2장 변수와 수식

학습 목표

- 변수와 상수를 정의하고 사용할 수 있다.
- 주석의 개념을 이해한다.
- 산술 연산자와 할당 연산자에 대하여 이해한다.
- 연산자의 우선순위 개념을 이해한다.
- 사용자로부터 입력을 받고 출력을 하는 프로그램을 작성할 수 있다.
- 문자열의 기초 연산을 이해한다.

이번 장에서 만들 프로그램

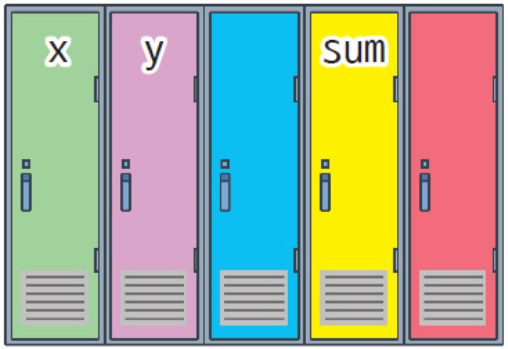
```
첫 번째 정수를 입력하시오: 10
두 번째 정수를 입력하시오: 3
10 의 3 승은 1000 입니다.
```

```
몸무게를 kg 단위로 입력하시오: 85.0
키를 미터 단위로 입력하시오: 1.83
당신의 BMI= 25.381468541909282
```

```
물건값을 입력하시오: 750
1000원 지폐개수: 1
500원 동전개수: 0
100원 동전개수: 0
500원= 0 100원= 2 10원= 5 1원= 0
```

변수

- 변수(variable)는 컴퓨터의 메모리 공간에 이름을 붙이는 것으로 우리는 여기에 값을 저장할 수 있다.
- 예를 들어보자. 헬스 클럽에 등록하면 개인 락커를 받게 된다. 우리는 거기에 우리 이름을 적어 놓을 수 있고 여러가지 물건들을 저장할 수 있다.
- 프로그램 안에서 락커와 같은 역할을 하는 것이 변수이다.



변수 정의하기

- 파이썬에서는 변수에 값을 저장하면 변수가 자동으로 생성된다.

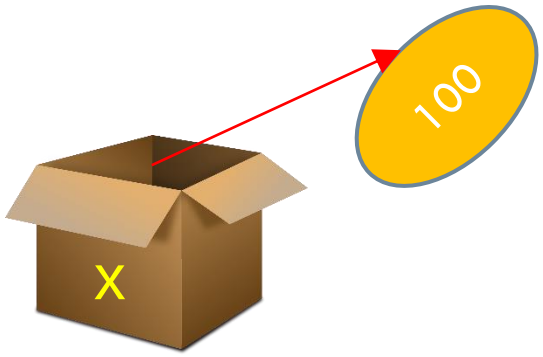
Syntax: 변수정의

형식 변수이름 = 값

예 x = 100

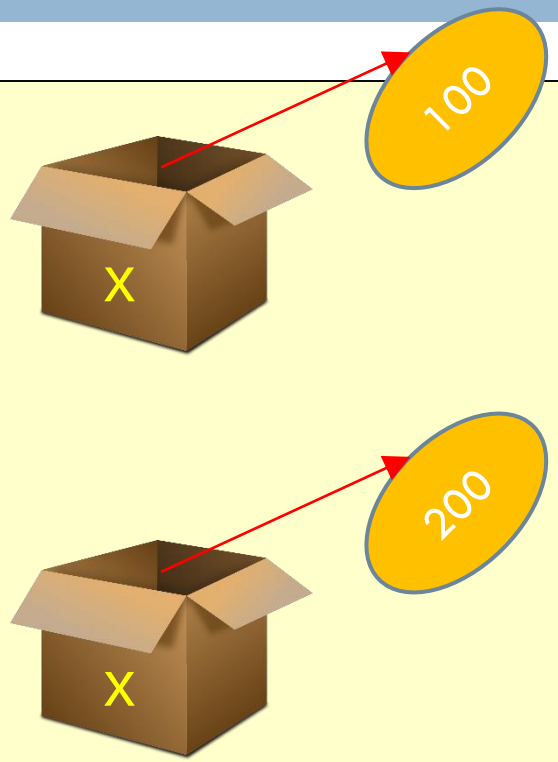
변수이름 값

변수는 상자와 같고, 상자 안에 값이 저장된다.



변수에 저장된 값을 변경할 수 있다.

```
>>> x = 100

>>> print(x)
100

>>> x
100

>>> x = 200
>>> x
200
```

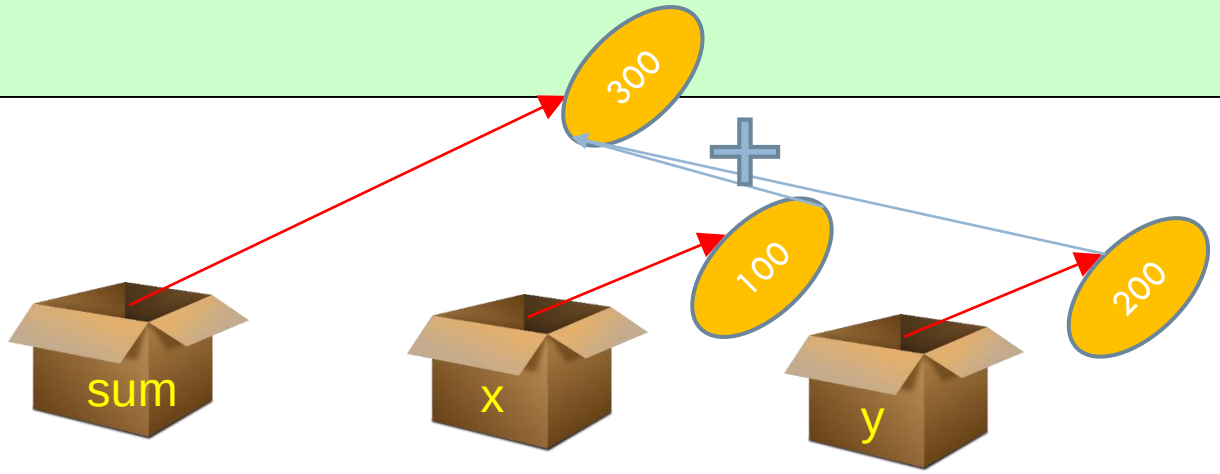


예제: 두 수의 합 계산하기

p61_sum.py

```
x = 100                 # 변수 x를 생성하고 100을 저장한다.
y = 200                 # 변수 y를 생성하고 200을 저장한다.
sum = x + y             # 변수 sum을 생성하고 x+y를 저장한다.
print("합은", sum)
```

