

CHAPTER 1. 파이썬 맛보기

section01. 파이썬 코드를 실행하는 첫번째 방법 - 파이썬 셸 이용

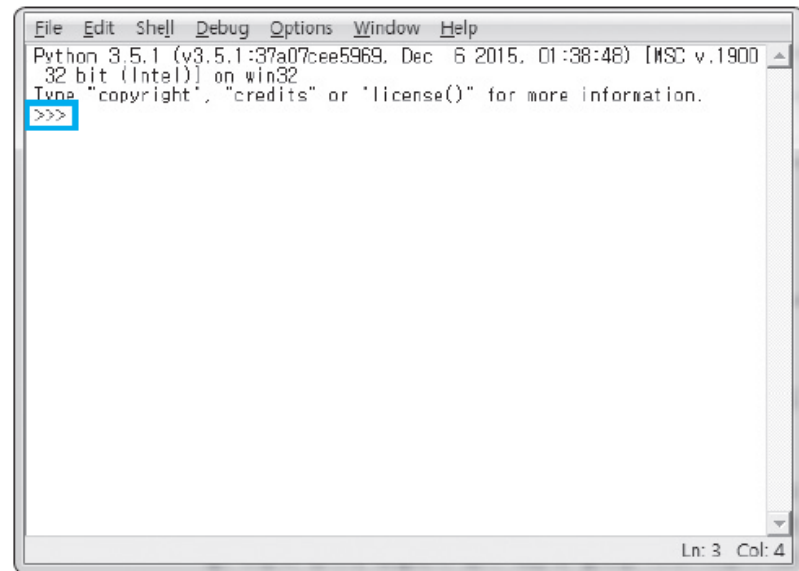
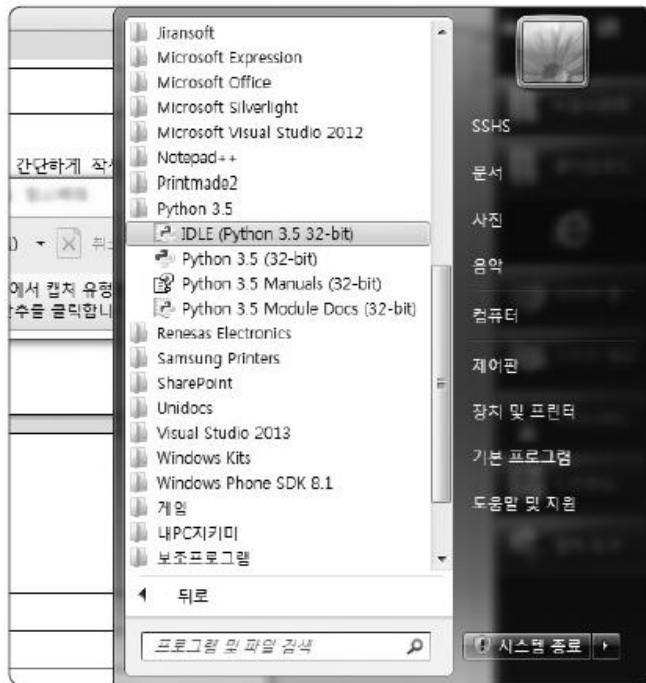
step 1. 알고가기

- 핵심내용
 - 파이썬 Shell로 실시간 코딩하기를 배운다.
- 코딩하기 전에
 - 파이썬을 코딩하는 방법에는 두 가지 방법이 있다.
 - 먼저 파이썬 셸(Shell)로 파이썬 코드를 실시간으로 실행하는 방법이다.
 - 파이썬이 무료로 제공해주는 IDLE(통합개발 환경)을 이용하면 파이썬 인터프리터를 간편하게 실행할 수 있다.

section01. 파이썬 코드를 실행하는 첫번째 방법 - 파이썬 셸 이용

step 2. 코딩

- IDLE 실행
 - 시작 메뉴에서 IDLE을 실행시킨다.
 - 파이썬이 잘 설치되어 있다면 파이썬 셸 화면이 나타난다. 셸 화면의 특징은 프롬프트(>>>) 기호이다.

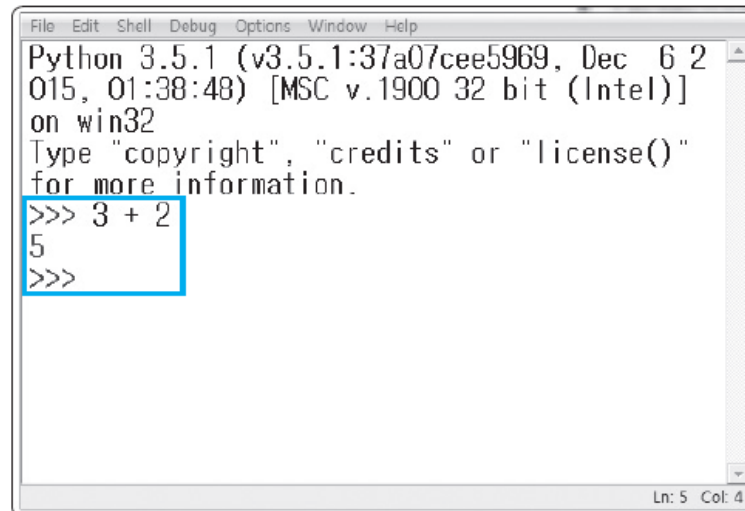


[파이썬 셸]

section01. 파이썬 코드를 실행하는 첫번째 방법 - 파이썬 셸 이용

step 3. 코드분석

- 파이썬 셸에서 코딩하기
 - 프롬프트 옆에 $3+2$ 를 입력하고 엔터를 친다.
 - 그러면 인터프리터가 이 명령을 즉시 해석해서 5라는 결과를 알려준다.
 - 셸 화면에서는 이렇게 실시간으로 코드의 결과를 확인할 수 있다.



```
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()"
for more information.
>>> 3 + 2
5
>>>
```

