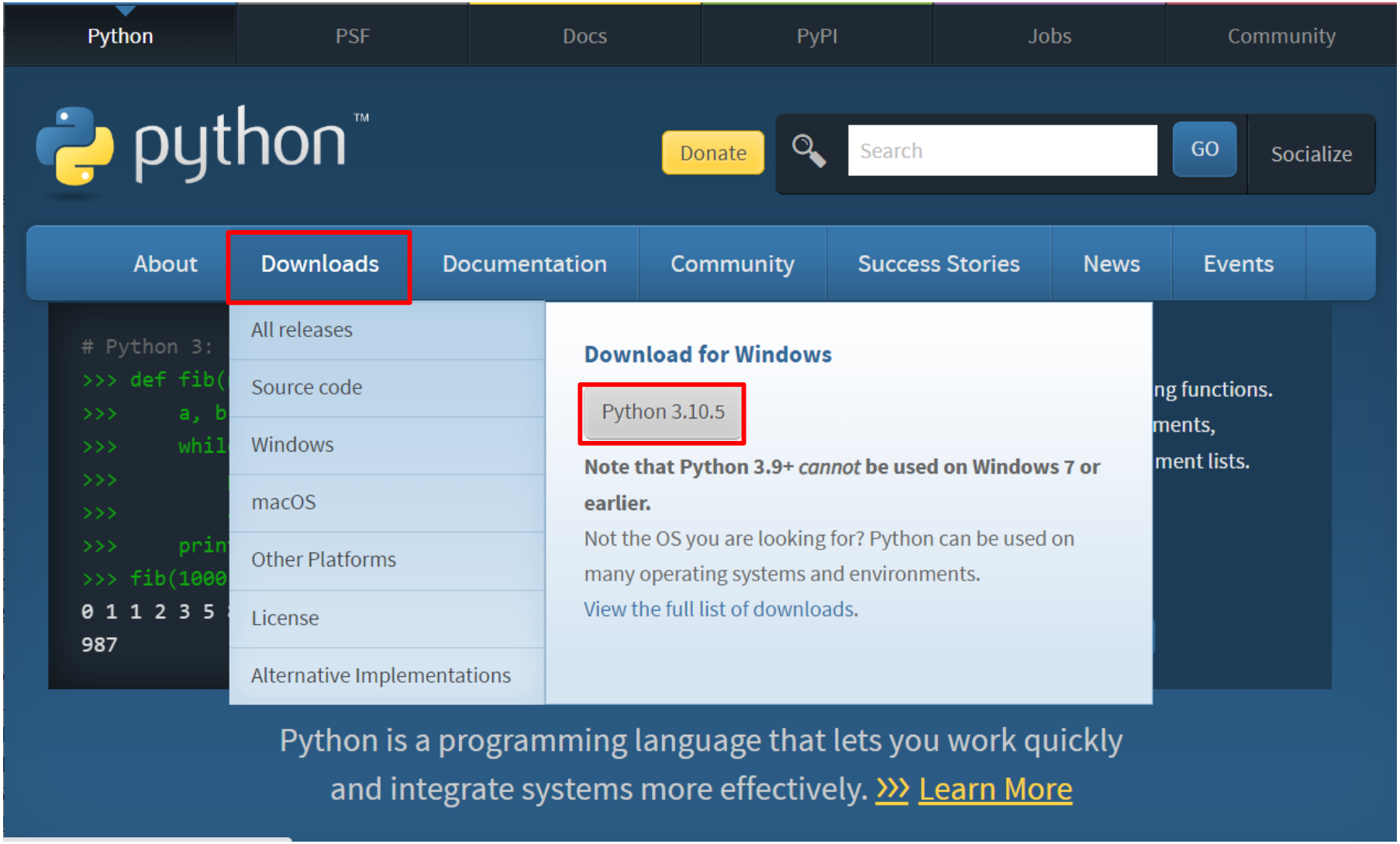
2. 빅데이터처리를 위한 파이썬 기초 01

파이썬 소개 및 설치

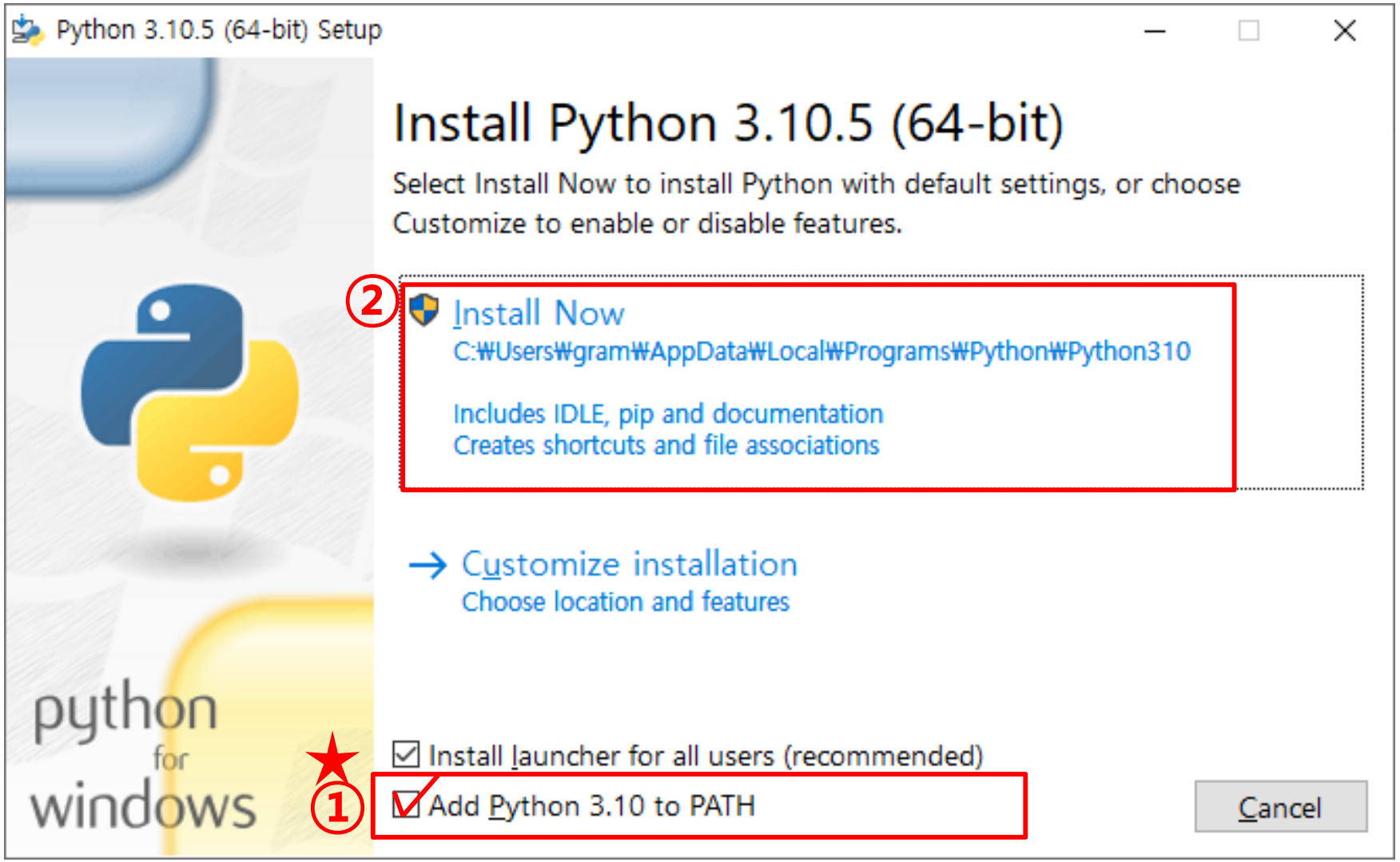
파이썬 소개

- 1990년대 초 등장
- 오픈 소스 프로그램
- 풍부한 라이브러리
- 다른 언어와의 호환성
- 범용 프로그래밍 언어
- 매우 직관적이다
- 초보자가 배우기 쉽다
- 객체 지향 언어
- 지속적인 발전

