

‘단계별 실습 과제’로 배우는 파이썬 프로그래밍 학습자료

2017.10.

- Python 2.7과 3.6의 차이(기초 함수)

e-koreatech.ac.kr



- Python3 버전은 출력할 문자열에 괄호를 필요로 함.

Python 2.7	Python 3.6
<pre>print "Hello Python" print str(err) print "Welcome to", "Python"</pre>	<pre>print ("Hello Python") print (str(err)) print ("Welcome to", "Python")</pre>

- Python3 버전은 출력 인자로 구분자(sep), 끝라인(end), 출력(file)을 지정할 수 있음.

Python 2.7	Python 3.6
<pre>print "Welcome to ~ Python!" print "Welcome to", "~", "Python!"</pre>	<pre>print ("Welcome to ~ Python!") print ("Welcome to", "~", "Python!") print ("Welcome to", "Python", sep="~", end="!")</pre>
실행결과	Welcome to ~ Python!

Python 2.7	Python 3.6
<pre>data1="Hello" data2="Python" print data1,"", data2 print data1+"",data2</pre>	<pre>data1="Hello" data2="Python" print (data1,"", data2) print (data1,data2, sep=",")</pre>
실행결과	<pre>Hello , Python Hello,Python</pre>



- `print` 문 실행 시 항상 문자열 마지막에 `\n` 문자가 출력되어 줄바꿈이 일어나게 된다.
- 이를 방지하는 방식이 2.7과 3.6 버전이 서로 다르다.

▣ 여러줄의 명령어를 한 줄에 사용하기 위해서는 명령줄 끝에 `;`을 추가하면 한 줄로 표현할 수 있다.

Python 2.7		Python 3.6	
문자열의 끝에 콤마(,)를 덧붙임.		끝 문자를 지정할 수 있는 <code>end</code> 파라미터를 설정함.	
<pre>print "No new line", print "ok?" 또는 print "No new line",;print "ok?"</pre>		<pre>print ("No new line", end=" ") print ("ok?") 또는 print ("No new line", end=" ");print ("ok?")</pre>	
실행결과		No new line ok?	



- Long 자료형이 없어지고 int로 단일화

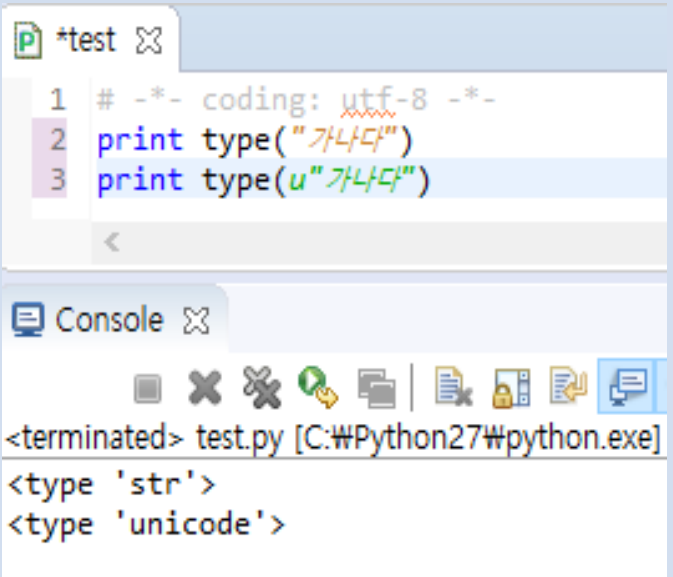
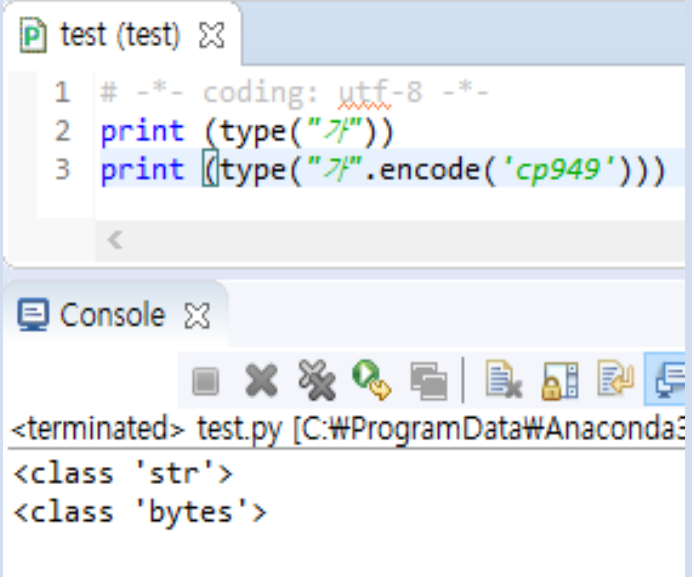
Python 2.7	Python 3.6
<pre>test.py 1 import sys 2 print type(2**31) 3 print "2**40=", 2**40, ":", type(2**40) 4 print "sys.maxint : ", sys.maxint</pre> <pre>Console <terminated> test.py [C:\Python27\python.exe] <type 'long'> 2**40= 1099511627776 : <type 'long'> sys.maxint : 2147483647</pre>	<pre>test (test).py 1 print (type(2**31)) 2 print ("2**40=", 2**40, ":", type(2**40))</pre> <pre>Console <terminated> test.py [C:\ProgramData\Anaconda3\python.exe] <class 'int'> 2**40= 1099511627776 : <class 'int'></pre>