[파이썬 셸에서 코딩]

section01. 파이썬 코드를 실행하는 첫번째 방법 - 파이썬 셸 이용

step 4.

3줄 요약

• 3줄 요약 •

- 인터프리터는 파이썬 명령을 기계어로 번역해주는 소프트웨어이다.
- 파이썬 셸은 파이썬 인터프리터를 사용할 수 있는 화면이다.
- 프롬프트(>>>) 옆에 파이썬 명령을 입력하면 즉시 실행 결과를 알려 준다.



section02. 파이썬 코드를 실행하는 두 번째 방법 - 파이썬 셸 + 메모장

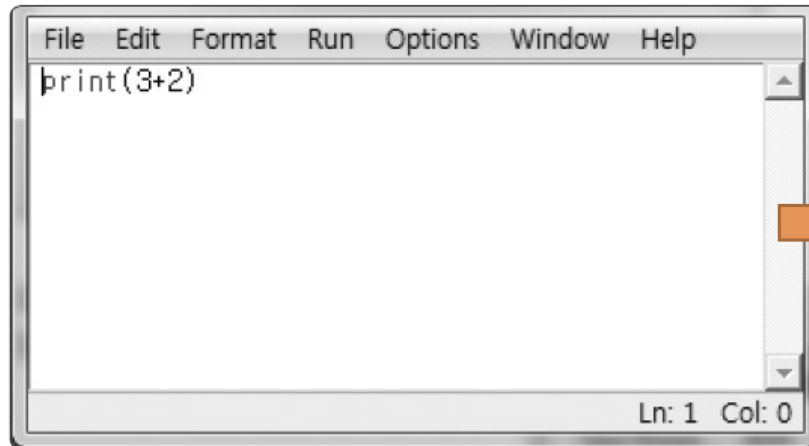
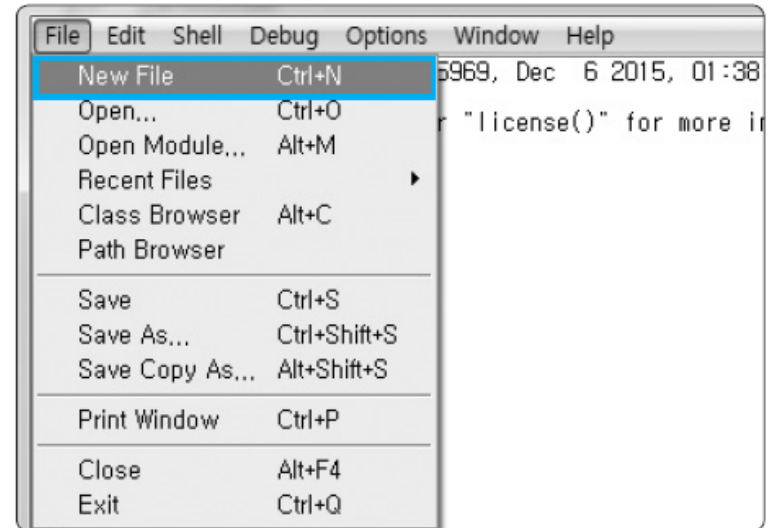
step 1. 알고가기

- 핵심내용
 - 파이썬을 *.py 파일로 저장하고 실행하는 방법을 배운다.

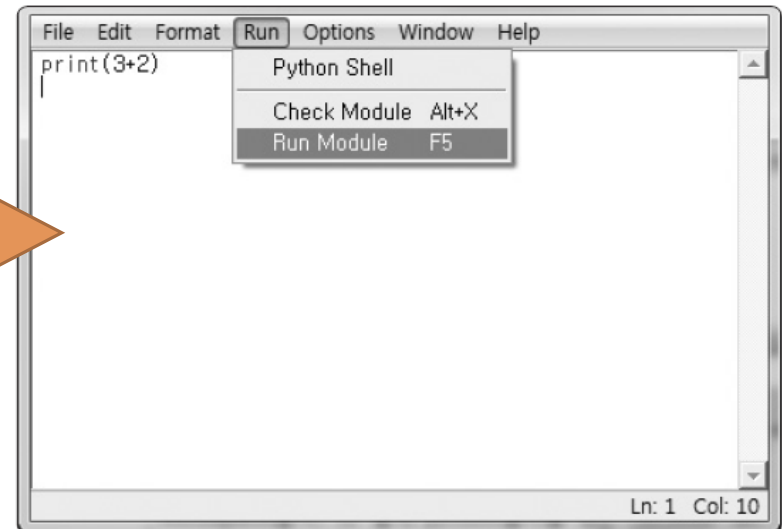
section02. 파이썬 코드를 실행하는 두 번째 방법 - 파이썬 셸 + 메모장

step 2. 코딩

- IDLE 실행
- 메모장 열기
- 메모장에 코딩하기
- 저장하기
- 실행하기



[메모장 코딩]