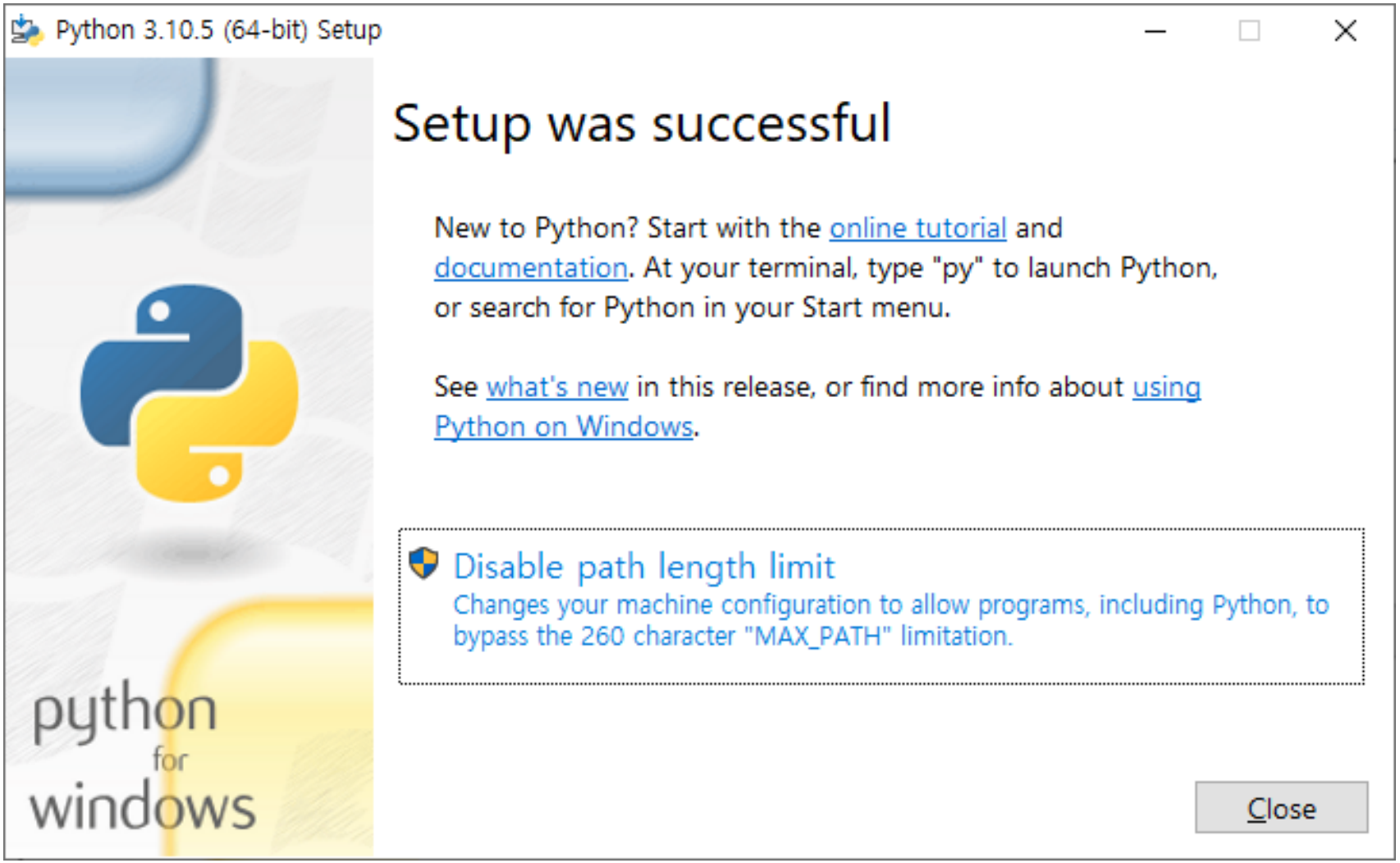
파이썬 설치 http://www.python.org/



파이썬 설치


Python 3.10.5 (64-bit) Setup
Install Python 3.10.5 (64-bit)
Select Install Now to install Python with default settings, or choose
Customize to enable or disable features.
②
Install Now
C:₩Users₩gram₩AppData₩Local₩Programs₩Python₩Python310
Includes IDLE, pip and documentation
Creates shortcuts and file associations
Customize installation
Choose location and features
python
for
windows
Install launcher for all users (recommended)
①
Add Python 3.10 to PATH
Cancel

파이썬 설치


Python 3.10.5 (64-bit) Setup
Setup was successful
New to Python? Start with the online tutorial and documentation. At your terminal, type "py" to launch Python, or search for Python in your Start menu.
See what's new in this release, or find more info about using Python on Windows.
Disable path length limit
Changes your machine configuration to allow programs, including Python, to bypass the 260 character "MAX_PATH" limitation.
python
for
windows
Close