- 자동 형 변환(연산결과에 따라 형이 자동변환 됨.)

Python 2.7	Python 3.6
<pre>test.py 1 print "3/2=", 3/2 2 print type(3/2)</pre> <pre>Console <terminated> test.py [C:\Python27\python.exe] 3/2= 1 <type 'int'></pre>	<pre>test (test).py 1 print ("3/2=", 3/2) 2 print (type(3/2))</pre> <pre>Console <terminated> test.py [C:\ProgramData\Anaconda3\python.exe] 3/2= 1.5 <class 'float'></pre>



- String => String, Unicode => Bytes 체계 변경

Python 2.7	Python 3.6
 <pre>*test 1 # -*- coding: utf-8 -*- 2 print type("가나다") 3 print type(u"가나다") Console <terminated> test.py [C:\Python27\python.exe] <type 'str'> <type 'unicode'></pre>	 <pre>test (test) 1 # -*- coding: utf-8 -*- 2 print (type("가")) 3 print (type("가".encode('cp949')))) Console <terminated> test.py [C:\ProgramData\Anaconda3\python.exe] <class 'str'> <class 'bytes'></pre>
일반스트링이 인코딩이 있는 문자열이고, 유니코드가 따로 존재함.	일반스트링이 유니코드와 동일하고, 인코딩이 있는 문자열은 bytes로 표현됨.

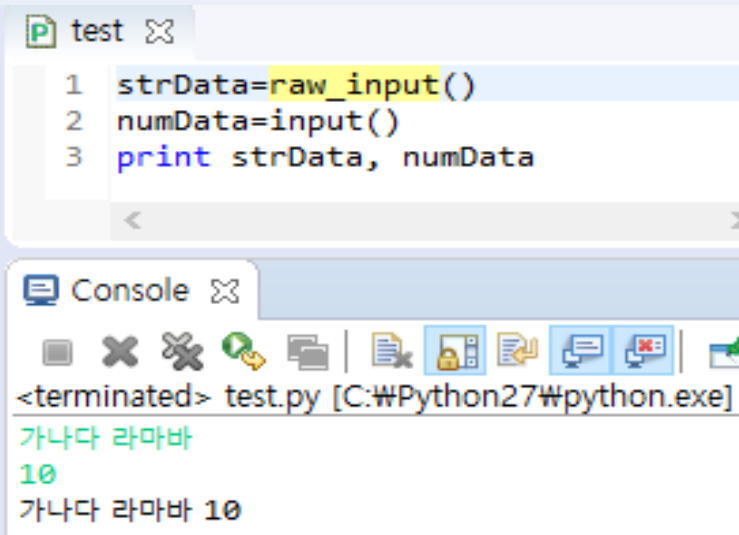
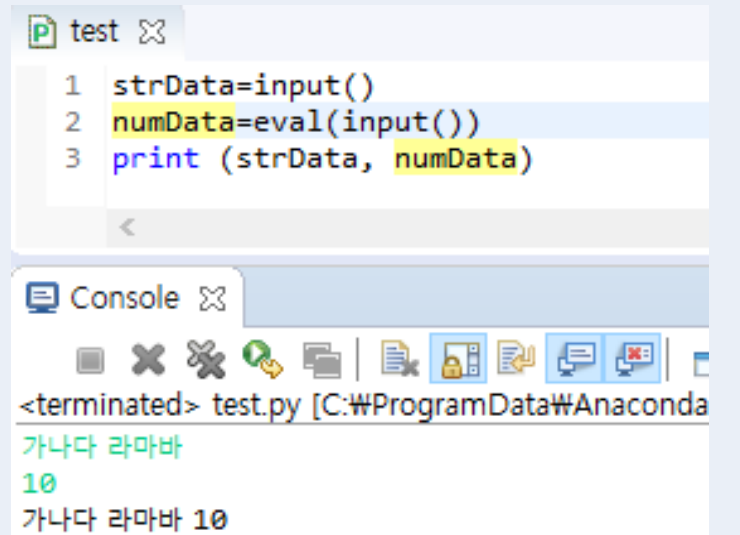
Python 2.7 버전에서 정수 값의 상한선을 읽어오는 `sys.maxint`를 실행시키려면 `sys` 패키지를 `import`해 주어야 함.