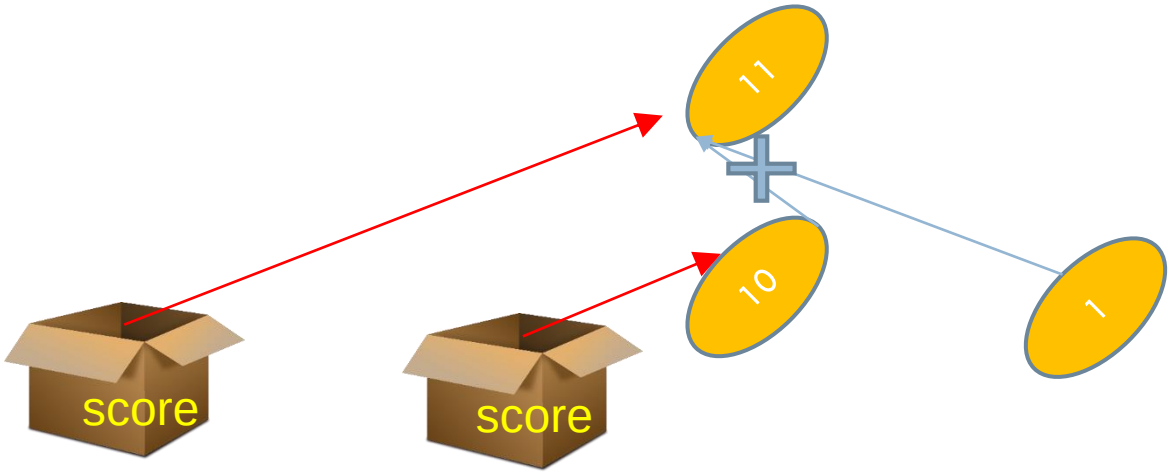
```
합은 300
```



이런 것도 가능하다!

```
score = 10
score = score + 1
```

- 프로그래밍에서 '=' 기호는 양변이 같다는 등호가 아니다. 오른쪽 값을 왼쪽의 변수에 저장하라는 의미가 된다.



예제: 원의 면적 계산하기

```
반지름 20인 원의 면적= 1256.0
```

알고리즘

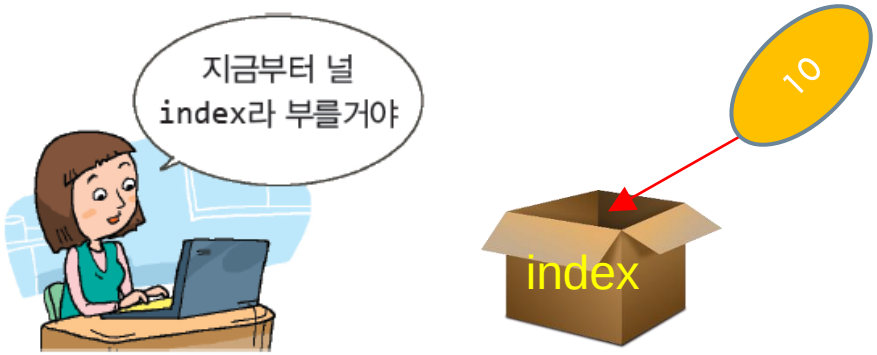
```
STEP #1. 사용자로부터 원의 반지름을 입력받는다.
STEP #2. 공식을 적용하여 면적을 계산한다.
                                    area = radius * radius * p
STEP #3. 면적을 화면에 출력한다.
```

코드

p64_area.py

```
# 변수 radius에 값을 저장한다.
radius = 10

# 공식을 적용하여 면적을 계산한다
area = 3.14 * radius * radius

# 면적을 화면에 출력한다.
print("반지름", radius, "인 원의 면적=", area)
```

변수의 이름



- 첫 글자는 영문자 또는 밑줄(_) 문자로 시작. 나머지 문자는 문자, 숫사 또는 밑줄이어야
- #와 같은 기호는 사용할 수 없다. 이름 안에 공백도 허용되지 않는다.
- 소문자와 대문자는 서로 다르게 취급된다.
- If와 같은 예약어를 이름으로 사용할 수 없다. 예약어란 파이썬이 사용하는 특수한 단어이다.(p65 참조)

변수의 이름

- 유효한 식별자와 유효하지 않은 식별자의 예

변수 이름	설명
size	가능하다.
cloud9	가능하다. 변수는 영문자, 숫자, _로 이루어진다.
max_size	가능하다. 변수의 중간에 _가 있어도 된다.
_count	가능하다. _가 앞에 붙으면 클래스 내부에서만 사용하는 변수라는 의미도 있다.
6pack	올바르지 않다! 숫자가 앞에 오면 안된다.
mid score	올바르지 않다! 중간에 공백이 있으면 안된다.
class	올바르지 않다! 예약어를 변수의 이름으로 사용할 수 없다.
money#	올바르지 않다! 기호를 변수의 이름으로 사용하면 안 된다.

중간점검

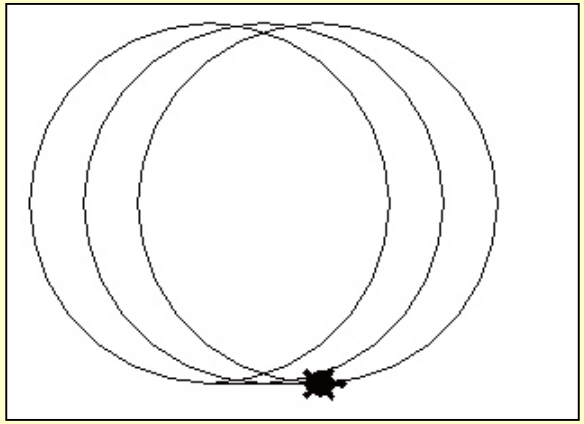
1. 변수를 다시 설명해보자.
2. 밑변이 10이고 높이가 10인 삼각형의 면적을 계산하는 프로그램을 작성해보자.
3. 변수 이름을 만들 때 지켜야 하는 규칙은 무엇인가?
4. 변수 이름의 첫 번째 글자로 허용되는 것은 무엇인가?
5. 파이썬에서 고유한 의미를 가지고 있는 단어들을 무엇이라고 하는가?

Lab: 변수는 어디에 응용할까?

- 다음과 같이 터틀 그래픽을 사용하여 반지름이 100픽셀인 3개의 원을 그리는 프로그램이 있다.

p67_lab.py

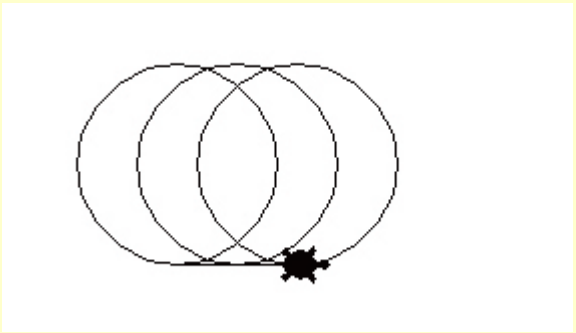
```
import turtle
t = turtle.Turtle()
t.shape("turtle")

t.circle(100) # 반지름이 100인 원이 그려 진다.
t.fd(30)
t.circle(100) # 반지름이 100인 원이 그려 진다.
t.fd(30)
t.circle(100) # 반지름이 100인 원이 그려 진다.

turtle.mainloop()
turtle.bye()
```