[소스 프로그램 실행하기]

section02. 파이썬 코드를 실행하는 두 번째 방법 - 파이썬 셸 + 메모장

step 3. 코드분석

- 파이썬 메모장을 열고 `print(3+2)`와 같이 코드를 작성한 후 메모장에 저장된 모든 코드 (현재는 단 한 줄이지만..)를 한꺼번에 실행시킬 수 있다.
- 첫 번째 방법은 실시간으로 결과를 확인할 수 있다는 장점이 있는 반면 여러 줄의 명령을 한꺼번에 처리하려면 좀 복잡하고 힘이 든다.
- 따라서 앞으로 우리는 두 번째 방법을 이용하여 코딩해 나갈 것이다.

section02. 파이썬 코드를 실행하는 두 번째 방법 - 파이썬 셸 + 메모장

step 4. 3줄 요약

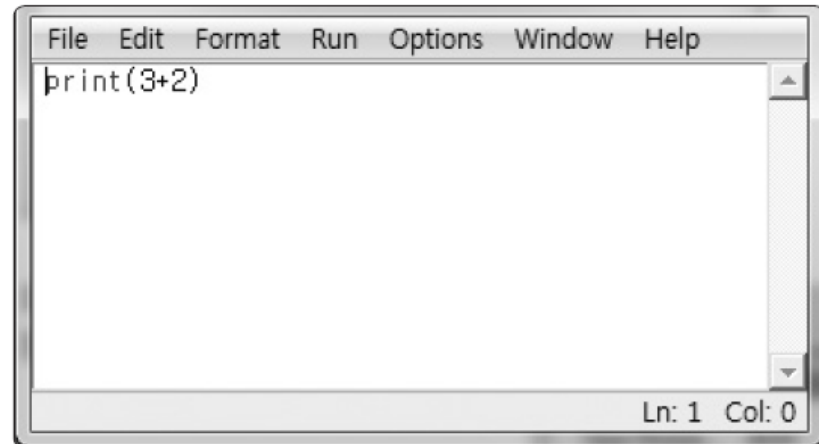
• 3줄 요약 •

- IDLE 메모장을 이용해 파이썬 코드를 작성하고 저장한다.
- 파이썬 파일의 확장자는 '.py'이니까 저장할 때는 파일명 다음에 반드시 '.py'를 적어준다.
- IDLE 메모장에서 **F5**를 누르면 파이썬 코드를 쉽게 실행시킬 수 있다.



3. 코딩하기

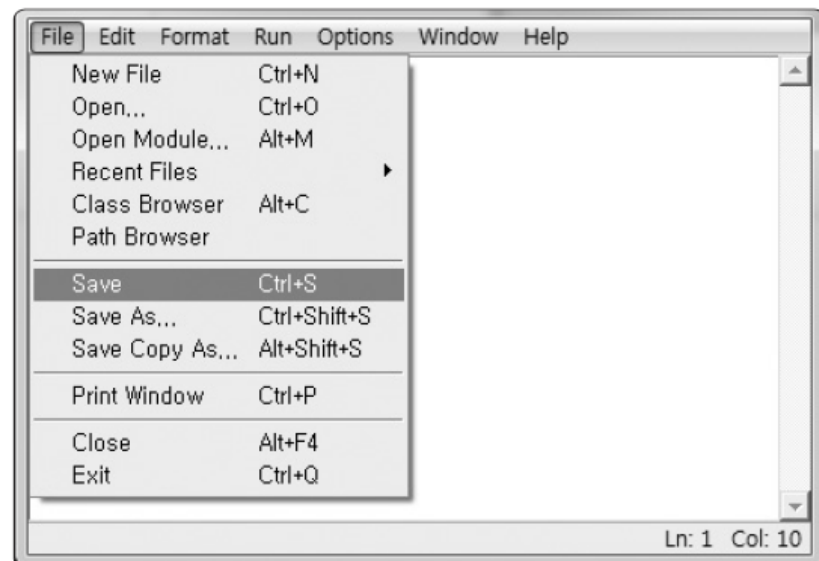
메모장이 실행되면 그림과 같이 코드를 작성해 보.



[메모장 코딩]

