def draw_circle(x, y, r):
    

turtle.shape("turtle")
draw_circle(0, 0, 50)
draw_circle(200, 200, 100)
draw_circle(100, -100, 50)

Ln: 15 Col: 0
```

```
*draw_circle.py - W:/work2018/KPU/등계파이썬교육/draw_circle.py ...
File Edit Format Run Options Window Help

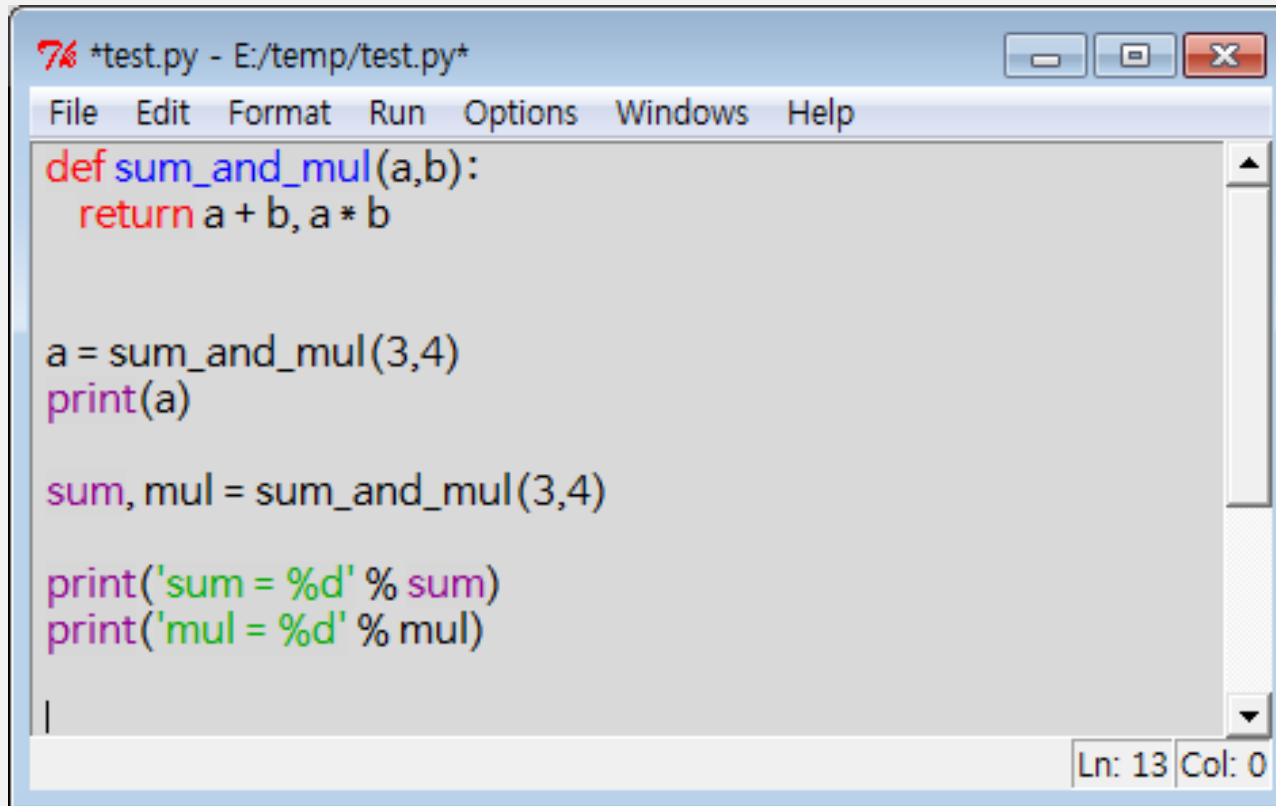
import turtle

def draw_circle(x, y, r):
    turtle.penup()
    turtle.goto(x, y)
    turtle.pendown()
    turtle.stamp()
    turtle.penup()
    turtle.goto(x+r, y)
    turtle.setheading(90)
    turtle.pendown()
    turtle.circle(r)
    turtle.penup()