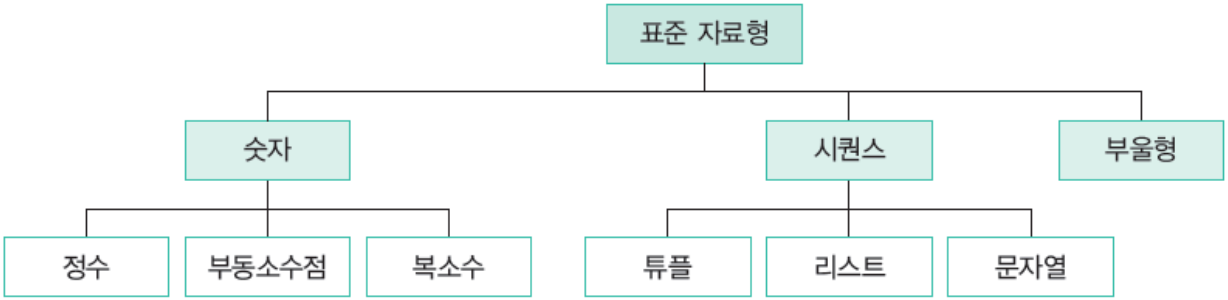
- 갑자기 원의 반지름을 50으로 변경하여서 다시 그려야 한다면 어떨까? -> 3개의 문장을 수정하여야 한다 -> 좀 더 쉬운 방법은 없을까? -> 원의 반지름을 변수로 표현하였다면 좀 더 쉽게 변경할 수 있다.

Solution

```
import turtle
t = turtle.Turtle()
t.shape("turtle")

radius = 50
t.circle(radius)      # 반지름이 50인 원이 그려 진다.
t.fd(30)
t.circle(radius)      # 반지름이 50인 원이 그려 진다.
t.fd(30)
t.circle(radius)      # 반지름이 50인 원이 그려 진다.

turtle.mainloop()
turtle.bye()
```



자료형



자료형	예
정수(int)	..., −2, −1, 0, 1, 2, ...
부동소수점수(float)	3.2, 3.14, 0.12
문자열(str)	'Hello World!', "123"

```
>>> radius = 10

>>> radius = 10.003

>>> radius = "Unknown"
```

파이썬은 변수에 어떤 자료형의 값이든 저장할 수 있다. 하지만 하나의 변수에 여러 종류의 값을 저장하는 것은 바람직하지 않다.
-> 변수에는 한 종류의 값만 저장하는 것이 바람직

자료형을 알려면

Syntax: 변수의 자료형

형식 type(수식)

예
```
>>> type(1234)
<class 'int'>
```

```
>>> type(12.30)              # float 형
<class 'float'>

>>> type("hello")            # str(문자열) 형
<class 'str'>
```

중간점검

1. 파이썬에는 어떤 자료형이 있는가? 3가지만 말해보자.
2. 파이썬 변수에 정수를 저장하였다가 실수를 저장하는 것도 가능한가? 그 이유는 무엇인가?
3. 변수의 자료형을 알려면 어떤 함수를 사용하는가?

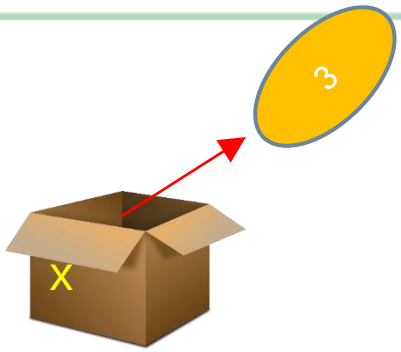
변수의 세부 구현 사항

- 변수에 저장되는 것은 실제 값이 아니고 객체의 참조값(주소)이다.

Syntax: 변수의 참조값

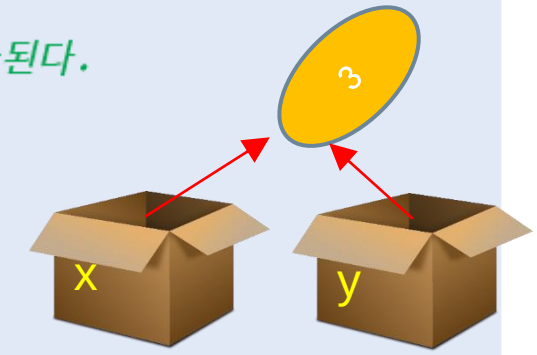
형식 id(변수)

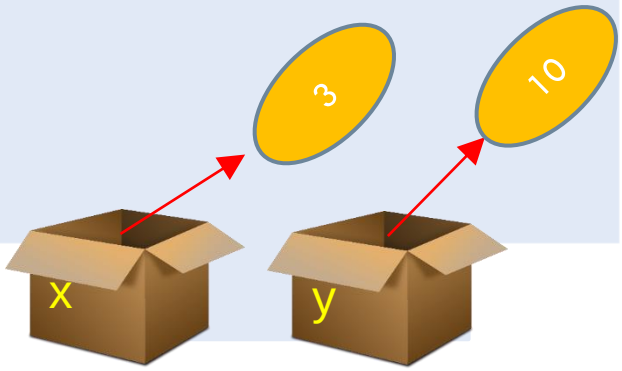
예
```
>>> x = 3
>>> id(x)
140721955779472
```



변수를 복사할 때

```
>>> x = 3
>>> y = x                # 변수 y에 변수 x의 참조값이 복사된다.

>>> id(x)
140721955779472

>>> id(y)                # 같은 주소를 가리킨다.
140721955779472

>>> y = 10               # 변수 y에 새로운 값이 할당되면 주소가 달라진다.
>>> id(y)
140721955779696
```



