Python 프로그램

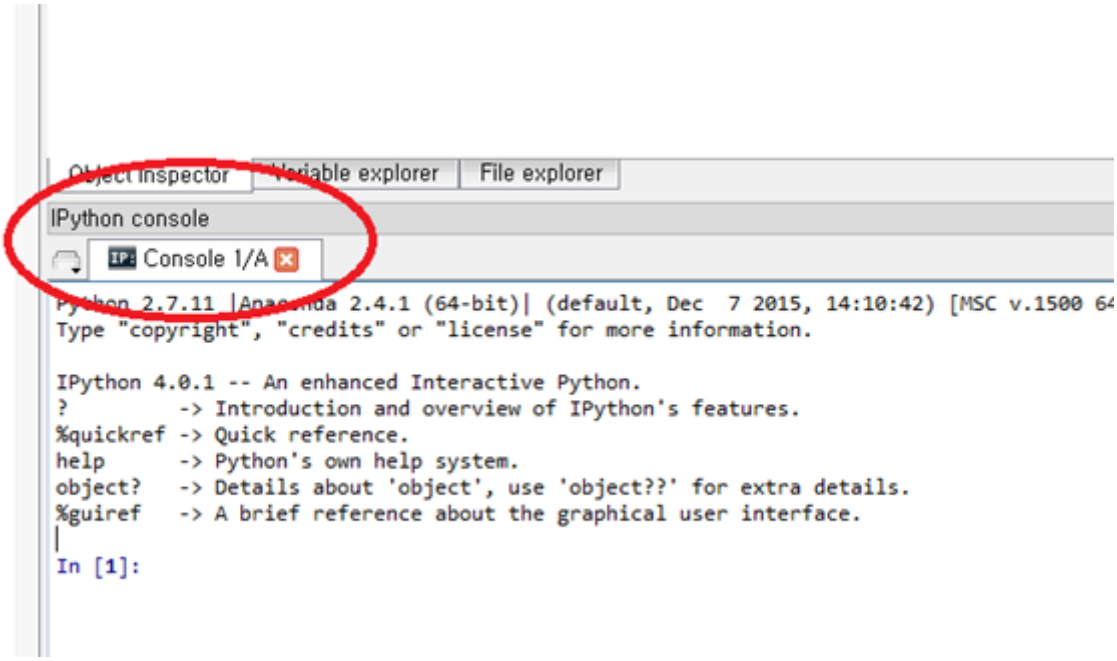
- Python 프로그램의 소스 코드는 일련의 정의와 명령문으로 구성된다.
- Python 프로그램이나 소느 코드는 스크립트(script)라고도 불림.
- 프로그램을 작성하는 사람이 command 혹은 statement라고 불리는 문장들을 작성하여, 이 문장들을 통해 Python 인터프리터가 프로그램 작성자의 목적에 맞는 일을 하도록 지시한다.
- 다음은 print 문장을 이용한 간단한 프로그램이다.
  - In [1]: 의 오른쪽의 내용이 우리가 작성하는 프로그램의 내용이며 그 다음 줄의 내용이 프로그램의 실행 결과이다.
- 이 자료는 Python 3.6을 기준으로 작성되었다.

In [1]:

```
print('This is my first Python program.')
```

```
This is my first Python program.
```

만약 Spyder를 이용 중이라면, Spyder 오른쪽 하단의 IPython Console을 이용하여 위의 문장을 실행시켜 보자. Spyder는 Anaconda 배포판을 통해 설치할 수 있는 Python 개발 환경이다. 아래 그림의 In [1]: 오른쪽 부분에 print 'This is my first Python program.' 를 입력하고 엔터를 입력한다.



데이터 타입

프로그래밍에 사용되는 데이터들은 데이터 타입(데이터 형)을 구분한다.