turtle.shape("turtle")
draw_circle(0, 0, 50)
draw_circle(200, 200, 100)
draw_circle(100, -100, 50)
```

Ln: 15 Col: 0

여러 개의 return 값 가능

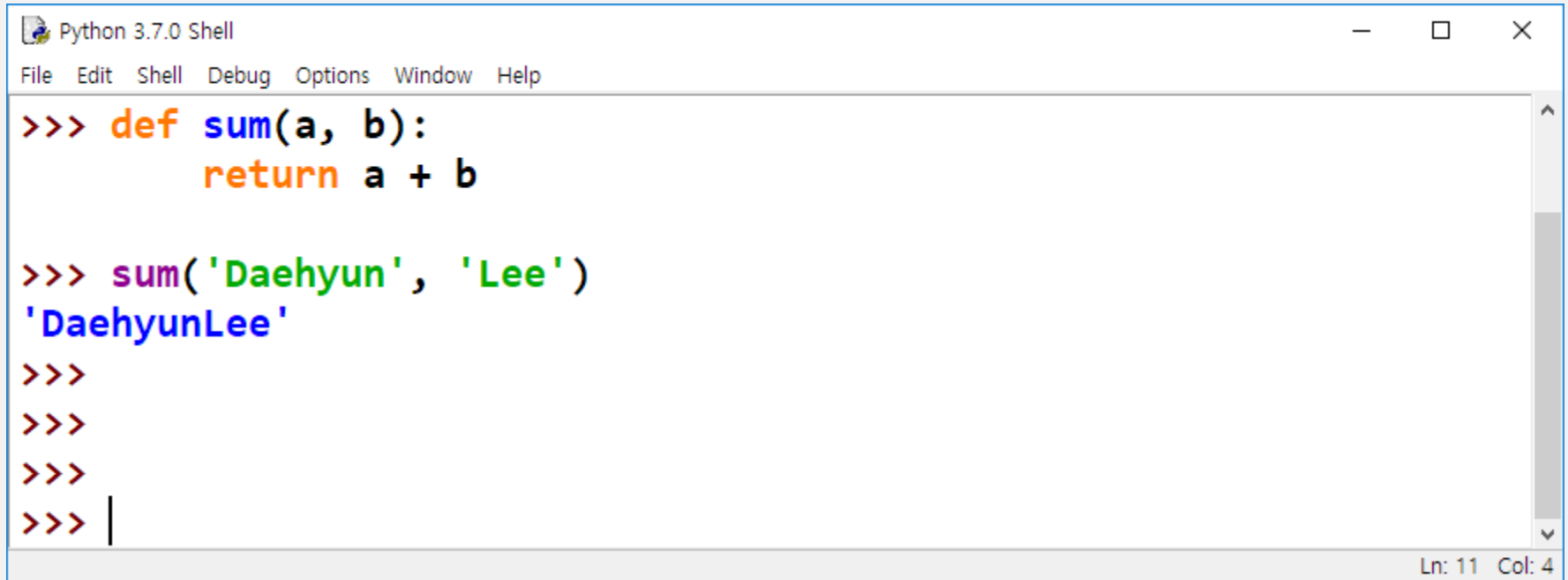


The screenshot shows a Python IDE window titled "7% *test.py - E:/temp/test.py*". The window has a menu bar with "File", "Edit", "Format", "Run", "Options", "Windows", and "Help". The code editor contains the following Python code:

```
def sum_and_mul(a,b):  
    return a + b, a * b  
  
a = sum_and_mul(3,4)  
print(a)  
  
sum, mul = sum_and_mul(3,4)  
  
print('sum = %d' % sum)  
print('mul = %d' % mul)  
|
```

The status bar at the bottom right indicates "Ln: 13 Col: 0".

인자의 타입에 따라 자동으로 연산 기능이 결정