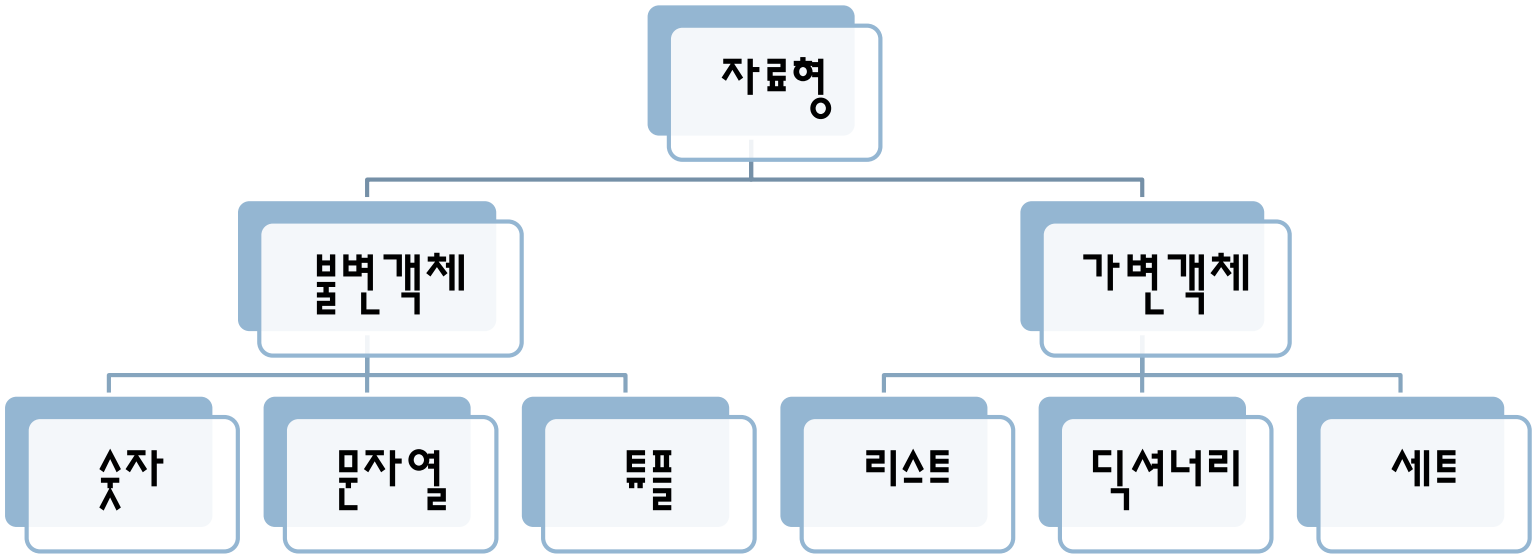


불변 객체와 가변 객체

- 불변 객체(immutable object)은 한번 만들어지면 변경할 수 없는 객체
  -> 우리가 변수에 저장된 값을 변경하면 값을 저장하는 새로운 객체가 생성되어서 새로운 객체의 참조값이 변수에 저장된다.
- 가변 객체(mutable object)는 변경할 수 있는 객체

불변 객체와 가변 객체의 예

- 불변 객체 : 정수, 실수, 문자열, 튜플
- 가변 객체: 리스트, 세트, 딕셔너리, ...



Lab: 별까지의 거리 계산하기

지구에서 가장 가까운 별은 어디일까? 정답은 태양이다. 태양 다음으로 가까운 별은 프록시마 센토리(Proxima Centauri) 별이라고 한다. 프록시마 센토리는 지구로부터 40×10^{12}km 떨어져 있다고 한다. 빛의 속도로 프록시마 센토리까지 간다면 시간이 얼마나 걸리는지 직접 계산해보기로 하자. 단위는 광년(light year)이다. 빛의 속도는 300000*km*/sec이다. 변수를 최대한 많이 사용하여 보자.

4.227972264501945 광년

Solution

p72_light_speed.py

```
##
#  이 프로그램은 프록시마 센토리까지 빛이 가는 시간을 계산한다.
#
speed = 300000.0                                  # 빛의 속도(300000km/sec)
distance = 40000000000000.0                       # 거리(40 x 10^12km)

secs = distance / speed                           # 걸리는 시간, 단위는 초
light_year = secs / (60.0*60.0*24.0*365.0)        # 광년으로 변환

print(light_year, "광년")
```

```
4.227972264501945 광년
```

주석(comment)

p73_tempcony.py

```
##
# 이 프로그램은 화씨 온도를 받아서 섭씨 온도로 변환한다.
#
ftemp = 100             # 화씨 온도 100를 변수에 저장한다.

ctemp = (ftemp-32.0)*5.0/9.0     # 화씨온도->섭씨온도
print("섭씨온도:", ctemp)        # 섭씨온도를 화면에 출력한다.
```

주석으로 컴파일러에게 무시되지만 프로그램에 대한 설명이나 메모를 붙이는 것이다.

주석의 용도가 하나 더 있다. 만약 코드중에 실행하고 싶지 않은 문장이 있다면

```
##
# 이 프로그램은 정수들의 합을 계산한다.
#
x = 100
y = 200
sum = x + y
#diff = x - y
print("합은 ", sum)
```

이 문장은 실행되지 않는다.

실행결과

```
합은 300
```

상수(constant)

- 한번 값이 변경되면 절대로 변경되지 않은 변수(값)
- 변수의 이름을 대문자로 하여서 일반적인 변수와 구분

```
TAX_RATE = 0.35
PI = 3.141592
MAX_SIZE = 100

…
…

INCOME = 1000                    # 소득과 소득세율이 변경되었다면 상수의 정의만 변경하면 된다.
TAX_RATE = 0.35

tax = INCOME * TAX_RATE
net_income = INCOME – tax
...
```

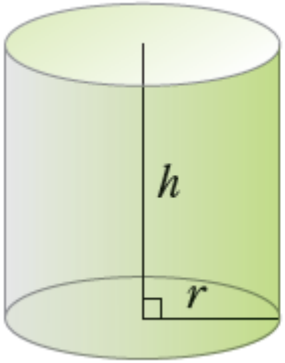
중간점검

1. 변수 k에 300을 저장하는 문장을 작성하여 보자.
2. 다음 프로그램은 사각형의 면적을 계산한다. 코드에 주석을 붙여보자.

```
width = 3.0
height = 5.0
print("면적=“, width*height)
```

Lab: 원기둥의 부피 계산

반지름= 5, 깊이= 10인 원기둥의 부피= 785.0



$V = \pi r^2 h$

원기둥의 부피

```
print(3.14*5*5*10)
```

변수를 사용하여서 탱크의 수위가 변경되어도 부피가 정확하게 제어판에 표시될 수 있도록 프로그램을 작성할 수 있다.

Solution

p76_volume.py

```
##
#          이 프로그램은 원기둥의 부피를 계산한다.
#
# 원주율을 나타내는 상수를 정의한다.
PI = 3.14

# 변수를 정의한다.
radius = 5
height = 10

# 부피를 계산한다.
volume = PI * radius * radius * height

# 결과를 출력한다.
print("반지름=", radius, "높이=", height, "원기둥의 부피=", volume)
```

산술 연산

연산자	기호	사용예	결과값
덧셈	+	7 + 4	11
뺄셈	–	7 – 4	3
곱셈	*	7 * 4	28
정수 나눗셈	//	7 // 4	1
실수 나눗셈	/	7 / 4	1.75
나머지	%	7 % 4	3

나눗셈과 나머지 연산

p78_remainder.py

```
p = 7
q = 4
print("나눗셈의 몫=", p // q)
print("나눗셈의 나머지=", p % q)
```

```
나눗셈의 몫= 1
나눗셈의 나머지= 3
```

```
today = 0
print( (today + 10) % 7 )     # 오늘부터 10일 후는 무슨 요일일까?
```