4. 저장하기

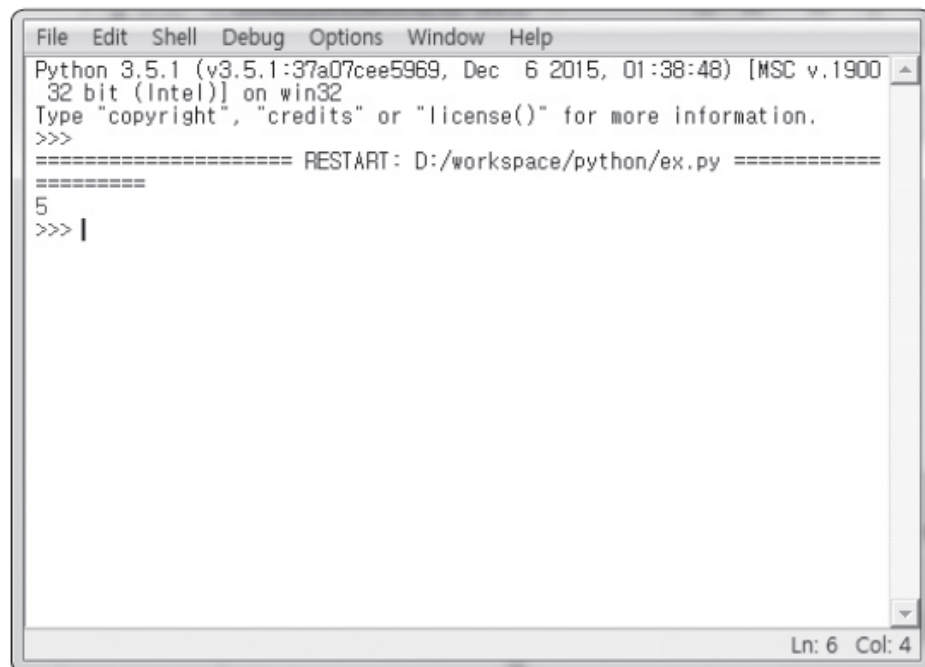
코딩이 완료되면 [File] > [Save] 버튼을 이용해서 저장해(파일을 저장하기 전에 먼저 C 드라이브에 workspace 폴더를 만들어서 C:\workspace를 만들어 두기로 해). 이 폴더는 앞으로 우리의 작업 파일을 저장할 폴더야. 파일명은 ex.py로 저장해.



[소스 프로그램 저장하기]

6. 실행 결과 확인하기

그러면 셸 화면에서 실행 결과를 확인할 수 있어. 현재 우리는 셸 화면과 메모장 이렇게 2개의 화면을 사용하고 있다는 것을 알아 뒀어. 자, 파이썬 코드를 파일로 저장해서 실행시켜 봤는데 어때? 단계는 좀더 늘었지만, IDLE 메모장을 사용하면 긴 줄의 코드를 작성하고 실행시키는데 편리하다는 장점이 있어. 그리고 앞으로 특별한 얘기가 없다면 이 방법대로 코딩하고 실행시키면 돼. 프롬프트(>>>)가 있는 코드가 보이면 그것은 첫 번째 방법처럼 파이썬 셸에서 직접 실행하면 되는 거야.



```
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1900
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:/workspace/python/ex.py =====
5
>>> |
```

[실행 결과 확인하기]

section03. 값 출력하기와 주석넣기

step 1. 알고가기

- 핵심내용
 - print() 함수는 화면에 값을 출력하는 명령이다.

section03. 값 출력하기와 주석넣기

step 2. 코딩

```
1:  # section_003.py
2:
3:  print(2020)    # 2020을 출력한다
4:  print(3 + 2)
5:  print("반갑다 파이썬!")
```

```
2020
5
반갑다 파이썬!
```

section03. 값 출력하기와 주석넣기

step 3. 코드분석

- # 은 소스 프로그램에 주석을 넣을 때 사용한다.
- 주석
 - 코드를 설명하기 위한 삽입 글을 말한다.
 - 그래서 실행에는 아무런 영향을 주지 않는다.
- 소스 프로그램에 들어간 빈 줄도 프로그램 실행에 아무런 영향을 주지 않는다.
- print() 함수
 - 화면에 값을 출력할 때 사용한다.
 - print() 함수는 출력과 관련해서 다양한 기능을 가지고 있다.

section03. 값 출력하기와 주석넣기

step 4.

3줄 요약

• 3줄 요약 •

- 화면에 값을 출력할 때는 `print()` 함수를 사용한다.
- `print()`의 괄호 안에 화면에 출력하고 싶은 값을 넣어주면 된다.
- 코드에 주석을 넣고 싶으면 `#`을 사용한다.



section04. 변수와 변수 출력

step 1. 알고가기

- 핵심내용
 - 변수는 데이터를 임시로 저장하는 공간이다.
- 코딩하기 전에
 - 변수 그 자체가 자신이 가진 값을 대표하기 때문에 값 대신 변수를 사용할 수 있다.
 - 각각의 변수명을 구분하기 위해 변수명을 사용한다.

section04. 변수와 변수 출력

step 2. 코딩

```
1: # section_004.py
2:
3: apple = 10
4: lemon, banana = 20, 50
5: fruit = apple + lemon + banana
6:
7: print(apple) .....①
8: print(banana) .....②
9: print(fruit) .....③
10:
11: apple = 30
12: print(apple) .....④
```

10 }
50 }
80 }
30 }

print() 함수가 네 번 사용되었으니
출력 결과에도 네 개의 값이
출력되어 있어.

section04. 변수와 변수 출력

step 3. 코드분석

- apple, banana처럼 변수가 가진 값의 성격을 알 수 있도록 변수명을 결정한다.
- ‘=’
 - 대입연산자(또는 할당연산자)라고 부른다.
 - 오른쪽의 ‘값’을 왼쪽의 변수에 저장하라는 의미이다.
 - 예) apple = 10
 - 대입연산자로 인해 변수 apple에 값 10을 저장한다.
- 여러 개의 변수를 한 줄에서 저장할 수 있다.