```
Python 3.7.0 Shell
File Edit Shell Debug Options Window Help

>>> def sum(a, b):
        return a + b

>>> sum('Daehyun', 'Lee')
'DaehyunLee'
>>>
>>>
>>>
>>> |
```

Ln: 11 Col: 4

심화 개별 학습

거북이를 키 입력을 통해서 조정하기

- onkey() 함수를 이용하여, 키 입력에 따라 반응하는 함수를 연결.
- listen() 함수를 이용해서, 거북이가 키 입력을 확인할 수 있게 함.

move 라는 이름의 함수가 호출됨.

`turtle.onkey(move, 'w')`

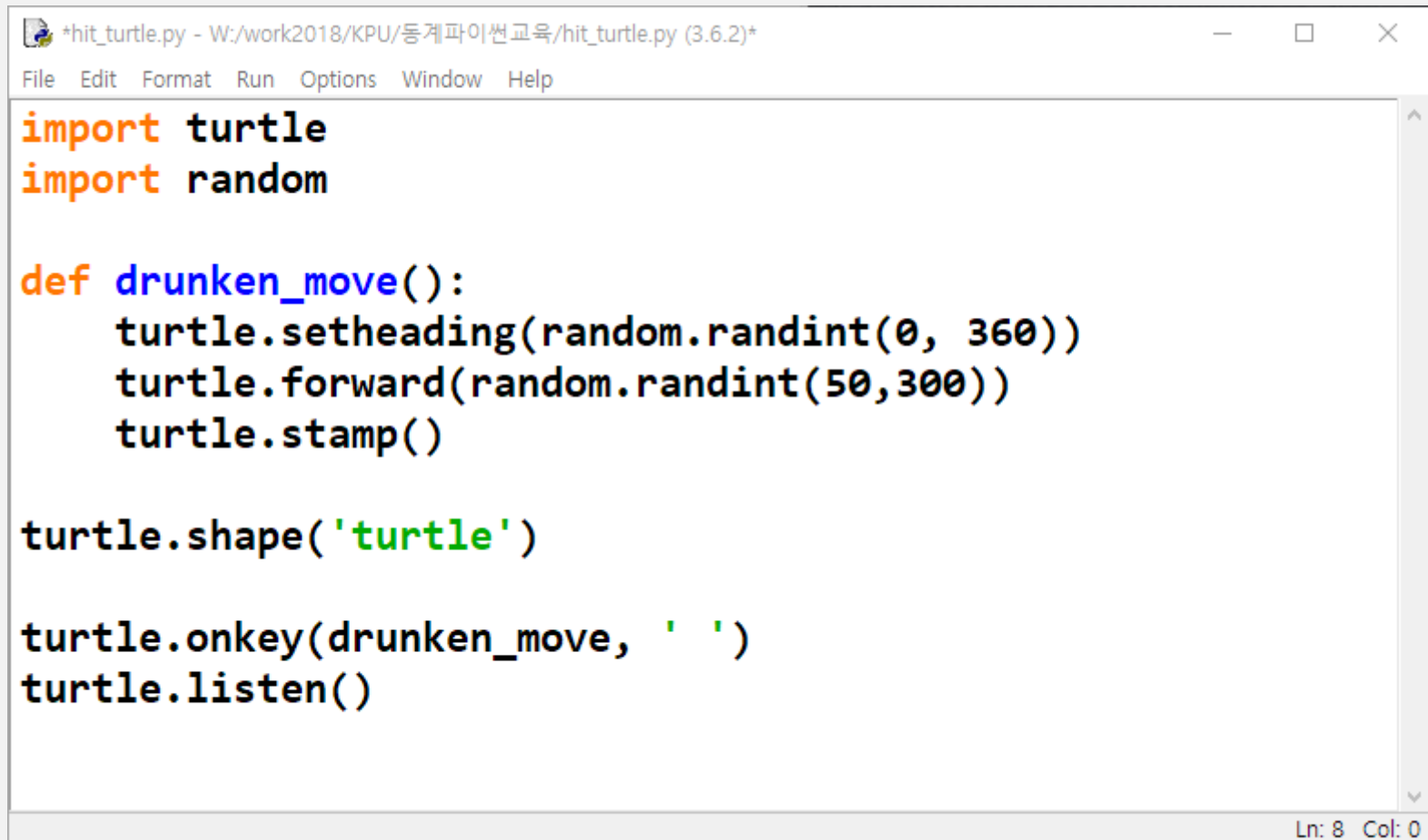
`turtle.listen()`

w 키를 누르면,

거북이가 키 입력을 들을 수 있게 함.

기호	뜻
'w'	w 키
'a'	a 키
's'	s 키
'd'	d 키
' '	스페이스 키
'Escape'	ESC 키

거북이 채찍질하기



The image shows a screenshot of a Python IDE window titled "*hit_turtle.py - W:/work2018/KPU/동계파이썬교육/hit_turtle.py (3.6.2)*". The window contains the following Python code:

```
import turtle
import random

def drunken_move():
    turtle.setheading(random.randint(0, 360))
    turtle.forward(random.randint(50, 300))
    turtle.stamp()

turtle.shape('turtle')

turtle.onkey(drunken_move, ' ')
turtle.listen()
```

The status bar at the bottom right of the window indicates "Ln: 8 Col: 0".

ESC 키를 누르면 다시 시작



```
*hit_turtle.py - W:/work2018/KPU/동계파이썬교육/hit_turtle.py (3.6.2)*
File Edit Format Run Options Window Help

import turtle
import random

def drunken_move():
    turtle.setheading(random.randint(0, 360))
    turtle.forward(random.randint(50, 100))
    turtle.stamp()

def restart():
    turtle.reset()

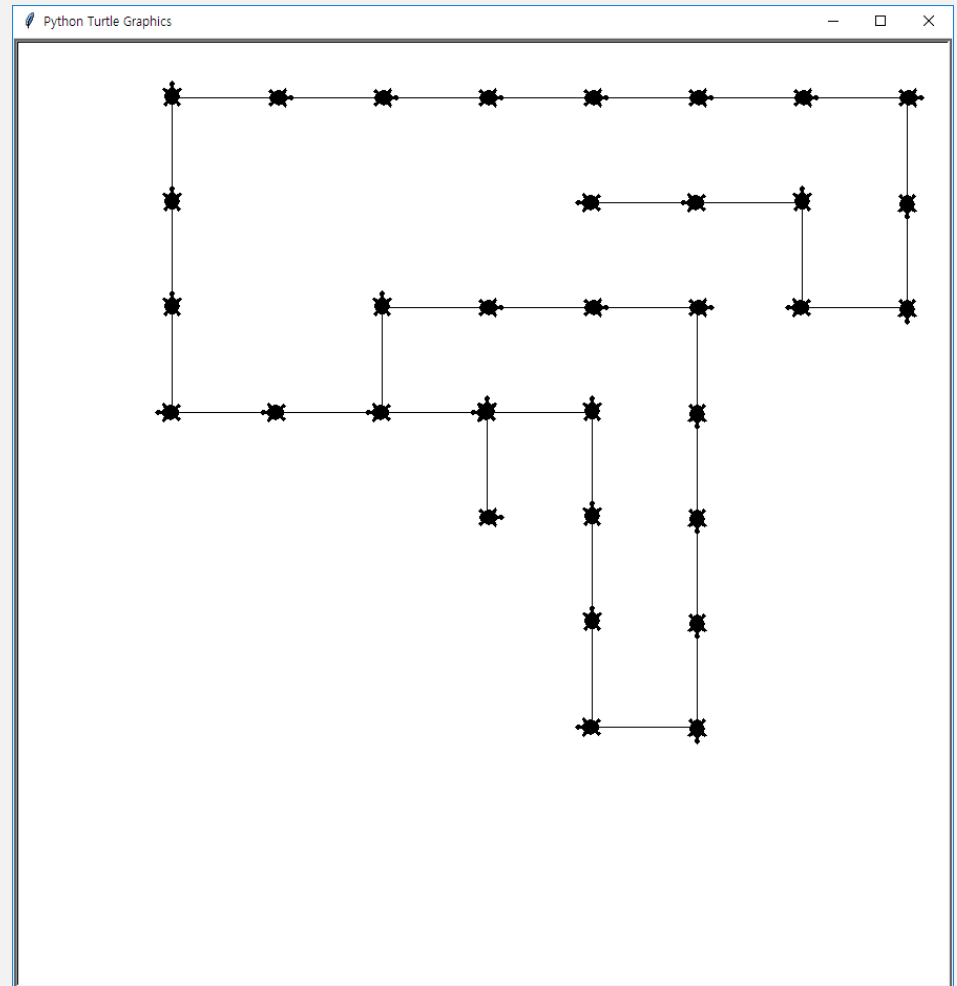
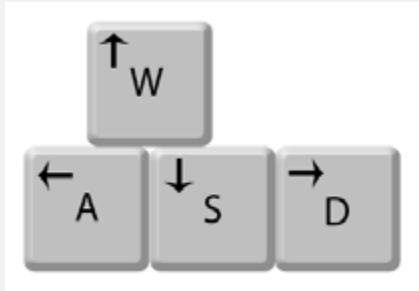
turtle.shape('turtle')

turtle.onkey(drunken_move, ' ')
turtle.onkey(restart, 'Escape')
turtle.listen()