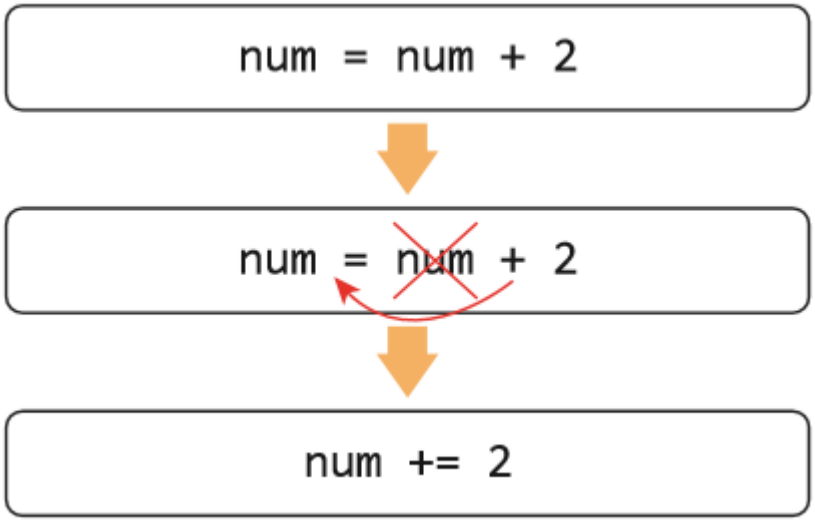
```
3
```

할당 연산(assignment operator)

```
x = 1
value = 3.0
x = (1/2)+3

x = y = z = 0

x, y, z = 10, 20, 30      # 한번에 여러 개의 변수 초기화

x, y = y, x               # x와 y의 값을 서로 교환한다.

   # 위와 같은 문장이 허용되지 않는 언어라면, 다음과 같이 임시 변수를 사용해야 한다.
   tmp = x
   x = y
   y = temp
```

복합 연산자(compound operator)

- 복합 연산자(compound operator)란 +=처럼 대입 연산자와 다른 연산자를 합쳐 놓은 연산자이다.



복합 연산자

복합 연산자	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

p80_compound.py

```
x = 1000
print("초기값 x=", x)
x += 2;
print("x += 2 후의 x=", x)
x -= 2;
print("x -= 2 후의 x=", x)
```

```
초기값 x= 1000
x += 2 후의 x= 1002
x -= 2 후의 x= 1000
```

지수(power) 계산

- 지수를 계산하려면 ** 연산자를 사용한다.

```
>>> 2 ** 7
128
```

- 원리금 합계를 복리로 계산

p81_power.py

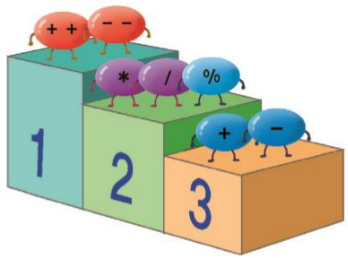
```
a = 1000                    # 원금
r = 0.05                    # 이자율
n = 10                      # 기간
result = a*(1+r)**n         # 원리금 합계

print("원리금 합계=", result)
```

```
원리금 합계= 1628.894626777442
```

우선 순위

$x + \underbrace{y * z}_{①}$ $\quad\underbrace{}_{②}$

$\underbrace{(x + y)}_{①} * z$ $\quad\underbrace{}_{②}$



연산자	설명
**	지수 연산자
~, +, -	단항 연산자
*, /, %, //	곱셈, 나눗셈, 나머지 연산자
+, -	덧셈, 뺄셈
>>, <<	비트 이동 연산자
&	비트 AND 연산자
^, |	비트 XOR 연산자, 비트 OR 연산자
<=, <, >, >=	비교 연산자
<>, ==, !=	동등 연산자
=, %=, /=, //=, -=, +=, *=, **=	대입, 복합 연산자
is, is not	동등 연산자
in, not in	소속 연산자
not, or, and	논리 연산자

중간점검

1. 복합 대입 연산자 x *= y의 의미를 설명하라.
2. 10%6의 값은 무엇인가?
3. 나눗셈 연산인 10//6의 값은 얼마인가?
4. 다음의 할당문에서 무엇이 잘못되었는가?

   3 = x

1. 다음의 할당문이 실행된 후의 변수 a, b, c의 값은?

   a = b = c = 100

Lab: 복리 계산

- 1626년에 아메리카 인디언들이 뉴욕의 맨하탄섬을 단돈 60길더(약 24달러)에 탐험가 Peter Minuit에게 팔았다고 한다. 380년 정도 경과한 2005년 맨하탄 땅값은 약 8조 달러라고 한다. 인디언들은 큰 손해를 보았다고 할 수 있다.
- 하지만 만약 인디언이 24달러를 은행의 정기예금에 입금해두었다면 어떻게 되었을까? 예금 금리는 복리로 8%라고 가정하자. 그리고 382년이 지난 후에는 원리금을 계산하여 보자.

Solution

p85_invest.py

```
init_money = 24
interest = 0.08
years = 382
print(init_money*(1+interest)**years)
```

```
140632545501736.62
```

- 놀랍게도 382년이 지나면 원리금은 140조 달러가 되어 현재 땅값을 넘어서게 된다. 만약 이자율이 약간이라도 더 높으면 그 차이는 더 벌어질 것이다. 이것이 바로 '복리효과' 이다. 재투자가 이루어지면 재산이 급격하게 증식되는 것이다.

타입 변환과 반올림

- 정수와 부동소수점수를 동시에 사용하여 수식을 만들면 파이썬은 자동으로 정수를 부동소수점수로 변환한다 -> 자동적인 타입변환

```
>>> 3 * 1.23
3.69
```

- 강제로 타입변환

```
>>> x = 3.14
>>> int(x)
3

>>> y = 3
>>> float(y)
3.0
```

- round() : 실수를 반올림

```
>>> x = 1.723456
>>> round(x)
2

>>> x = 1.723456
>>> round(x, 2)
      #소수점 2번째자리에서 반올림
1.72
```

반올림

- 물건 값의 7.5%가 부가세라고 하자. 물건값이 12345원일 때, 부가세를 소수점 2번째 자리까지 계산하는 프로그램

p87_tax.py

```
price = 12345
tax = price * 0.075
tax = round(tax, 2)
print(tax)
```

```
925.88
```

문자열

- 컴퓨터에게는 숫자가 중요하지만 인간은 주로 문자열(string)를 사용하여 정보를 표현하고 저장하므로 문자열의 처리도 무척 중요하다.