Python에는 크게 세 가지 데이터 타입이 존재

- 숫자형
  - 5, 10.3, -2, -1.02
  - 사칙연산 등을 수행할 수 있음
  - 숫자형 데이터는 다시 정수형과 실수형으로 구분할 수 있다.
- 문자열
  - "apple", "Hello, world!", 'Yeungnam University', "15"
  - 큰 따옴표나 작은 따옴표로 표현
- 부울(Bool)
  - True, False

수치와 연산

- 숫자형 데이터에는 사칙연산이나 나머지, 제곱 등의 연산을 적용할 수 있다.
- IPython 콘솔은 마치 계산기처럼 이용 가능하다.

```
In [2]:
```

```
3 + 4
```

```
Out[2]:
```

```
7
```

```
In [3]:
```

```
3 - 4
```

```
Out[3]:
```

```
-1
```

```
In [4]:
```

```
3 * 4
```

```
Out[4]:
```

```
12
```

- Python 2.x에서는 정수끼리 나누기 연산의 결과는 정수로 나타난다. 소수점 아래의 숫자는 버린다.
- 3/4 -> 0
- 나누기를 통하여 실수 결과를 얻고자 할 때는 연산되는 수를 실수형으로 먼저 바꾼 후 진행한다.
- 3/4.0 혹은 3.0/4

- Python 3.x에서는 정수끼리 나누기 연산의 결과는 실수로 나타난다.

In [5]:

```
3/4
```

Out[5]:

0.75

다음은 나머지 연산을 수행한다.

In [6]:

```
5%2
```

Out[6]:

1

제곱이나 거듭제곱은 **를 이용한다.

In [7]:

```
5**2
```

Out[7]:

25

In [8]:

```
2**3
```

Out[8]:

8

복잡한 연산의 순서는 기본적인 수학 규칙을 따른다.

In [9]:

```
2+2*2
```

Out[9]:

6

In [10]:

```
(3-5)*(6+(2-4))
```

Out[10]:

-8

변수

- 계산을 하다보면 특정 변수를 생성하여 값을 지정하고 변수를 통하여 수식을 계산하는 것이 더 편리할 경우가 많다.
- 수학에서 x=10과 같이 변수와 그 값을 지정하듯이 Python에서도 등호(=)를 이용하여 변수의 값을 지정한다.
- 변수의 이름, 변수명은 기본적인 규칙을 지키는 선에서 마음대로 지어주면 된다.
  - 이 규칙에 대해서는 잠시 후에 살펴본다.
- 대입 연산에서는 반드시 왼쪽에 변수명이 등장한다.
- 정수형 변수 지정
  - 다음을 'my_integer는 10과 같다'고 해석을 해서는 안 된다. 'my_integer에 10을 대입한다'와 같이 해석한다.

In [11]:

```
my_integer = 10
```

- 실수형 변수 지정

In [12]:

```
my_number = 5.2
```

- 문자열 변수 지정

In [13]:

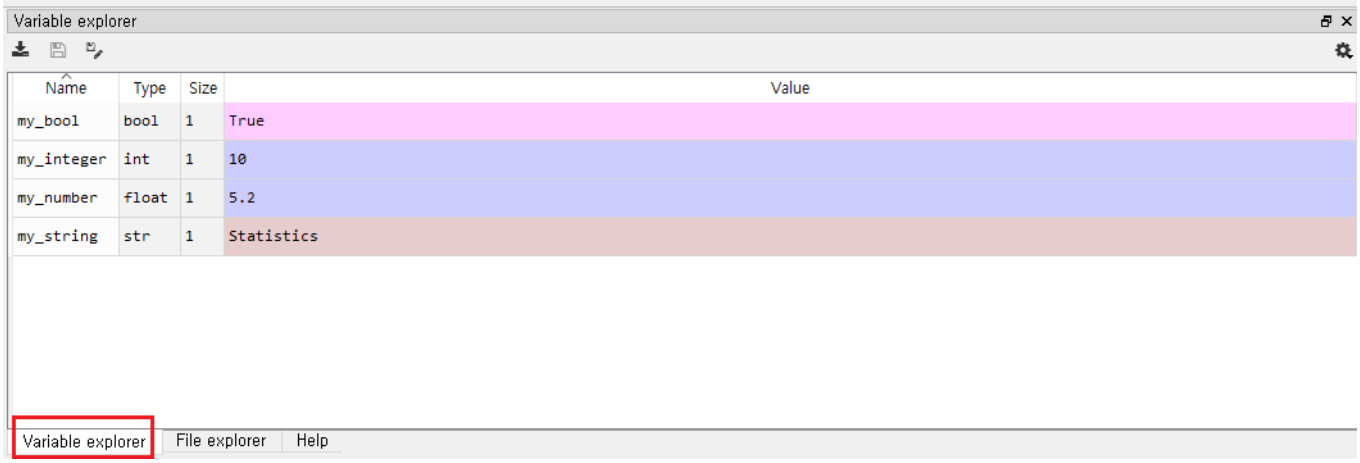
```
my_string = "Statistics"
```

- 부울형 변수 지정

In [14]:

```
my_bool = True
```

만약 Spyder를 이용 중이라면 Variable explorer를 이용하여 입력한 변수의 내용을 확인할 수 있다.



- 위의 예제들처럼 대입 연산의 오른쪽에는 상수값이 오거나, 다음의 예제처럼 기존의 변수 혹은 기존의 변수를 이용한 식 등이 올 수 있다.

In [15]:

```
my_second_integer = my_integer + 1
```

변수명 규칙

- Python 변수명은 문자나 _로 시작
  - _the_string
  - my_variable
- 첫 글자가 아닌 나머지 부분은 문자, 숫자, _가 올 수 있음
  - password1
  - n00b
  - un_der_score
- 대소문자 구별 : X와 x는 서로 다른 변수
- 변수명을 짓는 방법
  - 알아보기 쉽게 짓는다.
    - apartmentname -> apartment_name 혹은 apartmentName
  - 무슨 변수를 나타내는지 알아보도록 짓는다.
    - afmowpv : 아무 의미 없어 보이며 이렇게 짓지 않음
    - average_num_students : 변수명이 다소 긴 것처럼 느껴지지만, 학생 수의 평균을 의미하는 변수 임을 추측할 수 있음

변수 지정에 있어 Python의 특징

- 많은 다른 프로그래밍 언어의 경우 변수형을 따로 선언해 주어야 함. 예를 들어, C언어 등의 경우
  - int n = 10
    - 정수(integer)형 변수
  - double x = 2.12
    - double precision을 가지는 실수형 변수
  - char c = 'a'
    - 문자(character)형 변수
- Python은 int, double, char 등과 같은 변수형에 대한 keyword를 따로 선언해 주지 않는다.

Dynamic types

- Python에서는 데이터 type을 따로 명시하는 것이 아니기 때문에, 프로그램 중간에 데이터 type이 바뀔 수 있다.
  - x = 10
    …
    x = 'John'
    …
    x = True
- 그러나, 하나의 변수가 한 프로그램 내에서 데이터 타입이 변하는 것은 바람직하지 않으며, 동일한 데이터 타입을 유지하도록 한다.
- 반면 변수의 값은 계속 변할 수 있다.

변수값 재지정

한 번 지정한 변수값을 얼마든지 새 변수값으로 바꿀 수 있음

In [16]:

```
a = 1
a = 2
a = 2.1
b = "Hi"
b = "Hello"
c = True
c = False
```

다음의 예제에서는 radius의 값이 계산 결과에 따라 바뀌는 모양을 나타내고 있다.