통합개발환경

IDLE : Integrated Development Environment

- 코드 입력, 편집, 번역, 실행, 저장, 디버깅 등과 같이 프로그램 개발을 위해 필요한 기능들을 통합적으로 제공하는 프로그램

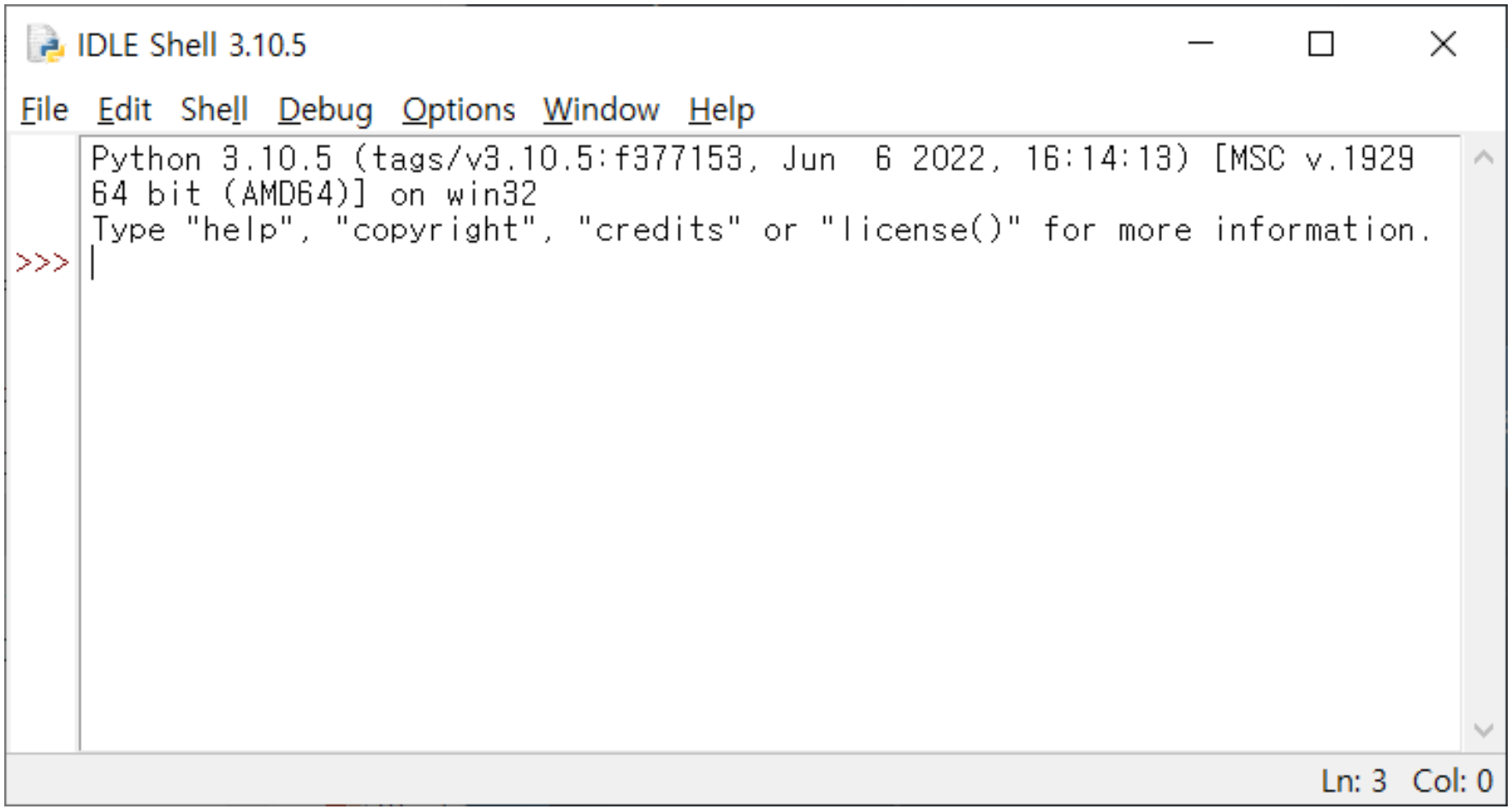
파이썬 버전

- 2.x 버전과 3.x 버전은 호환되지 않는다
- 2.x 의 프로그램은 3.x 인터프리터에서 실행되지 않는다

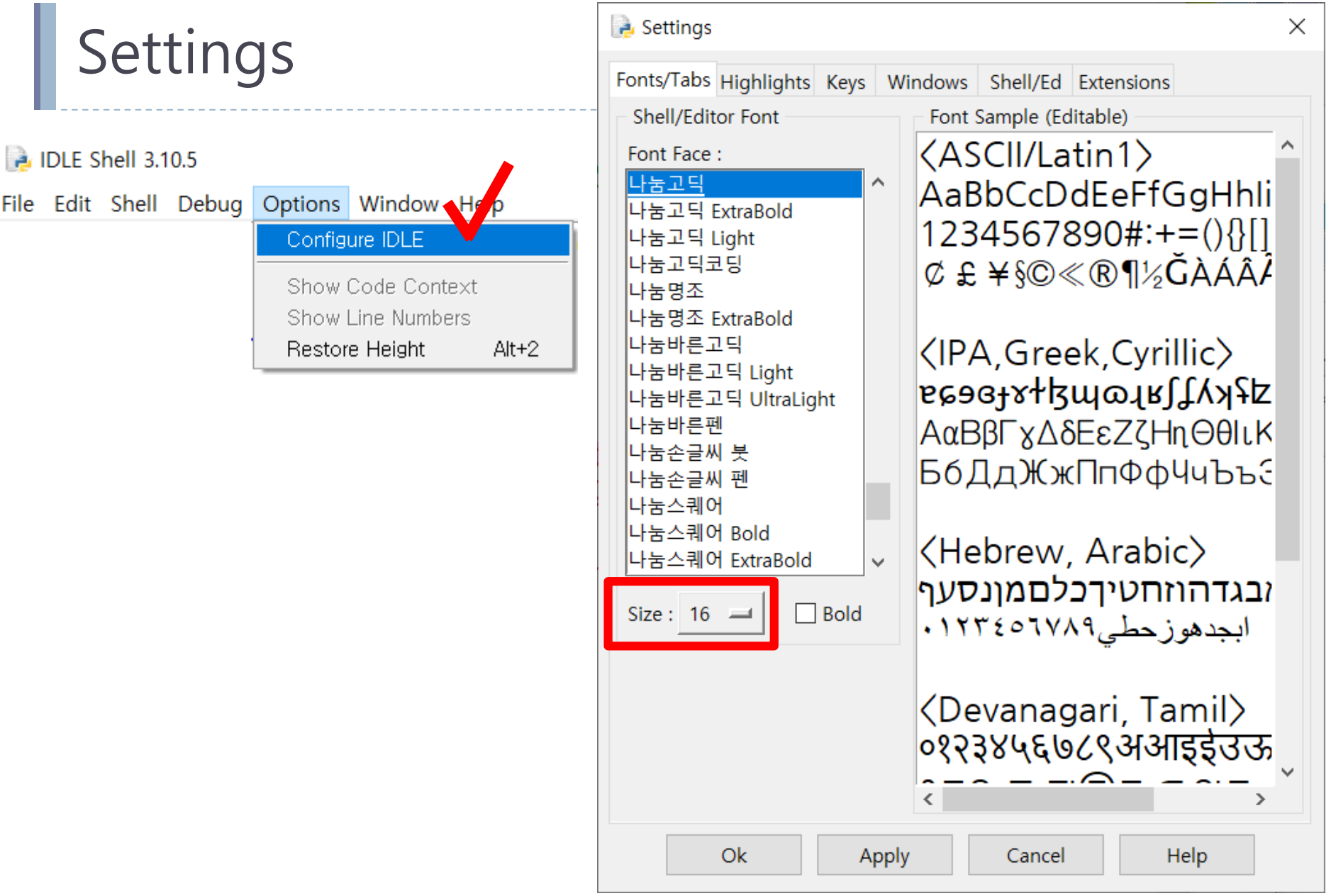
- 이전에 개발된 라이브러리를 사용하려면 2.x 선택
- 미래를 대비하는 차원에서 3.x 선택할 것을 권장

쉘 모드에서 대화식 코딩

쉘 모드의 초기 화면


IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Python 3.10.5 (tags/v3.10.5:f377153, Jun 6 2022, 16:14:13) [MSC v.1929
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
Ln: 3 Col: 0

Settings


IDLE Shell 3.10.5
File Edit Shell Debug Options Window Help
Configure IDLE
Show Code Context
Show Line Numbers
Restore Height Alt+2
Settings
Fonts/Tabs Highlights Keys Windows Shell/Ed Extensions
Shell/Editor Font
Font Face :
나눔고딕
나눔고딕 ExtraBold
나눔고딕 Light
나눔고딕코딩
나눔명조
나눔명조 ExtraBold
나눔바른고딕
나눔바른고딕 Light
나눔바른고딕 UltraLight
나눔바른펜
나눔손글씨 붓
나눔손글씨 펜
나눔스퀘어
나눔스퀘어 Bold
나눔스퀘어 ExtraBold
Size : 16
Bold
Font Sample (Editable)
〈ASCII/Latin1〉
〈IPA,Greek,Cyrillic〉
〈Hebrew, Arabic〉
〈Devanagari, Tamil〉
Ok
Apply
Cancel
Help

문자열 출력하기

문자열 : 큰 따옴표와 작은 따옴표 모두 사용 가능

```
>>> print("Hello World~~")
...
    Hello World~~
>>> print("Hello World~~₩n₩n")
...
    Hello World~~


>>> print("안녕하세요~~₩n₩n")
...
    안녕하세요~~
```

줄 바꿈 문자 "₩n"

수식 결과 출력하기

```
>>> print(2+3)
5
>>> 2+3
5
>>> print(2.5+6.1)
8.6
>>> 2.5+6.1
8.6
>>> print(5+2.7)
7.7
>>> 5/2
2.5
```

수식에서 사용 가능한 연산자

연산자	기능	사용의 예	결과
+	더하기	7 + 4	11
-	빼기	7 - 4	3
*	곱하기	7 * 4	28
/	나누기	7 /4	1
//	몫	7 // 4	1.75
%	나머지	7 % 4	3
**	지수	2**3	8

- 주의!!!

파이썬 버전 2.X에서는 / 연산자의 결과가 정수가 됩니다.

문자열 연결하기

```
>>> print("I" + "am" + "happy.")
Iamhappy.
>>> print("I " + "am " + "happy.")
I am happy.
>>> print("I" , "am" , "happy.")
I am happy.
>>>
```

문자열 반복하여 출력하기

```
>>> print("사랑합니다" * 100)
사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니
다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합
니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑
합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사
랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다
사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니
다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합
니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑
합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사
랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다
사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니
다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합
니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑
합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사랑합니다사
랑합니다사랑합니다사랑합니다사랑합니다사랑합니다
>>>
```

작은 따옴표와 큰 따옴표 혼용할 수 없다!!

Syntax Error : 문법 오류

```
>>> print("안녕~')
...
SyntaxError: unterminated string literal (detected at line 1)
```

함수 이름 뒤에는 반드시 소괄호!!

Syntax Error : 문법 오류

```
>>> print "안녕"
SyntaxError: Missing parentheses in call to 'print'.
Did you mean print(...)?
```

대소문자를 다르게 취급한다!!

Name Error : 이름 오류

```
>>> Print("안녕")
...
Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    Print("안녕")
NameError: name 'Print' is not defined. Did you mean: 'print'?
```

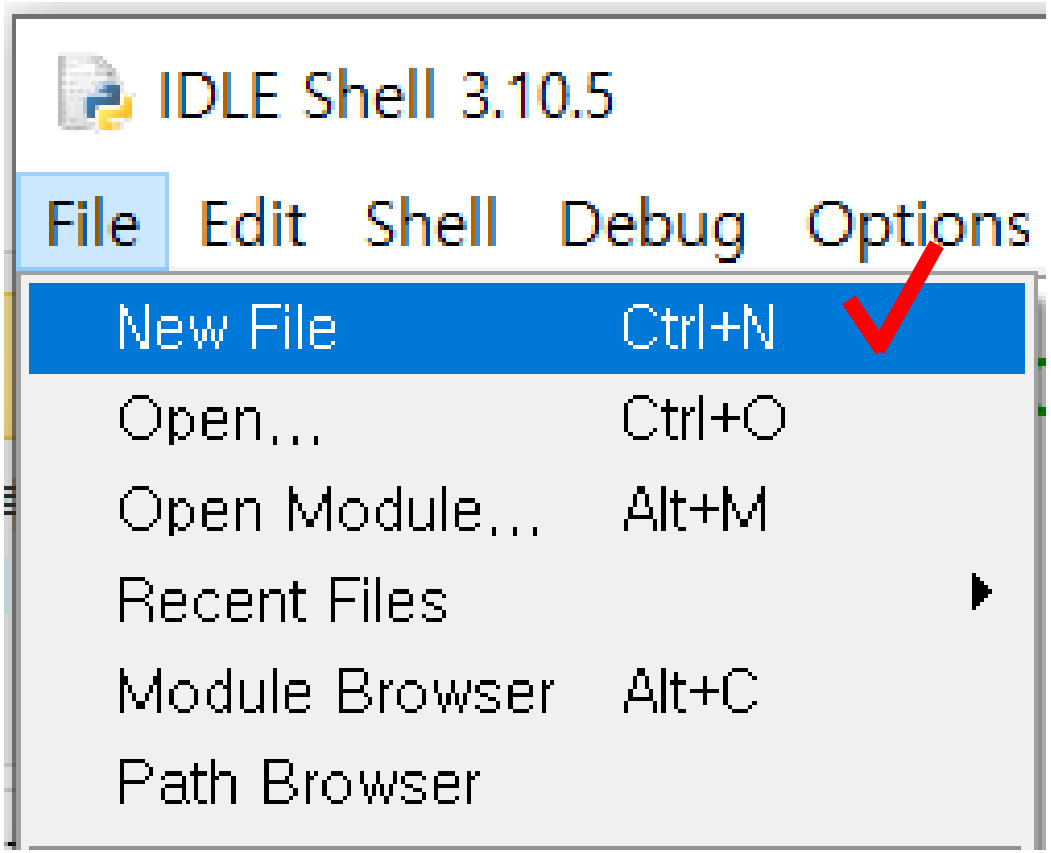
정의되지 않은 이름은 사용할 수 없다!!