변수 = 값

section04. 변수와 변수 출력

step 3. 코드분석

- 항상 대입연산자의 오른쪽을 먼저 계산한 후 계산 결과를 왼쪽의 변수에 저장한다.
- 변수에 저장된 값은 대입연산자를 이용해서 언제든지 변할 수 있다

```
fruit = apple + lemon + banana
```



```
fruit = 10 + 20 + 50
```



```
fruit = 80
```

section04. 변수와 변수 출력

step 3. 코드분석

변수명 만들기

변수의 이름을 지을 때는 여러분이 원하는 단어를 선택하여 사용하면 되지만 다음 5가지 규칙을 지켜야 한다.

- ① 대문자와 소문자를 구분한다(apple과 Apple은 다른 변수이다).
- ② 영문 대문자, 소문자, 숫자 그리고 밑줄(_)을 사용하여 만들 수 있다.
- ③ 변수명 중간에 공백이 들어가면 안 된다.
- ④ 숫자로 시작하는 변수명도 안 된다.
- ⑤ 예약어는 변수로 사용할 수 없다.

좋은 변수명은 이름만 보고 변수가 저장한 값의 의미를 짐작할 수 있도록 하는 것이 좋다. 변수 값의 의미를 알 수 없는 나쁜 변수명은 가능한 안 쓰는 것이 좋다.

좋은 변수명	나쁜 변수명
sum	abc
maxAge	aaa
next_value	a
score	c1

section04. 변수와 변수 출력

step 3. 코드분석

- 예약어
 - 파이썬에서 이미 다른 목적을 위해 사용하는 단어로써 변수명으로 사용할 수 없다.
 - 33개의 예약어 중 True, False, None를 제외하면 모두 소문자로만 구성되어 있다.

파이썬의 예약어

and, as, assert, break, class, continue,
def, del, elif, else, except, False
finally, for, from, global, if,
import, in, is, lambda, None, nonlocal,
not, or, pass, raise, return, True
try, while, with, yield

section04. 변수와 변수 출력

step 4.

3줄 요약

• 3줄 요약 •

- 변수는 값을 임시로 저장하는 공간을 말한다.
- 변수명 만들기 규칙 5개를 기억하자.
- 변수가 '=' 왼쪽에 있으면 값을 저장한다는 의미이고, '=' 오른쪽에 있으면 변수를 사용한다는 의미이다.



section05. 간단한 연산 맛보기

step 1. 알고가기

- 핵심내용
 - 산술 연산과 산술 연산자

section05. 간단한 연산 맞보기

step 2. 코딩

```
1: # section_005.py
2:
3: add = 3 + 25
4: sub = 45 - 13
5: mul = 9 * 9
6: div = 3 / 2
7: div_int = 3 // 2
8: exp = 10 ** 3
9: mod = 101 % 10
10:
11: print(add)
12: print(sub)
13: print(mul)
14: print(div)
15: print(div_int)
16: print(exp)
17: print(mod)
```

각 줄에 있는
띄어쓰기는 가독성을 위한
거야. 띄어쓰기를 하든 안하든
프로그램 실행에는
관계가 없어.
즉, add=3+25 이렇게
작성해도 문제없어.

```
28
32
81
1.5
1
1000
1
```

section05. 간단한 연산 맞보기

step 3. 코드분석

- 산술연산
 - 더하기, 빼기, 곱하기, 나누기 등의 연산을 의미한다.
 - 산술연산자는 산술연산을 위한 명령 기호(+, -, *, /)를 말한다.
- 나누기 연산자
 - 실수 나눗셈(/)
 - 예) $3/2$ 의 계산결과는 1.5이다.
 - 정수 나눗셈(//)
 - 예) $3//2$ 의 계산결과는 1이다.
- 지수 연산자(**)
 - 예) $10^{**}3$ 의 계산결과는 1000이다.
- 나머지 연산자(%)
 - 예) $101\%10$ 의 계산결과는 1이다.

section05. 간단한 연산 맞보기

step 4.

3줄 요약

• 3줄 요약 •

- 곱셈 기호(*)와 나눗셈 기호(/ 와 //)는 수학 기호와 다르니까 잘 기억해 두자.
- 지수승을 구하는 연산자는 별 두 개를 연결한 **이다.
- %는 나머지 연산자로서 연산에서 자주 사용된다.



section06. 문자열 맛보기

step 1. 알고가기

- 핵심내용
 - 문자열은 문자를 표현하기 위한 것으로 따옴표로 묶어 처리한다.
- 코딩하기 전에
 - 프로그래밍에서 문자열 처리는 매우 중요하다. 학교 이름을 입력하거나 영화를 검색하려면 문자를 사용해야 하기 때문이다.
 - 우리가 평소에 문자를 자주 사용하는 만큼 프로그램에서 문자열을 다루는 것은 매우 중요한 작업이다.

section06. 문자열 맞보기

step 2. 코딩

```
1: # section_006.py
2:
3: print('안녕하세요.')
4: print("I love the Earth!")
5: print("내가 좋아하는 숫자는 1, 4, 7입니다.")
6:
7: city = 'Seoul'
8: dosi = "미래 도시"
9: print(dosi)
10: print(city)
```

```
안녕하세요.
I love the Earth!
내가 좋아하는 숫자는 1, 4, 7입니다.
미래 도시
Seoul
```

section06. 문자열 맛보기

step 3.