In [17]:

```
pi = 3
radius = 11
area = pi * (radius**2)
radius = 14
```

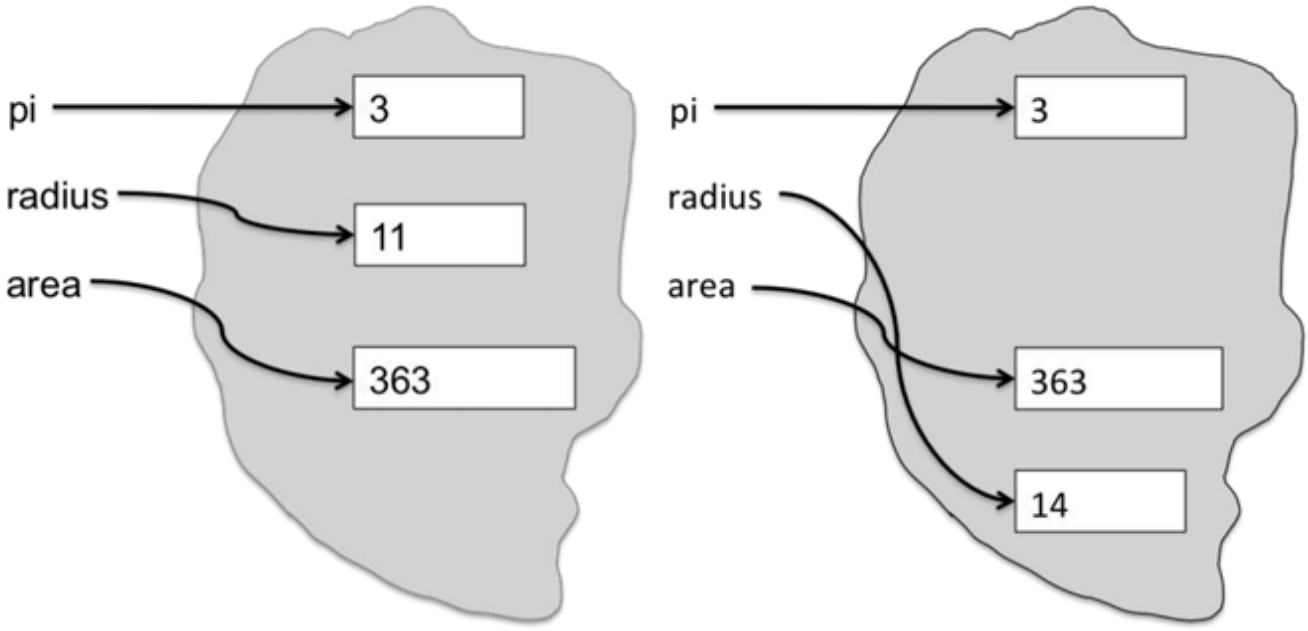


그림 출처 : Introduction to Computation and Programming Using Python, John V. Guttag

multiple assignment

Python에서는 다음과 같이 여러 개의 변수에 여러 값들을 한 번에 지정할 수 있다.

In [18]:

```
x, y = 2, 3
x, y = y, x
print('x = ', x)
print('y = ', y)
```

```
x =  3
y =  2
```

문자열 출력해 보기

프로그래밍의 가장 기본 예제를 연습해 보자. Python은 인터프리터 언어로서 매우 간단한 Hello world 프로그램을 가진다.

In [19]:

```
print("Hello, world!")
```

```
Hello, world!
```

In [20]:

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

```
Hello, world!
```

따옴표로 둘러싸여 있는 부분은 문자열을 의미하며 print 명령을 통해 해당 문자열 - 여기서는 Hello, world! - 을 화면에 출력하라는 뜻이다.

- 큰 따옴표, 작은 따옴표 모두 가능
- Python 2.x에서는 print 명령에 괄호 생략 가능
- Python 3.0 이상에서는 print가 함수로 변경되었기 때문에, 반드시 괄호를 이용해야 한다.
- https://docs.python.org/3/whatsnew/3.0.html#print-is-a-function (https://docs.python.org/3/whatsnew/3.0.html#print-is-a-function)

문자열 입력 받기

input() 함수를 사용하여 콘솔로부터 문자열을 입력 받을 수 있음

- python 2.x에서는 raw_input() 함수를 이용한다.

In [22]:

```
input_string = input("Input any string : ")
```

```
Input any string : Simple test
```

- "Input any string : " 부분은 콘솔 창에 다시 나타나는 부분으로 원하는 대로 바꾸어 쓸 수 있음
- 콘솔에서 입력받은 문자열은 등호 왼쪽의 변수에 저장된다.
  - 여기서는 input_string
- print 명령을 이용하여 저장된 내용을 다시 출력해 보자.