Name Error : 이름 오류

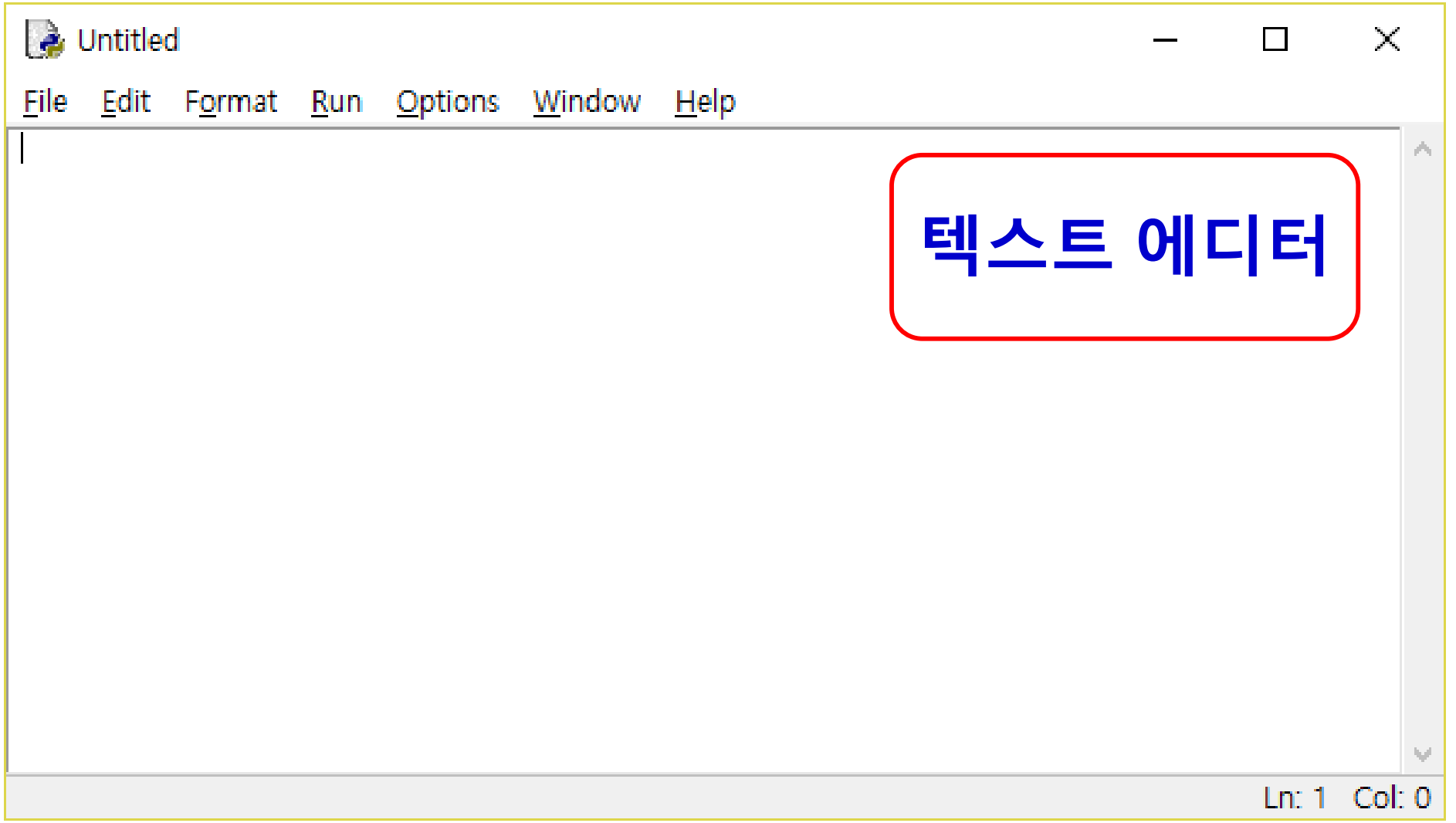
```
>>> print(안녕)
...
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    print(안녕)
NameError: name '안녕' is not defined
>>> 안녕 = "안녕하세요"
...
>>> print(안녕)
...
안녕하세요
```

편집 모드에서 스크립트 작성

스크립트 모드에서 코딩

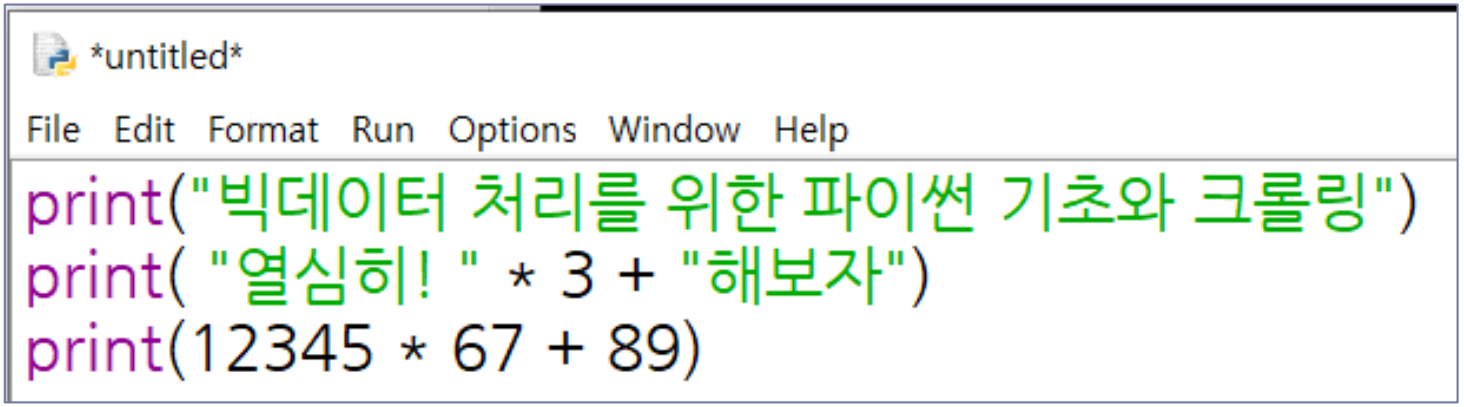

IDLE Shell 3.10.5
File Edit Shell Debug Options
New File Ctrl+N
Open... Ctrl+O
Open Module... Alt+M
Recent Files
Module Browser Alt+C
Path Browser