코드분석

- 문자열
 - 글자 또는 숫자, 특수 문자, 공백 등이 나열된 것을 말한다.
 - 문자열을 표현하기 위해서는 항상 작은따옴표(') 또는 큰따옴표(")로 묶어준다.
 - 예) '안녕하세요.'
 - "I love the Earth! "
 - "내가 좋아하는 숫자는 1, 4, 7입니다."
 - 작은따옴표(')로 시작한 문자열은 작은따옴표(')로, 큰따옴표(")로 시작한 문자열은 큰따옴표(")로 끝나야 한다
- 문자열을 변수에 저장할 수도 있다.

section06. 문자열 맛보기

step 4.

3줄 요약

• 3줄 요약 •

- 프로그래밍에서 문자열을 다루는 것은 매우 중요한 일이다.
- 작은 따옴표 또는 큰 따옴표로 묶여 있으면 문자열이다.
- 문자열을 저장한 변수는 문자열 변수이다.



section07. 파이썬 문장

step 1. 알고가기

- 핵심내용
 - 파이썬은 문장 단위로 처리한다.
- 코딩하기 전에
 - 문장이란 파이썬 코드로 한 줄을 구성할 수 있는 모든 것을 말한다.
 - 그러나 때로는 한 줄에 여러 문장을 표현할 수 있고, 한 문장을 여러 줄에 걸쳐 작성할 수도 있다.

section07. 파이썬 문장

step 2. 코딩

```
1: # section_007.py
2:
3: number = 1
4: print(number)
5:
6: num1 = 10; num2 = 20; num3 = 30
7: print(num1 + num2 + num3)
8:
9: sum = 1 + 2 + 3 + \
10:      4 + 5 + 6 + \
11:      7 + 8 + 9
12: print(sum)
13:
14: tup = (1 + 2 + 3 +
15:        4 + 5 + 6 +
16:        7 + 8 + 9)
17: print(tup)
18:
19: color = ['red',
20:          'blue',
21:          'green']
22: print(color)
```

```
1
60
45
45
['red', 'blue', 'green']
```

section07. 파이썬 문장

step 3.

코드분석

- `number=1`은 한 줄 구성된 문장이다.
- 특히 이렇게 변수에 값을 저장하는 문장을 할당문(assignment statement)이라고 한다.
- `print()` 함수를 호출하는 4번 줄도 하나의 문장이다.
- 여러 문장을 한 줄에 나열할 때는 세미콜론(;)으로 구분해주면 된다.
- 연결문자인 역슬래시(\)를 줄의 끝에 입력하면 하나의 문장을 여러 줄에 걸쳐 작성할 수 있다.

section07. 파이썬 문장

step 4.

3줄 요약

• 3줄 요약 •

- 문장은 보통 파이썬 코드로 한 줄을 구성할 수 있는 모든 것을 말한다.
- 여러 문장을 한 줄에 나타내고 싶으면 세미콜론(;)을 사용한다.
- 한 문장을 여러 줄로 나누고 싶을 땐 연결 연산자인 역슬래시(\)를 사용한다.



section08. 함수 맛보기

step 1. 알고가기

- 핵심내용
 - 함수란 특정한 일을 처리해주는 역할을 한다.
- 코딩하기 전에
 - 함수(function)는 특정한 일을 처리해주는 코드들의 집합이다.
 - 특정한 일이란 최댓값을 구하는 일, 합계를 구하는 일 등 다양하다. 이미 배운 `print()` 함수는 괄호 안에 주어진 값을 셸 화면에 출력하는 일을 처리해주는 함수이다.

section08. 함수 맛보기

step 2. 코딩

```
1: # section_008.py
2:
3: print("안녕! 나도 함수야")
4:
5: length = len("이 문자열의 길이는 얼마일까?")
6: print(length)
7:
8: max_number = max(11, 2, 63, 47, 50)
9: print(max_number)
```

안녕! 나도 함수야

16

63

section08. 함수 맛보기

step 3. 코드분석

- 함수 호출
 - 함수를 사용하고 싶을 때 함수의 이름을 불러 주는 것을 말한다.
 - 함수 호출 시 함수명과 ()를 붙여서 작성한다.
 - 예) print(), len(), max() 등등
- 함수의 입력값
 - 함수의 쓰임에 따라 입력값이 필요한 함수가 있다.
 - 함수의 입력값은 () 안에 작성한다.
 - 예) print("안녕! 나도 함수야")
- 함수의 반환값
 - 어떤 함수는 함수 처리 결과를 반환해주기도 한다. 이 결과값을 사용하기 위해 대입연산자를 활용한다.

section08. 함수 맛보기

step 4.