Ln: 6 Col: 37
```

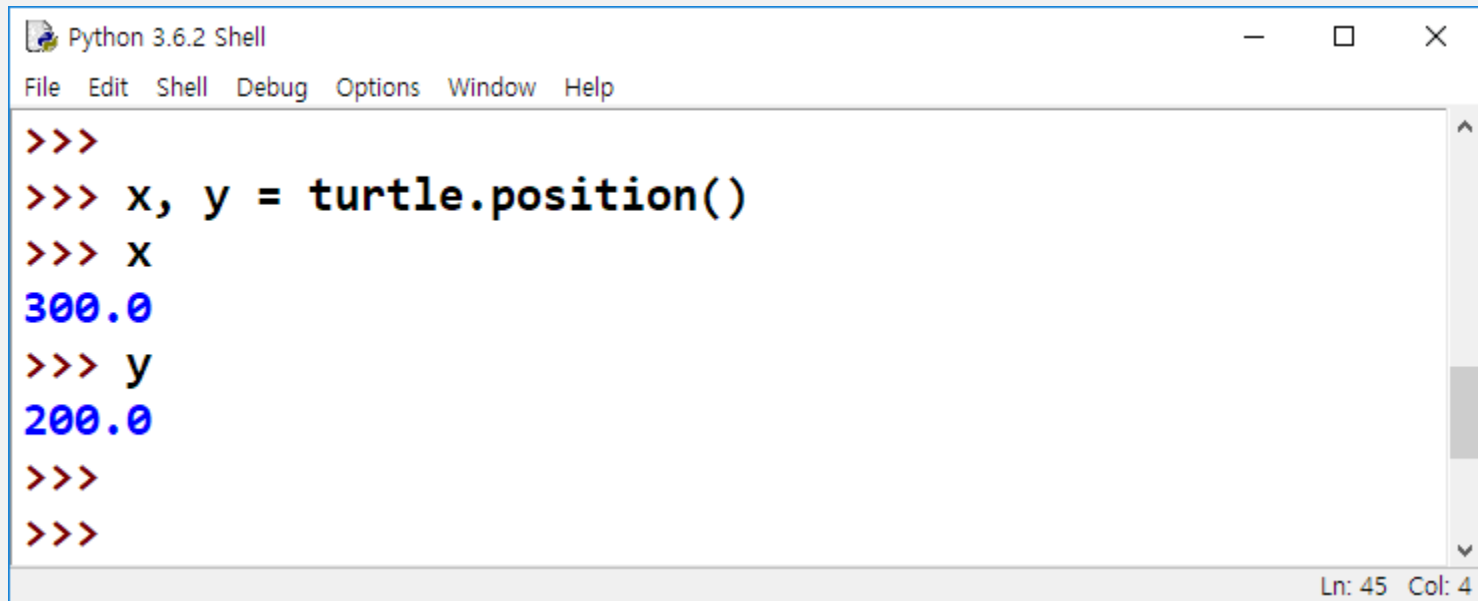
과제#1: 거북이를 “WASD”키로 조종하기

- WASD를 이용하여, 거북이를 상하좌우로 이동할 수 있음.한번 이동 거리는 50포인트
- ESC 키를 누르면 처음부터 다시 시작함.



심화과제: 원 따먹기 게임

- 무작위 위치에 원이 만들어짐.
- 거북이를 이동시켜서 원에 닿으면, 다시 게임 시작
- global 변수를 쓸 줄 알아야 함.
- 거북이의 현재 위치는 `turtle.position()`으로 알아낼 수 있음.



```
Python 3.6.2 Shell
File Edit Shell Debug Options Window Help
>>>
>>> x, y = turtle.position()
>>> x
300.0
>>> y
200.0
>>>
>>>
```

Ln: 45 Col: 4