스크립트 모드에서 코딩

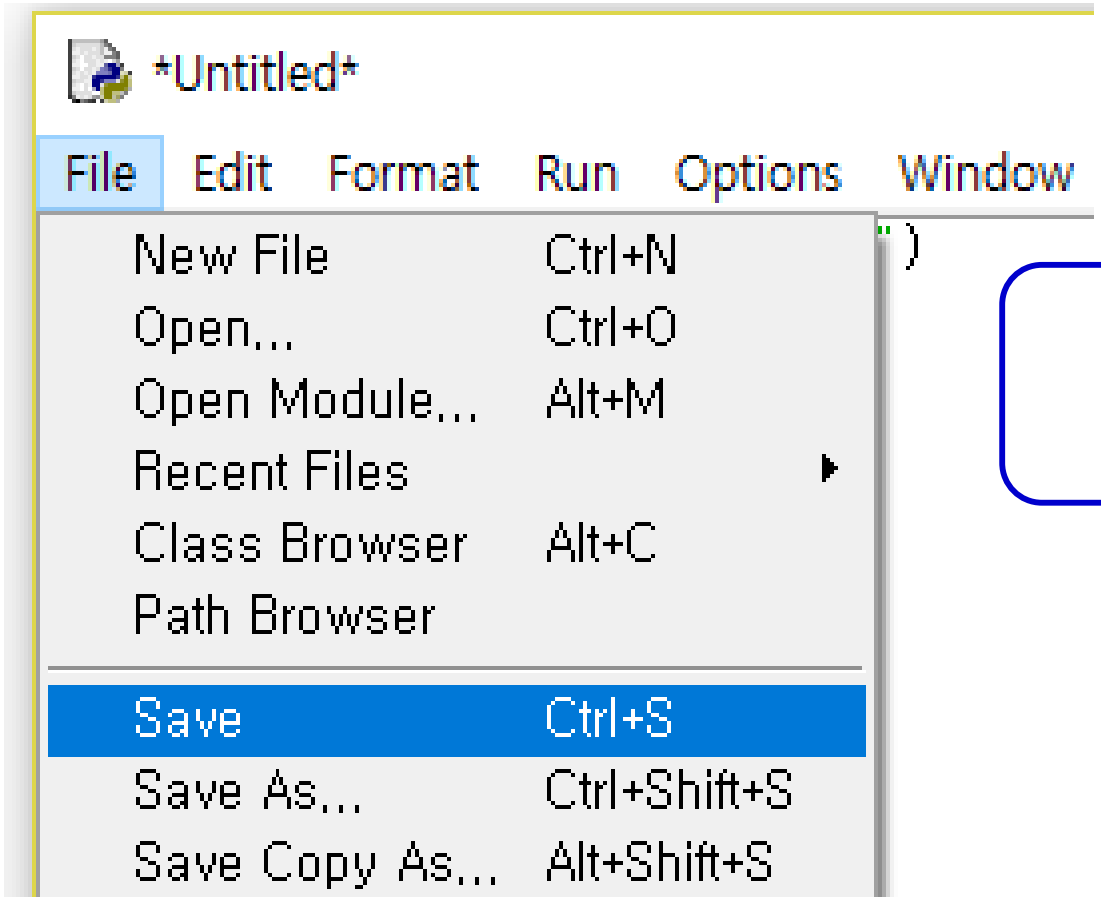

Untitled
File Edit Format Run Options Window Help
텍스트 에디터
Ln: 1 Col: 0

스크립트 모드에서 코딩

- 파이썬 쉘과 다른 점
  - 반드시 저장해야 실행가능!!
  - 저장하면 처음부터 끝까지 모든 코드를 실행



스크립트 모드에서 코딩



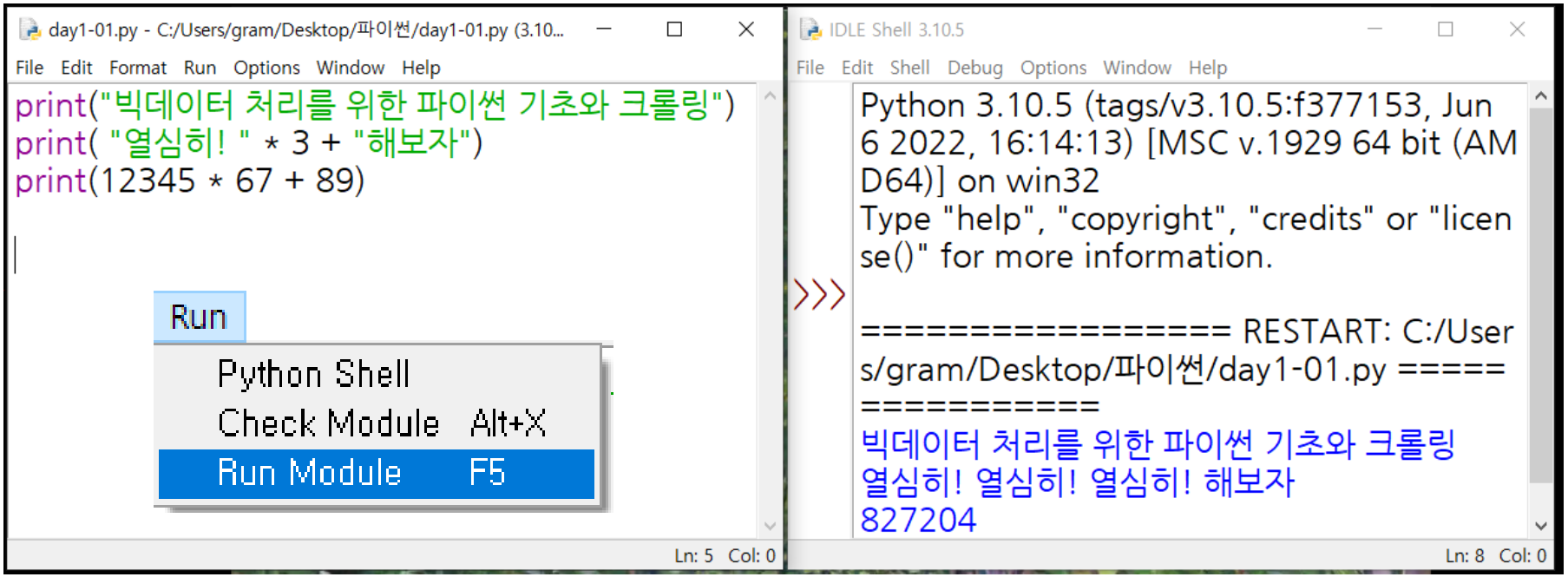
확장자
.py

스크립트 모드에서 코딩/ 결과는 쉘 창에.

- 실행하면 결과는 쉘 창에 나타남

실행 : F5

실행 결과



스크립트 모드에서 코딩

- 대화식 코딩으로 즉각적인 실행 결과를 보려면
  파이썬 쉘에서 작성하고 실행한다

- 재 사용하게 될 긴 프로그램은
  스크립트 모드에서 작성하고 저장 후 실행한다

다음과 같이 출력되도록 코딩 하려면?

이름을 입력하세요: 박정현
국어 점수를 입력하세요: 90
영어 점수를 입력하세요: 80
수학 점수를 입력하세요: 80
박정현 님의 총점은 250 이고 평균은 83.33333333333333 입니다.

* 밑줄(파란색) 부분은 사용자가 입력한 값입니다.

*** 키보드로부터 입력되는 모든 값은
숫자이기 전에 **문자열**이다!!!

변수(Variable)

프로그램의 구성

- 일반적인 프로그램은 외부로부터 데이터를 입력 받아서 데이터를 처리한 후에 결과를 화면에 출력한다.