3줄 요약

• 3줄 요약 •

- 함수는 특정한 일을 처리하는 코드들의 집합이다.
- 함수는 호출됨으로써 그 기능을 실행시킬 수 있다.
- 함수를 공부할 때는 함수의 기능과 함께 입력값과 반환값이 어떤 형태인지를 기억해야 한다.



section09. 사용자로부터 입력받기

step 1. 알고가기

- 핵심내용
 - 셸 화면을 통해 사용자로부터 값을 입력받을 때는 `input()` 함수를 사용한다.
- 코딩하기 전에
 - 프로그램은 기본적으로 입력 - 처리 - 출력의 세 단계를 거치면서 실행된다.
 - 입력
 - 사용자로부터 값을 입력받는 것이다.
 - 예) `input()` 함수는 화면을 통한 입력을 위한 함수이다.
 - 처리
 - 산술연산이나 함수처럼 값을 조작하는 것
 - 출력
 - 결과값을 사람들에게 보여주는 것이다.
 - 예) `print()` 함수는 화면 출력을 위한 함수이다.

section09. 사용자로부터 입력받기

step 2. 코딩

```
1: # section_009.py
2:
3: print('회원가입 기본정보')
4: name = input()
5: age = input('나이를 입력하세요:')
6:
7: print('당신의 이름과 나이는 다음과 같습니다.')
8: print(name)
9: print(age)
```

회원가입 기본정보

하하

나이를 입력하세요: 38

당신의 이름과 나이는 다음과 같습니다.

하하

38

사용자가 입력하는
부분이야.

section09. 사용자로부터 입력받기

step 3. 코드분석

- input() 함수
 - input() 함수 코드가 실행되면 사용자 입력을 기다리기 때문에 쉘 화면이 마치 멈춘 것처럼 보인다.
 - input() 함수로 입력받은 값을 프로그램 내부에서 처리하기 위해 보통 대입연산자와 함께 사용한다.(4, 5번 줄)
 - input() 함수의 () 안에 어떤 값을 입력하면 입력값에 대한 안내 메시지로 활용할 수 있다.
 - 예) input(' 나이를 입력하세요 ')

section09. 사용자로부터 입력받기

step 4.

3줄 요약

• 3줄 요약 •

- 프로그램은 기본적으로 입력-처리-출력의 세 단계를 계속 반복하면서 실행된다.
- 셸 화면을 통해 사용자로부터 값을 입력받을 때는 `input()` 함수를 사용한다.
- `input()` 함수의 괄호에 사용자에게 전달할 지시 내용을 삽입할 수 있다.



section10. 모듈 맛보기

step 1. 알고가기

- 핵심내용
 - 모듈을 임포트하고 모듈이 가진 기능을 사용해보자.
- 코딩하기 전에
 - 모듈은 서로 관련있는 변수, 함수 등을 모아놓은 파일을 말한다.
 - 예) math 모듈은 수학 계산 관련 모듈로서 원주율, 제곱근 함수 등을 가지고 있다.
 - 파이썬이 기본적으로 제공해주는 모듈을 내장모듈이라고 한다.
 - 내장모듈을 활용하면 내가 직접 작성해야 하는 코드가 줄어드니까 많은 시간을 절약할 수 있다.

section10. 모듈 맛보기

step 2. 코딩

```
1:  # section_010.py
2:
3:  import math
4:
5:  print(math.pi)
6:  print(math.sqrt(16))
7:
8:  r = 10
9:  cir = 2 * math.pi * r
10:
11: print('반지름이', r, '인 원의 둘레는', cir, '이다.')
```

3.141592653589793

4.0

반지름이 10인 원의 둘레는 62.83185307179586 이다.

section10. 모듈 맛보기

step 3. 코드분석

- импорт(import)
 - import는 모듈을 내 프로그램에 삽입하는 과정이다.
 - 모듈을 내 프로그램에서 사용하기 위해서는 모듈을 import해야 한다.
 - 예) `import math`
- 모듈을 import 한 다음에는 모듈이 가지고 있는 각종 변수와 함수를 사용할 수 있다.
- 모듈이 가진 변수나 함수를 사용할 때는 기본적으로 `모듈명.변수` 또는 `모듈명.함수()` 처럼 사용한다.
- 예) `math.pi`, `math.sqrt()`

section10. 모듈 맛보기

step 4.

3줄 요약

• 3줄 요약 •

- 모듈은 서로 관련 있는 코드들을 모아 놓은 파일이다.
- 모듈을 내 프로그램에서 사용하려면 import문을 이용해서 모듈을 임포트한다.
- 모듈을 임포트한 후에는 모듈이 가진 기능을 편리하게 사용할 수 있다.



section11. 오류와 친해지기

step 1. 알고가기

- 핵심내용
 - 오류를 무서워하지 말자.
- 코딩하기 전에
 - 오류란 컴퓨터가 이해할 수 없는 문법적, 논리적 오류를 말한다.
 - 프로그래머가 명령한 내용을 이해할 수 없으면 오류를 발생시키는데, 이 오류를 공부하면 프로그래밍이 훨씬 쉽고 재미있어진다.

section11. 오류와 친해지기

step 2. 코딩

```
1: # section_011.py
2:
3: frint(2020)          # 함수 이름을 잘못 입력한 오류
4:
5: # print("안녕하세요") # 문자열의 시작과 끝을 서로 다른 따옴표로 작성한 오류
```

section11. 오류와 친해지기

step 3. 코드분석

- 오류 표시의 형태

Traceback (most recent call last):

File "D:\workspace\section_011.py", line 3, in <module>

frint(2020) # 함수이름을 잘못 입력한 오류

NameError: name 'frint' is not defined

- 두 번째 줄: 오류가 발생한 파일과 줄 번호 표시
 - 예) "D:\workspace\ex-1-8.py", line 3
- 세 번째 줄: 오류가 발생한 명령을 표시
 - 예) frint(2020)
- 네 번째 줄: 어떤 종류의 오류인지 표시
 - 예) **NameError:** name 'frint' is not defined
- 디버깅(debugging)
 - 오류를 찾아서 수정하는 것을 디버깅이라고 한다.
 - 몇 가지 오류에 익숙해지면 쉽게 수정할 수 있다.

section11. 오류와 친해지기

step 4.