문자열을 만드는 방법

```
>>> "Hello"
'Hello'

>>> msg = "Hello"
>>> msg
'Hello'
>>> print(msg)
Hello

>>> msg = ‘Hello’      # 작은 따옴표(‘…’)를 사용해도 된다.

>>> msg = “Hello’
SyntaxError: ….    #큰 따옴표로 시작했다가 작은 따옴표로 끝내면 문법적 오류
```

왜 큰 따옴표와 작은 따옴표를 동시에 사용할까

```
>>> message="철수가 "안녕"이라고 말했습니다."
SyntaxError: invalid syntax

>>> message="철수가 '안녕'이라고 말했습니다."
>>> print(message)
철수가 '안녕'이라고 말했습니다.
```

세 따옴표(docstring)

p89_docstring.py

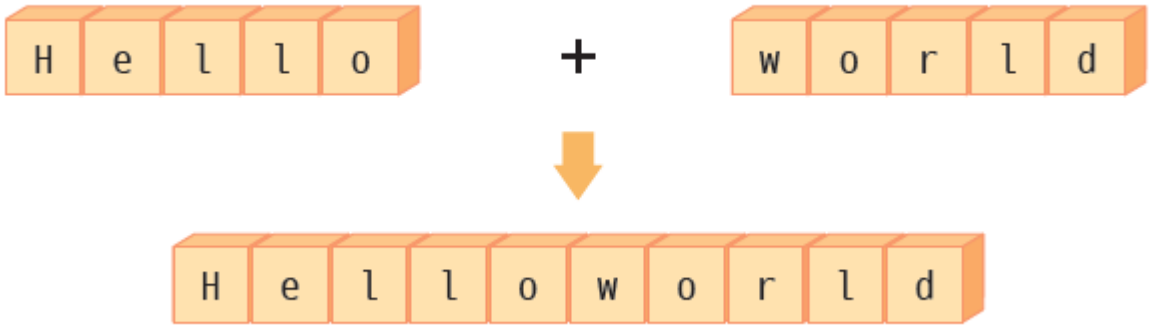
```
a = """TWINKLE, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky."""

print(a)
```

```
TWINKLE, twinkle, little star,
How I wonder what you are!
Up above the world so high,
Like a diamond in the sky.
```

- 세따옴표를 이용하면 문자열의 줄바꿈이나 스페이스 등이 그대로 유지된 채 출력된다.

문자열의 결합

```
>>> "Hello " + "World!"
'Hello World!'
```



문자열의 반복

```
>>> lines = "-" * 30
>>> lines
'------------------------------'



>>> song = "뚜 루루 뚜루 " * 5
>>> song
'뚜 루루 뚜루 뚜 루루 뚜루 뚜 루루 뚜루 뚜 루루 뚜루 뚜 루루 뚜루 '
```

100과 "100"은 구별하여야 한다


```
>>> print(100+200 )
300
>>> print("100"+"200")
100200
```

숫자와 문자열의 변환

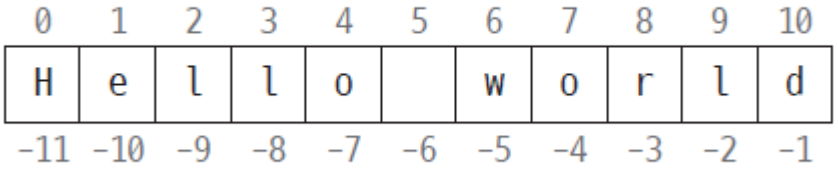
```
>>> movie = "Terminator" + 3
TypeError: can only concatenate str (not "int") to str


>>> movie = "Terminator" + str(3)
>>> movie
'Terminator3'


>>> price = int("100")                  # price = 100
>>> PI = float("3.14")                  # PI = 3.14
```

특수 문자열

특수 문자열	의미
\n	줄바꿈 문자
\t	탭문자
\\	역슬래시 자체
\"	큰따옴표 자체
\'	작은따옴표 자체

```
>>> print("말 한마디로\n천냥빚을 갚는다")
말 한마디로
천냥빚을 갚는다
```

문자와 문자열



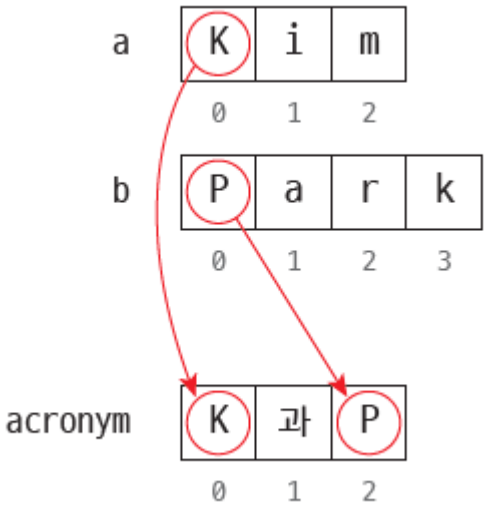

```
>>> s = "Hello World"
>>> s[0]
'H'

>>> s = "Hello World"
>>> s[-1]
'd'
```

예제

p93_acronym.py

```
a = "Kim"
b = "Park"
acronym = a[0] + "과" + b[0]
print(acronym)
```

```
K과P
```



중간점검

1. 문자열 “100”을 정수 100으로 변환하는 명령문을 작성하라.
2. 정수 100을 문자열 “100”으로 변환하는 명령문을 작성하라.
3. 다음과 같이 3개의 문자열이 3개의 변수에 저장되어 있다. 첫글자만을 따서 “BTW"라고 만들고 싶다. 어떻게 하면 되는가?

```
a = "By"
b = "The"
c = "Way"
```

문자열 메소드

- 파이썬에서 문자열은 객체이다. 객체(object)란 프로그래밍에서 관련 있는 변수와 함수를 하나로 묶은 것이다.
- 객체안에 정의된 함수를 메소드(method)라고 한다. 메소드를 사용하면 객체에 대하여 다양한 작업을 수행할 수 있다.


```
name = "Harry Parter"
lower_name = name.lower()                          # 'harry parter'

name = "Harry Parter"
new_name = name.replace("Parter", "Porter")        # Harry Porter
```

Lab: 뢧 기자 만들기

- 사용자에게 경기장, 점수, 이긴 팀, 진 팀, 우수 선수를 질문하고 변수에 저장한다. 이들 문자열에 문장을 붙여서 기사를 작성한다.

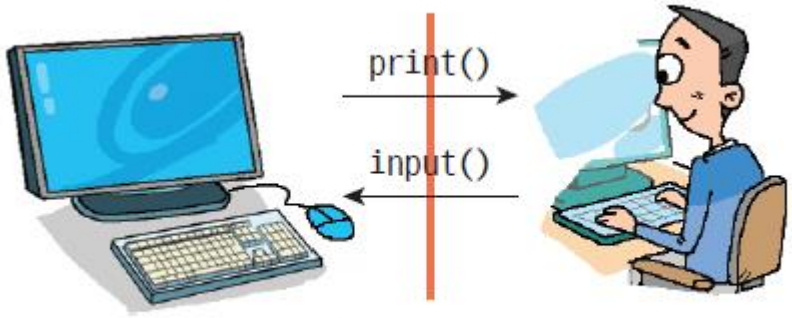
```
경기장은 어디입니까?서울
이긴팀은 어디입니까삼성
진팀은 어디입니까?LG
우수선수는 누구입니까?홍길동
스코어는 몇대몇입니까?8:7


==========================================
오늘 서울 에서 야구 경기가 열렸습니다.
삼성 과 LG 은 치열한 공방전을 펼쳤습니다.
홍길동 이 맹활약을 하였습니다.
결국 삼성 가 LG 를 8:7 로 이겼습니다.
==========================================
```