데이터 입력 데이터 처리 결과 출력

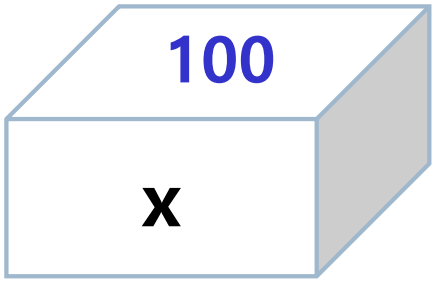
데이터는 변수(variable)에 저장한다

변수명 = 수식

- 변수를 생성하고, 변수는 데이터를 저장하기 위해 컴퓨터의 메모리 공간을 할당 받는다.

```
>>> x = 100
>>>
```

기호 '='으로 표현
=의 왼쪽은 반드시 변수
=의 오른쪽은 어떤한 수식이든 가능

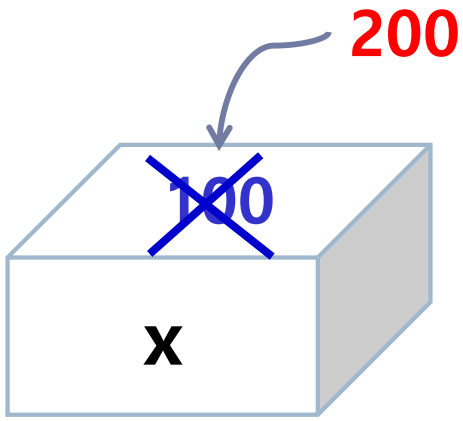


X 라는 변수 이름으로
메모리에 저장되어 있는
100 이라는 데이터에 접근한다

변수의 값은 변경될 수 있다

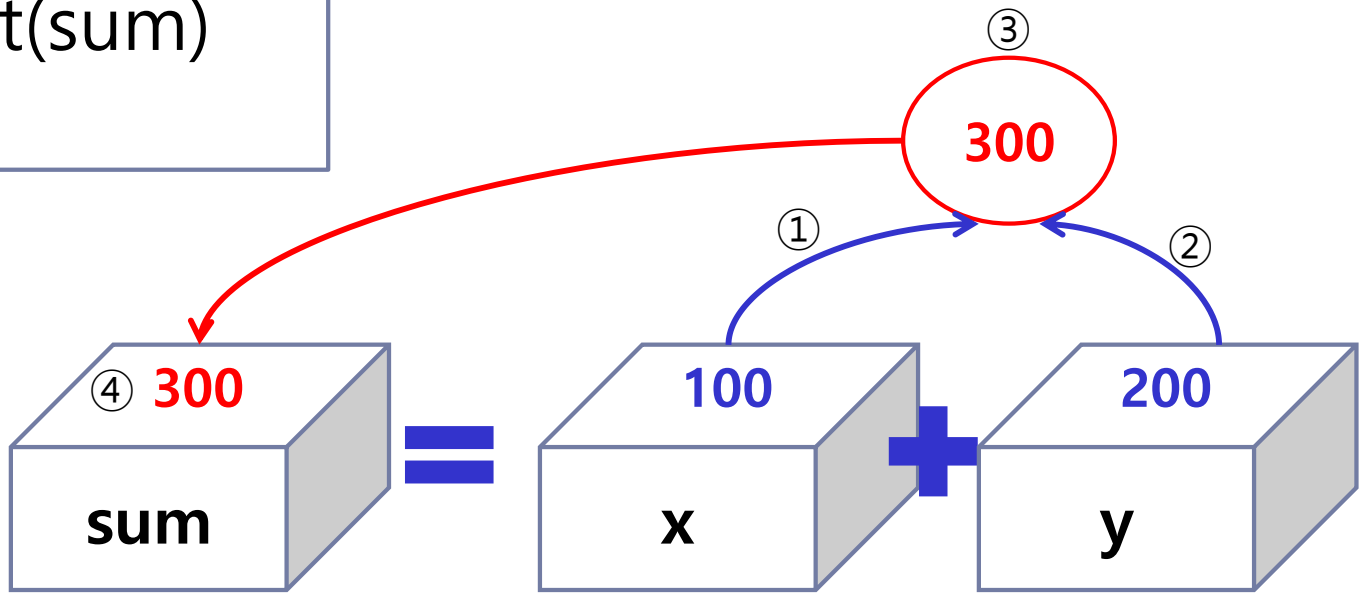
- 변수의 값은 실행 중에 변경될 수 있다.

```
>>> x = 100
>>> x = 200
>>> print(x)
200
```

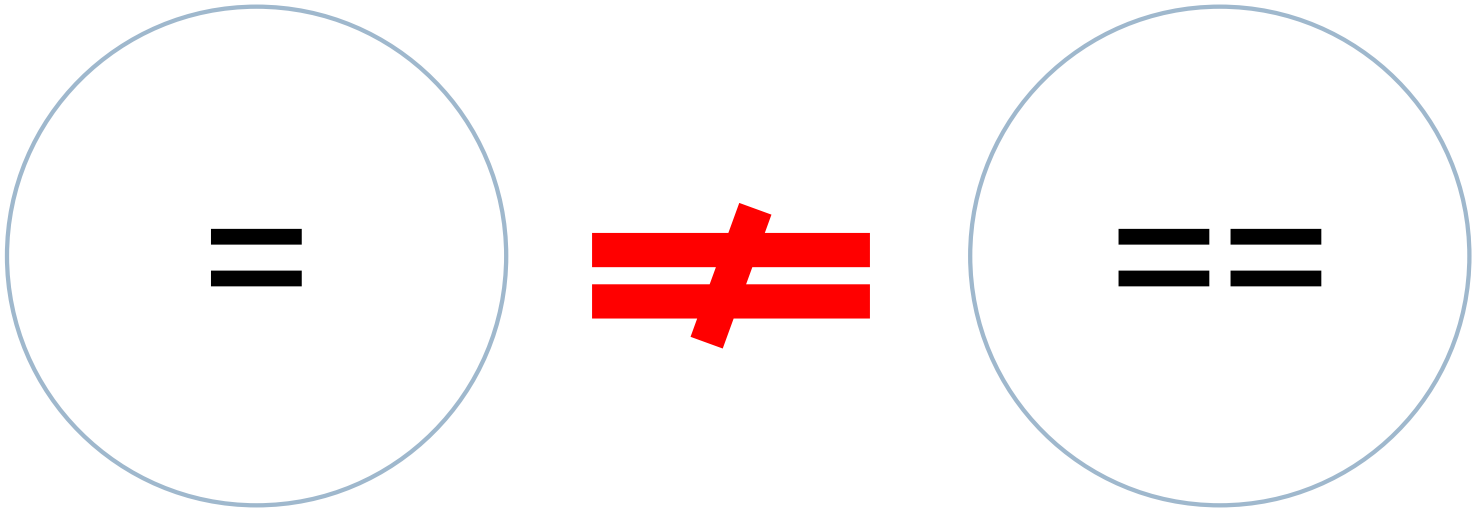


변수를 이용한 계산

```
>>> x = 100
>>> y = 200
>>> sum = x + y
>>> print(sum)
300
```



주의할 것!!!



X = 100
뜻 : 대입하라
저장하라

실행 결과 :
변수 X의 값이 100이 된다

A == 100
뜻 : 같은가?

실행 결과 :
A의 값이100이면 참(True)
그렇지 않으면 거짓(False)

변수의 이름은 어떻게 정할까?

- 첫 글자는 영문자, 밑줄문자(_)이어야 함.
- 영문자, 숫자, 밑줄문자(_) 로만 구성.
- 중간에 공백이 들어가면 안 됨.
- 대문자, 소문자는 다르게 취급.
- 예약어 사용 안됨.
- 변수의 역할을 가장 잘 설명하는 이름으로 지어주세요

  - name, _name
  - english_name, english name
  - korea1 , korea_1 , 1korea
  - index, Index, INDEX
  - dollar$, ₩won
  - if, for, while, break

데이터 타입

- 기본 데이터 타입
  - 정수형 : -100, 0, 125 …
  - 실수형 : 3.14, 175.8, …
  - 문자열형 : ‘Apple’, “Hello!”, “안녕”, ‘2022011021’
  - 논리형 : True, False

변수는 어떤 경우에 필요한가?

- 각 과목의 점수를 입력 받아서 총점을 계산하고 싶다.

- 사용자로부터 이름을 입력 받은 후,
  "당신의 이름은 OOO입니다" 라고 출력해 주고 싶다.

세 과목의 총점과 평균 구하기

앞의 문제 해결

```
이름을 입력하세요:  박정현
국어 점수를 입력하세요: 90
영어 점수를 입력하세요: 80
수학 점수를 입력하세요: 80
박정현 님의 총점은 250 이고 평균은  83.33333333333333 입니다.
```

* 밑줄(파란색) 부분은 사용자가 입력한 값입니다.

*** 키보드로부터 입력되는 모든 값은
숫자이기 전에 **문자열**이다!!!

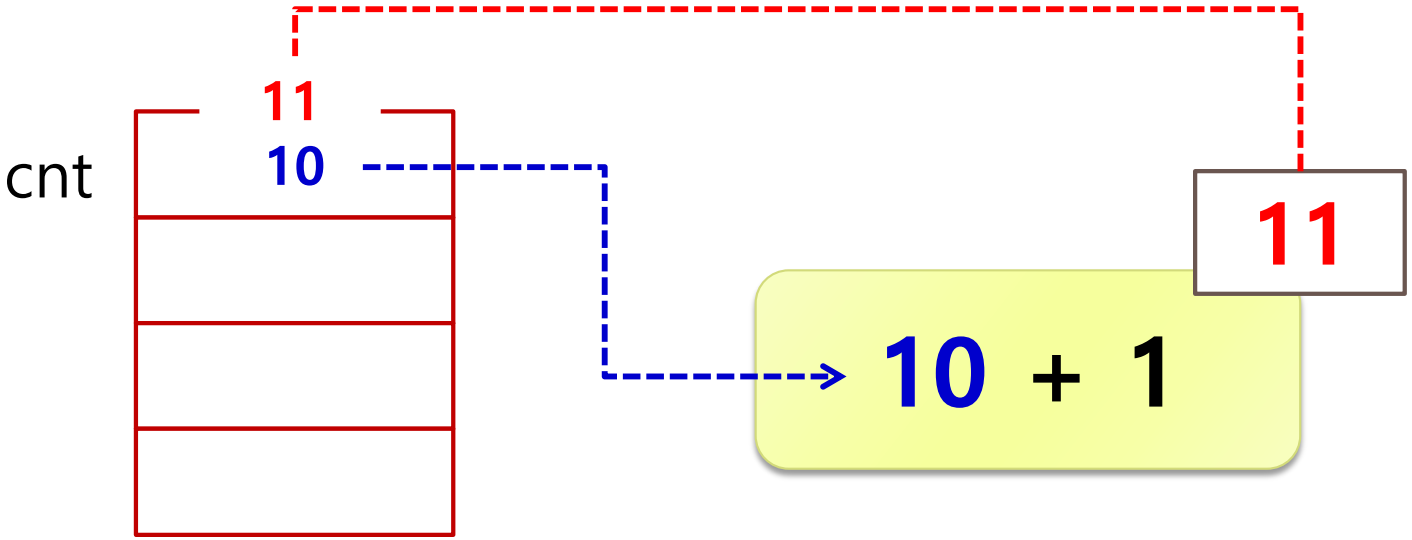
평균 = 총점 / 3

세 과목의 총점과 평균 구하기

<소스 코드 작성하기>

다음 명령문의 뜻은 매우 중요하다

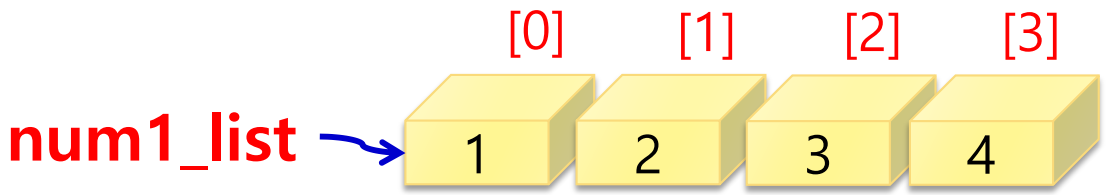
```
cnt = 10
cnt = cnt + 1        # cnt += 1
```



리스트(list)로 여러 개의 데이터를 저장!!