3줄 요약

• 3줄 요약 •

- 컴퓨터가 이해할 수 없는 문법적, 논리적 문제를 오류라고 한다.
- 오류는 컴퓨터와의 대화 과정이므로 무서워하지 말자.
- 오류를 수정하는 과정을 디버깅이라고 한다.



section12. 복합 대입 연산자

step 1. 알고가기

- 핵심내용
 - 복합 대입 연산자는 두 개의 연산자를 결합해 놓은 것이다.

section12. 복합 대입 연산자

step 2. 코딩

```
1:  # section_012.py
2:
3:  x = 5
4:
5:  x = x + 1
6:  print(x)
7:
8:  x += 1      # x = x + 1
9:  print(x)
10:
11: y = 10
12: y *= x      # y = y * x
13: print(y)
```

```
6
7
70
```

section12. 복합 대입 연산자

step 3. 코드분석

- $X = X + 1$
 - 이 문장에는 두 개의 연산자(=, +)가 있다.
 - 대입연산자(=)가 보이면 무조건 오른쪽을 먼저 계산한다.

$x = x + 1$



$x = 5 + 1$



$x = 6$

- $X += 1$
 - 이 문장은 $x = x + 1$ 을 줄여서 표현한 것이다.
 - 이때 사용한 +=를 복합대입연산자라고 한다.
 - 복합대입연산자를 사용하는 이유는 빠른 코딩을 위해서이다. 그러나 너무 많이 사용하면 오히려 코드가 복잡해보일 수 있다.

section12. 복합 대입 연산자

step 3. 코드분석

- 복합 대입 연산자의 종류

연산자	같은 의미	연산자	같은 의미
$x += y$	$x = x + y$	$x //= y$	$x = x // y$
$x -= y$	$x = x - y$	$x \% = y$	$x = x \% y$
$x *= y$	$x = x * y$	$x ** = y$	$x = x ** y$
$x /= y$	$x = x / y$		

section12. 복합 대입 연산자

step 4.

3줄 요약

• 3줄 요약 •

- 복합 대입 연산자는 대입 연산자와 계산에 사용되는 연산자를 합쳐놓은 것이다.
- 복합 대입 연산자를 이용하면 코드를 간결하게 표현할 수 있다.
- 복합 대입 연산자를 너무 많이 사용하면 오히려 코드가 복잡해 보일 수 있다.



section13. 연산자 우선 순위

step 1. 알고가기

- 핵심내용
- 연산자들 사이에는 우선순위가 있다.
- 코딩하기 전에
 - 한 줄에 여러 연산자들이 나열되면 어느 연산자를 먼저 계산해야 할지 결정해야 한다.
 - 그런데 이 순서가 이미 정해져 있는데 이것을 우선 순위라고 한다.

section13. 연산자 우선 순위

step 2. 코딩

```
1:  # section_013.py
2:
3:  x = 5
4:
5:  x = x + 1 * 10      # x = x + (1 * 10)
6:  print(x)
7:
8:  y = 42 - 1 + 1 - 10
9:  print(y)
10:
11: z = (x + y) * 6
12: print(z)
```

```
15
32
282
```

section13. 연산자 우선 순위

step 3. 코드분석

- $X = X + 1 * 10$
 - 대입연산자는 무조건 오른쪽 먼저 계산하다.
 - 오른쪽에 두 개의 연산자(+, *)가 있다.
 - 우선순위가 높은 곱셈(*)을 먼저 계산한다.
- 연산자의 우선순위는 다음과 같이 정해져 있다.

[연산자 우선 순위]

우선 순위	연산자	참고
1	()	괄호(괄호를 가장 먼저 계산한다)
2	**	지수승
3	*, /, //, %	곱셈과 나눗셈, 나머지 연산
4	+, -	덧셈과 뺄셈
5	=, 복합 대입 연산자	대입 연산자와 복합 대입 연산자

section13. 연산자 우선 순위

step 3. 코드분석

- 괄호는 가장 높은 우선순위를 가진다.
- 같은 순위의 연산자들이 나열되면 왼쪽부터 오른쪽 방향(->)순으로 계산한다.
 - 예) $y = 42 - 1 + 1 - 10$ 은 $y = (((42 - 1) + 1) - 10)$ 와 같다.
- 대입연산자의 우선순위는 앞으로 배울 어떤 연산자보다 우선순위가 낮다. 따라서 대입연산자를 보면 무조건 오른쪽부터 계산한다.

section13. 연산자 우선 순위

step 4. 3줄 요약

• 3줄 요약 •

- 연산자들 사이에는 우선 순위가 있다.
- 동등한 우선 순위를 가지는 연산자들이 나열되어 있을 때에는 오른쪽 방향으로 순서대로 계산한다.
- 괄호는 가장 높은 우선 순위를, 대입 연산자/복합 대입 연산자는 가장 낮은 우선 순위를 갖는다.