Solution

p95_robot.py

```
# 사용자의 대답을 변수에 저장한다.
stadium = input("경기장은 어디입니까?")
winner = input("이긴팀은 어디입니까")
loser = input("진팀은 어디입니까?")
vip = input("우수선수는 누구입니까?")
score = input("스코어는 몇대몇입니까?")

# 변수와 문자열을 연결하여 기사를 작성한다.
print("")
print("==========================================")
print("오늘", stadium, "에서 야구 경기가 열렸습니다.")
print(winner, "과", loser, "은 치열한 공방전을 펼쳤습니다.")
print(vip, "이 맹활약을 하였습니다.")
print("결국", winner,"가", loser,"를 ", score,"로 이겼습니다.")
print("==========================================
```

입력과 출력



사용자 인터페이스

Syntax: input() 함수

형식 변수 = input(안내메시지)

예 x = input("이름을 입력하시오: ")

변수

안내 메시지를 출력하고 사용자가 입력한 값을 문자열 형태로 반환한다.

input() 함수

```
>>> name = input("이름을 입력하시오: ")
이름을 입력하시오: 홍길동

>>> print(name)
홍길동
```

p97_input.py

```
name = input("이름을 입력하시오: ")
print(name, "씨, 안녕하세요?")
print("파이썬에 오신 것을 환영합니다.")
```

```
이름을 입력하시오: 홍길동
홍길동 씨, 안녕하세요?
파이썬에 오신 것을 환영합니다.
```

정수 입력

p97_sum.py

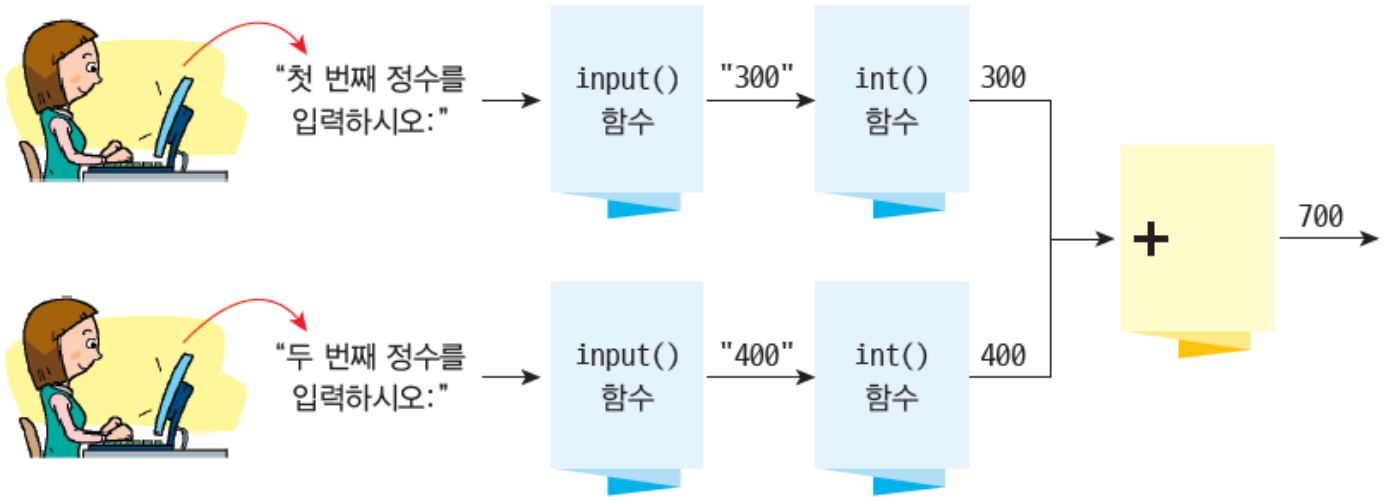
```
x = input("첫 번째 정수를 입력하시오:")
y = input("두 번째 정수를 입력하시오:")
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수를 입력하시오: 300
첫 번째 정수를 입력하시오: 400
합은  300400
```

정수 입력

```
s1 = input("첫 번째 정수를 입력하시오:")
x = int(s1)                                  # 문자열을 정수로 변환한다.
s2 = input("두 번째 정수를 입력하시오:")
y = int(s2)                                  # 문자열을 정수로 변환한다.
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수를 입력하시오: 300
첫 번째 정수를 입력하시오: 400
합은  700
```



정수 입력

```
x = int(input("첫 번째 정수를 입력하시오:"))
y = int(input("두 번째 정수를 입력하시오:"))
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수를 입력하시오: 300
첫 번째 정수를 입력하시오: 400
합은  700
```

중간점검

1. 사용자로부터 2개의 정수를 받아서 사칙연산(+, -, *, /)을 한 후에 결과를 출력하는 프로그램을 작성해보자.
2. 사용자의 이름을 물어보고 이어서 2개의 정수를 받아서 덧셈을 한 후에 결과를 출력하는 다음과 같은 프로그램을 작성해보자.

```
이름을 입력하시오: 홍길동
홍길동 씨, 안녕하세요?
파이썬에 오신 것을 환영합니다.
첫 번째 정수를 입력하시오: 300
두 번째 정수를 입력하시오: 400
300 과 400 의 합은 700입니다.
```

부동소수점수 입력

p99_float.py

```
SQMETER_PER_P = 3.3

area = float(input ("면적(제곱미터):"))
py = area / SQMETER_PER_P
print(py, "평")
```

```
면적(제곱미터):25.6
7.757575757575759 평
```

```
SQMETER_PER_P = 3.3

area = eval(input ("면적(제곱미터):"))
py = area / SQMETER_PER_P
print(py, "평")
```

```
면적(제곱미터):25.6
7.757575757575759 평
```

오류가 발생할 수 있다

```
x = int(input("첫 번째 정수를 입력하시오:"))
y = int(input("두 번째 정수를 입력하시오:"))
sum = x + y
print("합은 ", sum)
```

```
첫 번째 정수를 입력하시오:3.14
ValueError: invalid literal for int() with base 10: '3.14'
```

- 사용자가 3.14를 입력했기 때문에 이것을 정수로 변경하려고 했으나 소수점이 있어서 오류가 발생 -> 해결 : 부동소수점으로 먼저 변경한 후 정수로 변경하면 된다.

```
x = int(float(input("첫 번째 정수를 입력하시오:")))
y = int(float(input("두 번째 정수를 입력하시오:")))
sum = x + y
print("합은 ", sum)
```

변수와 문자열을 동시에 출력할때

```
x = 100
y = 200
print(x, "와 ", y, "의 합=", x+y)
```

```
100 와  200 의 합= 300
```

```
x = 100
y = 200
print(f"{x}와 {y}의 합={x+y}")
```

```
100와 200의 합=300
```

형식화된 출력

p101_sm2pyung.py

```
SQMETER_PER_P = 3.3

area = eval(input ("면적(제곱미터):"))
py = area / SQMETER_PER_P
print("%.2f" % py, "평")          # 출력 7.76 평
```

```
면적(제곱미터):25.6
7.76 평
```

Syntax: print() 함수

형식 형식문자열 % (값1, 값2, ..., 값n)