- 리스트를 사용하면
  여러 개의 자료들을 묶음으로 저장하고 다룰 수 있다.
- [ ] 안에 여러 개의 데이터를 ,(콤마)로 구분하여 입력.

```
>>> num1_list = [1, 2, 3, 4]
>>> print(num1_list)
[1, 2, 3, 4]
```



리스트를 조작 할 수 있다

```
>>> num1_list = [1, 2, 3, 4]
>>> num2_list = [5, 6, 7, 8]

>>> print(num1_list)
[1, 2, 3, 4]
>>> print(num2_list)
[5, 6, 7, 8]

>>> print(num1_list + num2_list)
[1, 2, 3, 4, 5, 6, 7, 8]
```

* 리스트 병합

빈 리스트 생성 후 데이터 추가하기

```
>>> list = []
>>> list.
```

append
clear
copy
count
extend
index
insert
pop
remove
reverse

tab 키를 눌러 선택

```
>>> list = [ ]
>>> list.append(1)
>>> list.append(3)
>>> list.append(5)
>>> list
[1, 3, 5]
>>> print(len(list))
3
>>> print(list[2])
5
```

문자열(String)

문자열 다루기

- 문자열은 ' ', " ", ''' ''', """ """ 을 통해 표현 가능하다!!

```
이름 = '박정현'
나이 = "27"
주소 = '''진주시'''
문자열 = """여럿이 함께"""
print(이름)
print(나이)
print(주소)
print(문자열)
```

```
===RESTART: D:/bd/string.py ===
박정현
27
진주시
여럿이 함께
```

여러 가지 기호의 필요성

- 다음 코드를 실행해 보자.

```
>>> print("우리 청년들에게 가장 큰 무기는 "열정" 입니다")
SyntaxError: invalid syntax
```

>>> print("우리 청년들에게 가장 큰 무기는 '열정' 입니다")
우리 청년들에게 가장 큰 무기는 '열정' 입니다

>>> print('여러분의 가장 큰 힘은 "도전하는 것" 입니다')
여러분의 가장 큰 힘은 "도전하는 것" 입니다

여러 가지 기호의 필요성

- 다음 코드를 실행해 보자.

https://goo.gl/E3b9Ak

day0-01.py - G:/2018 정보교육원/수업용소스코드/day0-01.py (3.7.1)

File Edit Format Run Options Window Help

```
code = "<span class="red">Heavens! what a virulent attack!</span>"
print(code)
```

SyntaxError 발생!!!

day0-01.py - G:/2018 정보교육원/수업용소스코드/day0-01.py (3.7.1)

File Edit Format Run Options Window Help

```
code = """<span class="red">Heavens! what a virulent attack!</span>"""
print(code)
```

문자열에 관련된 이미 정의된 함수들

```
>>> s = "hello world!!"
>>> s.
```

capitalize
casefold
center
count
encode
endswith
expandtabs
find
format
format_map

문자열 데이터를 다루기 위한
대표적인 기능들은
이미 함수로 구현되어져 있으므로
왼쪽과 같은 방법으로
호출해서 사용하면 된다.

문자 개수 세기와 치환 : count(), replace()

```
s = """You know I did all a father could for their education,
and they have both turned out fools."""

print(s.count('c'))          #문자 'c'의 갯수를 센다
print(s.count(' '))          #공백 문자의 갯수를 센다

s = s.replace('c', 'C')      #소문자 'c'를 대문자 'C'로 치환한다.
print(s)
```

```
========== RESTART: D:/string.py ==========
2
17
You know I did all a father Could for their eduCation,
and they have both turned out fools.
>>>
```

문자열 치환의 필요성 예 : 가격 처리

```
>>> cost ="35,000원“

>>> cost_num = cost.replace(',', '')
>>> cost_num
'35000원‘

>>> cost_num = cost_num.replace('원', '')
>>> cost_num
'35000'
```

문자열 치환의 필요성 예 : 가격 처리

```
>>> tax = cost_num * 0.1
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    tax = cost_num * 0.1
TypeError: can't multiply sequence by non-int of type 'float'

>>> tax = int(cost_num) * 0.1
>>> tax
3500.0
```

특정 문자 또는 문자열 찾기 : find( )

```
>>> s = "hello world!!“

>>> o_position  = s.find("o")    #문자 'o'를 찾았다면 위치를 반환
>>> print(o_position)
4
>>> print(s.find("k"))           #찾는 문자가 없으면 -1을 반환.
-1
>>> print(s.find("O"))
-1
```

인덱싱과 슬라이싱 : 문자열도 리스트처럼

- 인덱싱(Indexing)이란 특정 위치의 값을 가져오는 것
- 슬라이싱(Slicing)이란 인덱싱의 확장 개념으로 특정 범위의 문자열을 가져 오는 것

```
>>> s = "hello world"
>>> print(s[0])
h
>>> print(s[6])
w
>>> print(s[-1])
d
```

```
>>> print(s[20])
Traceback (most recent call last):
  File "<pyshell#20>", line 1, in <module>
    print(s[20])
IndexError: string index out of range
>>> print(s[4:7]) #4번째 문자부터 6번째까지
o w
```

문자열 나누기 : split( )

```
>>> s = "hello world "

>>> print(s.split('w'))
['hello ', 'orld']

>>> print(s.split())
['hello', 'world']

>>> s.split("d")
['hello worl', '']
```

split() 함수에게
문자열을 나눌 기준 값을
넘겨주지 않으면
space bar, enter, tab을
기준으로 나누어 준다.

대소문자 변환하기와 공백제거

```
>>> s = "hello world "
>>> print(s.upper())
HELLO WORLD

>>> s="Hello World!!"
>>> print(s.lower())
hello world!!
```

```
>>> s = "  hello world  "

>>> s.lstrip()
'hello world  '

>>> s.rstrip()
'  hello world'

>>> s.strip()
'hello world'
```

문자열과 숫자 : 이 둘은 다르다!!!

```
>>> A = "25"
>>> B = "30"
>>> print(A+B)
```

2530

```
>>> A = 25
>>> B = 30
>>> print(A+B)
```

55

숫자를 문자열로 변환 : str( )

- 다음 코드에 오류가 발생하는 이유는?

```
>>> print('나는 현재 ' + 21 + '살이다.')

Traceback (most recent call last):
  File "<pyshell#8>", line 1, in <module>
    print('나는 현재 ' + 21 + '살이다.')
TypeError: can only concatenate str (not "int") to str
```

```
>>> print('나는 현재 ' + str(21) + '살이다.')
나는 현재 21살이다.
```

문자열을 정수로 변환 : int( )

```
>>> t=input( " 숫자를 입력하세요 : ")
숫자를 입력하세요 : 30

>>> t
'30'

>>> int(t)
30
```

문자열을 실수로 변환 : float( )

```
>>> t=input("실수를 입력하세요 : ")
실수를 입력하세요 : 3.14

>>> t
'3.14'

>>> float(t)
3.14
```

```
>>> int(t)
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    int(t)
ValueError: invalid literal for int() with base 10:
'3.14'
```

함수(Function)

함수 정의하기

- 자주 사용하는 기능을 독립적인 함수로 정의함으로써
- 중복된 코드가 생기는 것을 방지하고
- 코드의 가독성과 유지보수의 효율성을 높인다

- 함수 만들기

```
>>> def  insa(x) :
        print( x + " 반가워요~")

>>> a = insa("여러분")
여러분 반가워요~
```

두 수의 합을 계산해 주는 함수 만들기

- 함수 hap(x, y)의 정의와 호출

```
>>> def hap(x, y) :    # 함수 정의
        h = x + y
        return  h       #h의 값을 호출한 쪽으로 반환한다

>>> a = hap(10, 20)    # 함수 호출
>>> print(a)
30
```