In [23]:

```
print(input_string)
```

```
Simple test
```

부울(Boolean)형

- True 혹은 False로 구성된 자료형이다.
- Boolean type의 변수는 프로그래밍에서 중요한 역할을 담당
  - 조건에 따라 처리를 달리하는 프로그램을 작성할 때 유용

In [24]:

```
2 > 0.9
```

Out[24]:

```
True
```

In [25]:

```
3 < -3
```

Out[25]:

```
False
```

In [26]:

```
a = True
```

In [27]:

```
b = False
```

- True의 값을 가지는 변수 a
- False의 값을 가지는 변수 b

비교연산과 대입연산

등호를 하나만 사용하는 경우 이를 대입연산자라 함

In [28]:

```
x = 5
```

- x라는 변수에 5라는 정수를 대입
- 따라서 4=4와 같은 문장은 잘못된 문장
  - 숫자 4에 4를 대입할 수 없음

등호를 두 개를 연속해서 사용하는 경우 : 비교연산

In [29]:

```
4 == 4
```

Out[29]:

True

In [30]:

```
"Korea" == "Japan"
```

Out[30]:

False

대입연산 예제

다음의 결과들을 순차적으로 생각해 보자.

In [31]:

```
x = 2
x = x + 4
x = x * x
x = x % 5
```

In [32]:

```
print(x)
```

1

비교연산

비교연산의 종류

비교연산자	대응되는 수학기호
>	$>$
>=	$\geq$
<	$<$
<=	$\leq$
!=	$\neq$
==	$=$

비교연산자를 사용한 식이 옳을 경우 True를, 그렇지 않을 경우 False를 반환

In [33]:

```
egg = 138
spam = 370
```

In [34]:

```
egg == spam
```

Out[34]:

```
False
```

In [35]:

```
egg != spam
```

Out[35]:

```
True
```

복합대입연산자

연산과 대입을 동시에 할 수 있음

In [36]:

```
i = 1
i += 2
i
```

Out[36]:

```
3
```

In [37]:

```
j = 10
j -= 1
j
```

Out[37]:

```
9
```

주석(comment)

- 코드에 설명을 넣고 싶을 때
- 한 줄로 주석 넣기, #으로 시작
  - # This code solves the P vs NP problem.
- 여러 줄에 걸쳐 주석을 넣을 때, """를 이용
  - """
    Turing used this code to crack the Enigma.
    With this code, anybody hack a server like a pro.
    """

문자열(string)

파이썬에서는 문자열(string)을 쉽게 다룰 수 있다.

In [38]:

```
name = "Kyungsub Lee"
```

In [39]:

```
print(name)
```

```
Kyungsub Lee
```

다음은 숫자 45가 아니라 문자열로서 "45"로 사용되었음을 유의

In [40]:

```
age = "45"
```

In [41]:

```
print(age)
```

```
45
```

문자열 연결 (concatenation)

Python에서 문자열은 + 기호를 이용하여 쉽게 연결할 수 있다.

In [42]:

```
first_name = "Kyungsub"
family_name = "Lee"
full_name = first_name + " " + family_name
print(full_name)
```

```
Kyungsub Lee
```

In [43]:

```
print("My age is " + str(20) + ".")
```

```
My age is 20.
```

문자열 관련 예제

다음의 결과를 테스트해 보자.

In [44]:

```
'a' + 'bcde'
```

Out[44]:

```
'abcde'
```

In [45]:

```
5 * 'ab'
```

Out[45]:

'abababababab'

In [46]:

```
3 + 'three'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-46-3b9786c72eb4> in <module>()
----> 1 3 + 'three'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [47]:

```
'a' < 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-47-5e7599499276> in <module>()
----> 1 'a' < 3

TypeError: '<' not supported between instances of 'str' and 'int'
```

In [48]:

```
'3' > 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-48-2c90c6ecfb90> in <module>()
----> 1 '3' > 3

TypeError: '>' not supported between instances of 'str' and 'int'
```

overloading

- 더하기(+) 기호는 연산되는 변수의 타입에 따라 다르게 작동한다.
  - 3 + 4 : 덧셈
  - "ab" + "cd" : 문자열 연결

In [49]:

```
3 + 4
```

Out[49]:

7

In [50]:

```
"ab" + "cd"
```

Out[50]:

```
'abcd'
```

이를 연산자 overloading라고 한다.

type checking

In [51]:

```
'a' * 'a'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-51-f770e0284bd4> in <module>()
----> 1 'a' * 'a'

TypeError: can't multiply sequence by non-int of type 'str'
```

실수로 잘못 입력된 형태의 type을 찾아준다.

확장 문자(escaping character)

문자열 중간에 따옴표 ' 기호를 넣고 싶을 때

- 잘못된 예

In [52]:

```
'This isn't flying, this is falling with style!'
```

```
  File "<ipython-input-52-712ccfb24280>", line 1
    'This isn't flying, this is falling with style!'
              ^
SyntaxError: invalid syntax
```

- 올바른 예 : backslash \를 삽입하여 문자열 내의 따옴표 사용임을 알림

In [53]:

```
'This isn₩'t flying, this is falling with style!'
```

Out[53]:

```
"This isn't flying, this is falling with style!"
```

\" : 큰 따옴표
\\ : backslash(\)

In [54]:

```
print('₩"')
```

```
"
```

In [55]:

```
print('₩₩')
```

```
₩
```

인덱스를 통한 특정 문자 접근

- 인덱스를 통하여 문자열의 특정 문자에 쉽게 접근
- 첫 번째 문자의 인덱스는 0부터 시작함 (1이 아님)

In [56]:

```
first_letter = "uncopywritable"[0]
print(first_letter)
```

```
u
```

In [57]:

```
my_word = "uncopywritable"
second_letter = my_word[1]
print(second_letter)
```

```
n
```

문자열 관련 함수와 method

- len()
  - python에 미리 만들어진 built-in 함수로 문자열이나 리스트의 길이를 구할 때 사용

In [58]:

```
my_name = "KS Lee"
print( len(my_name) )
```

```
6
```

- lower(), upper()
  - 소문자 혹은 대문자로 바꾸어주는 method
  - 문자열 뒤에 마침표(.)와 함께 사용
    - lower(), upper() 등은 생성한 문자열에 종속된 method로써 python에서 제공함

In [59]:

```
print("EGG".lower())
```

```
egg
```

In [60]:

```
my_string = "spam"
print(my_string.upper())
```

SPAM

- str()
  - 다른 데이터형을 문자열로 바꾸기

In [61]:

```
this_year = 2016
```

- this_year는 2016이라는 값을 지닌 정수형 데이터
- 이를 정수가 아닌 문자열로 바꾸고 싶을 때

In [62]:

```
this_year_string = str(this_year)
```

- this_year = 2016이고 this_year_string = "2016"으로 데이터형이 다름에 주의
- 다음의 값이 무엇이 나올지 살펴 보자.

In [63]:

```
print(this_year + this_year)
```

4032

In [64]:

```
print(this_year_string + this_year_string)
```

20162016

- lower()와 upper() 등의 method는 dot(.)을 이용하며 string에 종속된 method
- str(), len() 등은 dot(.)을 이용하지 않으며 괄호 안의 인자에 여러 형태의 데이터형이 올 수 있음

%s를 이용한 문자열 출력

In [65]:

```
name = "Tom"
home = "New York"
print("My name is %s and I am from %s." %(name, home))
```

My name is Tom and I am from New York.

문자열 입력

Python 3.x 에서는 input 함수를 이용하여 문자열을 입력받는다.
input()의 괄호 안에는 문자열 입력 전 화면에 보여주고 싶은 문장을 입력한다.

In [67]:

```
name = input('Enter your name: ')
```

```
Enter your name: Kyungsub Lee
```

In [68]:

```
print('Are you really ' + name + '?')
```

```
Are you really Kyungsub Lee?
```

입력 받은 문자열 type 변경

만약 입력 받은 문자열을 숫자로 활용하고자 하면 int()나 float() 함수를 이용하여 type을 변경

In [70]:

```
num = input('Input any number: ')
```

```
Input any number: 1.1
```

- num은 문자열이지만 float(num)을 통해서 실수로, 혹은 int(num)을 통해서 정수로 바꿀 수 있다.

In [71]:

```
sqr = float(num)*float(num)
print('The square of num ' + num + ' is ' + str(sqr))
```

```
The square of num 1.1 is 1.2100000000000002
```