예

```
p = 7.76
print("%10.2f" % py)
```

Lab: 친근하게 대화하는 프로그램

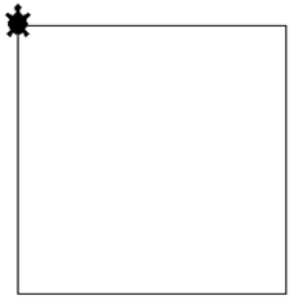
- 변수를 사용하여 사용자의 이름과 나이를 문자열 형태로 기억했다가 출력할 때 사용하는 프로그램을 작성해보자.

```
안녕하세요?

이름이 어떻게 되시나요? 홍길동
만나서 반갑습니다. 홍길동씨
이름의 길이는 다음과 같군요: 3

나이가 어떻게 되나요? 21
내년이면 22이 되시는군요.
```

```
##                                                    p102_char_bot.py
#          이 프로그램은 사용자와 친근하게 대화한다.
#
print("안녕하세요?")
name = input("이름이 어떻게 되시나요? ")
print("만나서 반갑습니다. " + name + "씨")

print("이름의 길이는 다음과 같군요:", len(name))

age = int(input("나이가 어떻게 되나요? "))
print("내년이면 "+ str(age+1) + "이 되시는군요.")
```

Lab: 사각형 그리기

사각형의 크기는 얼마로 할까요? 200



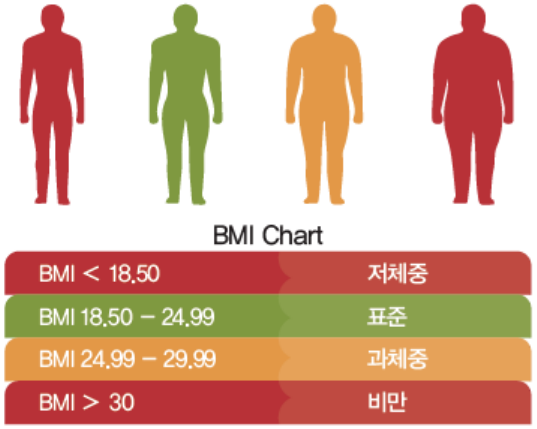
```
##                                                                    p104_rect.py
#          이 프로그램은 사용자로부터 크기를 받아서 사각형을 그린다.
#
import turtle
t = turtle.Turtle()
t.shape("turtle")

# 사용자로부터 사각형의 크기를 받아서 size라는 변수에 저장한다.
size = int(input("사각형의 크기는 얼마로 할까요? "))

# 사각형을 다음과 같은 코드로 그린다. 이때 변수 size를 사용하자.
t.forward(size)          # size 만큼 거북이를 전진시킨다.
t.right(90)              # 거북이를 오른쪽으로 90도 회전시킨다.
t.forward(size)
t.right(90)
t.forward(size)
t.right(90)
t.forward(size)

turtle.mainloop()
turtle.bye()
```

Lab: BMI 계산하기

- 사용자로부터 신장과 체중을 입력받아서 BMI 값을 출력하는 프로그램을 작성하여 보자.

```
몸무게를 kg 단위로 입력하시오: 85.0
키를 미터 단위로 입력하시오: 1.83
당신의 BMI= 25.381468541909282
```

$$BMI = \frac{\text{체중(킬로그램)}}{\text{신장(미터)의 제곱}}$$



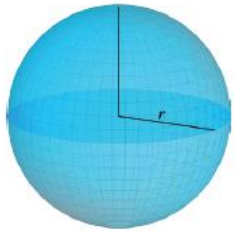
p105_bmi.py

```
##
#          이 프로그램은 BMI를 계산한다.
#
weight = float(input("몸무게를 kg 단위로 입력하시오: "))
height = float(input("키를 미터 단위로 입력하시오: "))

bmi = (weight / (height**2))
print("당신의 BMI=",  bmi)
```

Lab: 구의 부피 계산하기

- 반지름이 5m인 구의 부피를 계산하는 파이썬 프로그램을 작성해보자.

```
반지름을 입력하시오: 5.0
구의 부피= 523.5986666666666
```



$$\frac{4}{3}\pi r^2$$

p106_shpere.py

```
##
#          이 프로그램은  구의 부피를 계산한다.
#

# 사용자에게 구의 반지름을 입력하도록 한다. 구의 반지름을 문자열에서 실수로 변환한다.
r = float(input("반지름을 입력하시오: "))

# 구의 부피를 공식을 이용하여 계산한다.
volume = (4.0/3.0) * 3.141592 * r**3

# 구의 부피를 화면에 출력한다.
print("구의 부피=",volume)
```

Lab: 자동판매기 프로그램

- 자동 판매기를 시뮬레이션하는 프로그램을 작성하여 보자.

```
물건값을 입력하시오: 750
1000원 지폐개수: 1
500원 동전개수: 0
100원 동전개수: 0



500원= 0 100원= 2 10원= 5 1원= 0
```

p107_atm.py

```
##
#           이 프로그램은 자판기에서 거스름돈을 계산한다.
#
itemPrice = int(input("물건값을 입력하시오: "))
note = int(input("1000원 지폐개수: "))
coin500 = int(input("500원 동전개수: "))
coin100 = int(input("100원 동전개수: "))

change = note*1000 + coin500*500 + coin100*100 - itemPrice

# 거스름돈(500원 동전 개수)을 계산한다.
nCoin500 = change//500
change = change%500

# 거스름돈(100원 동전 개수)을 계산한다.
nCoin100 = change//100
change = change%100

# 거스름돈(10원 동전 개수)을 계산한다.
nCoin10 = change//10
change = change%10

# 거스름돈(1원 동전 개수)을 계산한다.
nCoin1 = change

print("500원=", nCoin500, "100원=", nCoin100, "10원=", nCoin10, "1원=", nCoin1)
```

이번 장에서 배운 것

- 변수는 값을 저장하는 상자와 같은 것으로 저장된 값은 나중에 유용하게 사용될 수 있다.
- 문자열은 큰따옴표("...")나 작은 따옴표('...')을 사용할 수 있다.
- 덧셈, 뺄셈, 곱셈, 나눗셈을 위하여 +, -, *, / 기호를 사용한다.
- 지수 연산자는 _____이다.
- 나눗셈에서 정수로 몫을 계산하려면 // 연산자를 사용한다.
- 나눗셈에서 나머지를 계산하려면 % 연산자를 사용한다.
- 우선순위가 높은 연산자가 먼저 계산된다.
- *와 /가 +와 –보다 우선순위가 높다.
- 연산자의 우선 순서를 변경하려면 괄호를 사용한다.
- input() 함수를 이용하여 사용자로부터 문자열을 받을 수 있다.
- 문자열을 정수로 변경하려면 _____함수를 사용한다.
- 문자열을 실수로 변경하려면 float() 함수를 사용한다.
- 정수나 실수를 문자열로 변경하려면 str()를 사용한다.
- "\n"은 줄바꿈을 나타내는 특수 문자열이다.

Q & A