인자(parameter)의 타입으로 인한 오류

- 함수를 호출할 때 인자의 타입과 인자의 수를 함수 정의에 맞추어 호출해야 한다.

```
>>> def hap(x, y) :      #인자는 2개이며,
        h = x + y          #x와 y의 타입이 같아야 한다
        return  h

>>> a = hap(10)
```

```
Traceback (most recent call last):
  File "D:/function_hap.py", line 5, in <mo
    a = hap(10)   # 함수 호출
TypeError: hap() missing 1 req
```

a = hap(10, "여러분")와 같이 호출하면

```
Traceback (most recent call last):
  File "D:/function_hap.py", line 5, in <module>
    a = hap(10, "여러분")   # 함수 호출
  File "D:/function_hap.py", line 2, in hap
    h = x+y
TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

함수 호출할 때 매우 중요한 return!!!

- **return이 반드시 있어야 하는 것은 아니다.**
- return이 없는 함수를 호출하면 None 값을 돌려받는다.

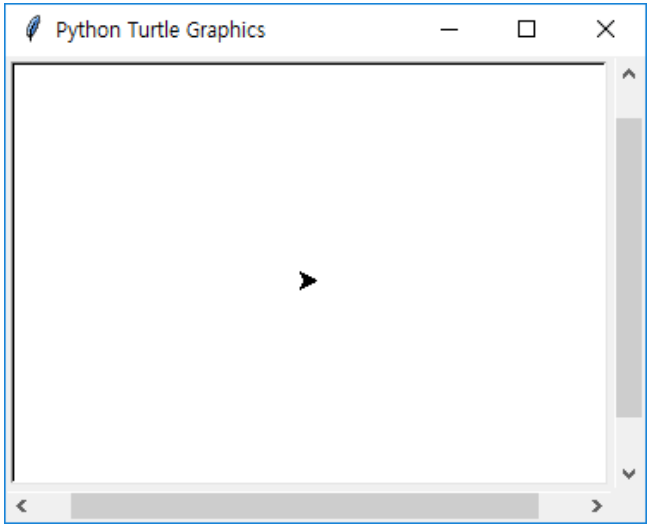
```
>>> def insa(x) :
        print(x +  "  반가워요~")

>>> a = insa("여러분")   #반환 값이 없으므로 insa("여러분")과 같이 호출!!
여러분 반가워요~
>>> a                    #대화 모드에서 반환 값을 저장한 변수 a와 엔터치면
>>> print(a)
None
```

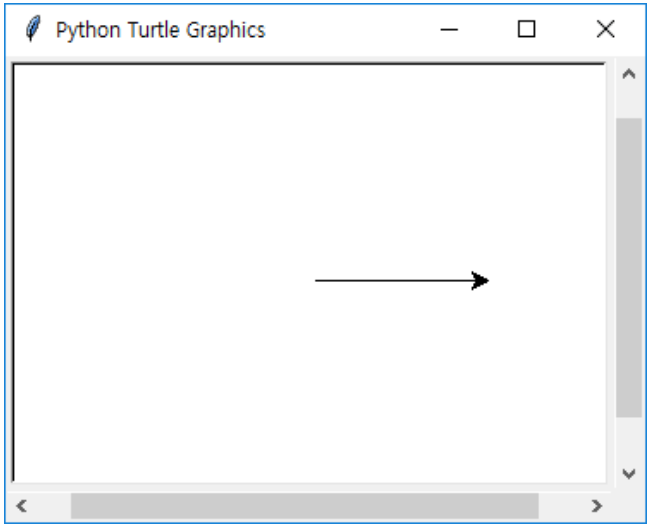
외부 모듈 불러오기

그래픽 다루기 : import turtle

```
>>> import turtle
>>> t = turtle.Turtle()
```

```
>>> t.forward(100)
```





그래픽 다루기

외부 모듈 불러오는 다양한 방법

① import turtle
② import turtle as t
③ from turtle import *

① 외부 모듈을 불러오는 가장 기본 방법 : turtle.forward(100)
② Turtle 이라는 단어 대신 t를 사용 : t.forward(100)
③ Turtle 을 생략하고 쓰겠다 : forward(100)
*외부 모듈이 많아지면 혼란스러워짐

④ 그 외, import turtle
t = turtle.Turtle( ) 과 같은 방법으로도 사용 가능하다!!

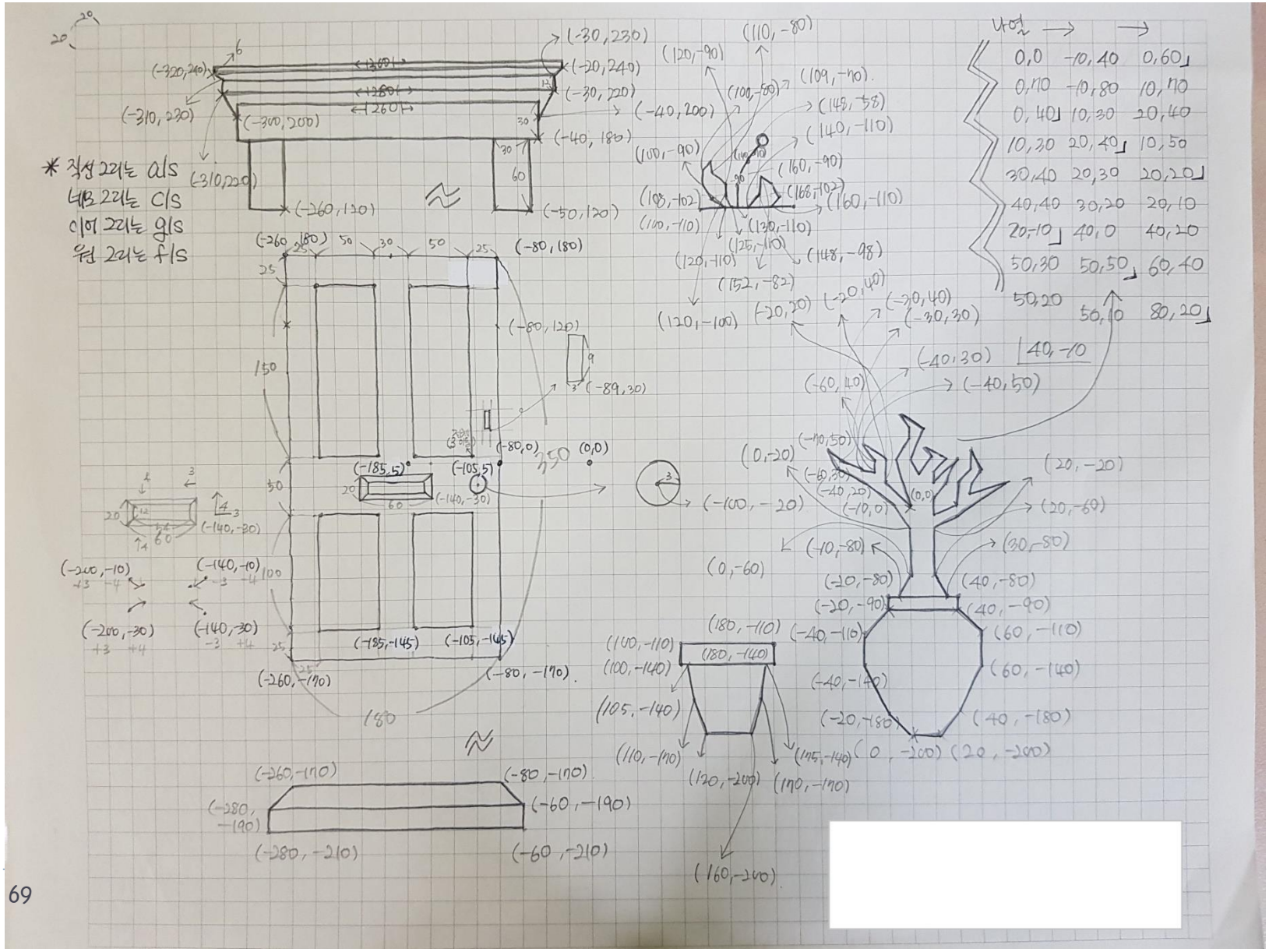
그래픽 다루기

다음과 같이 모듈에 속한 함수(기능)을 사용할 수 있다!!

```
>>> import turtle
>>> t = turtle.Turtle()
>>> t.forward(100)
>>> t.shape("turtle")
>>> t.shapesize(5)
>>> t.pencolor("red")
>>> t.left(90)
>>> t.fd(200)
>>> t.width(5)
>>> t.left(90)
>>> t.fd(200)
```

```
>>> t.lt(90)
>>> t.fd(100)
>>> t.circle(100)
>>> t.penup()
>>> t.goto(-300, -300)
>>> t.pendown()
>>> t.left(130)
>>> t.fd(400)
>>> t.right(50)
>>> t.pencolor("blue")
>>> t.fd(400)
```

그래픽 다루기



코드 분석하기

리스트 인덱스는 0부터, %는 나머지 연산자

```
import turtle as t

colors = ["red", "purple", "blue", "yellow"]
t.bgcolor("black")

t.speed(0)
t.width(3)
len = 10

while len <500 :
    t.forward(len)
    t.right(89)
    t.pencolor(colors[len%4])
    len +=5
```