위의 Python2.7 예제에서 `import sys` 명령줄은 `sys.maxint`를 실행시키기 위해 `sys` 패키지를 해당 프로젝트로 읽어오기 위한 명령줄임.



Input/raw_input 함수

- 파이썬 2.7버전에는 문자열을 입력받기 위한 `raw input()` 함수와 숫자를 입력받기 위한 `input()` 함수가 따로 존재. (`eval` 함수 : 문자열 숫자를 순수한 수치데이터로 변환)
- 파이썬 3버전부터는 `input()` 함수가 문자열과 숫자 입력처리를 지원함.

Python 2.7	Python 3.6
문자열 입력 : <code>raw_input()</code> 숫자 입력 : <code>input()</code>	문자열 입력 : <code>input()</code> 숫자 입력 : <code>eval(input())</code>
 <pre>test.py 1 strData=raw_input() 2 numData=input() 3 print strData, numData Console <terminated> test.py [C:\Python27\python.exe] 가나다 라마바 10 가나다 라마바 10</pre>	 <pre>test.py 1 strData=input() 2 numData=eval(input()) 3 print (strData, numData) Console <terminated> test.py [C:\ProgramData\Anaconda 가나다 라마바 10 가나다 라마바 10</pre>

File() built-in 함수

- 2.7 버전 : `si = file(self.stdin, 'r')`
- 3.6버전 : `si = open(self.stdin, 'r')`



소스코드 인코딩

- 파이썬 3 버전부터는 utf-8이 기본 소스코드 인코딩이므로 다음과 같은 문자열을 소스코드 첫줄에서 생략할 수 있다.
- # -*- coding: utf-8 -*-
- 하지만 utf-8 이 아닌 다른 형태의 소스코드 인코딩을 사용해야 할 경우에는 해당 인코딩을 명시해야 한다. 하지만 파이썬 2.7 버전은 무조건 위와 같은 문자열을 소스코드 첫 줄에 명시해야만 인코딩 오류가 발생하지 않는다.

에러처리

try ... except... 에러 처리 시 에러 변수명을 표기하는 방식이 파이썬 2버전은 ,(컴마)를, 3버전은 as를 사용한다.

Python 2.7	Python 3.6
<pre>test 1 try: 2 print 4/0 3 except ZeroDivisionError , e: 4 print e</pre>	<pre>test (test) 1 try: 2 print (4/0) 3 except ZeroDivisionError as e: 4 print (e)</pre>
<pre><terminated> test.py [C:\Python27\python.exe] integer division or modulo by zero</pre>	<pre><terminated> test.py [C:\ProgramData\Anaconda3\python.exe] division by zero</pre>



[파이썬 2.7의 xrange와 range 특징]

- **xrange(start, stop, step)**

: 정해진 숫자 만큼 xrange object를 생성, 제너레이터와 같은 방식으로 동작하여 next()를 호출할 때 마다 다음 순서의 값을 반환함. 비교적 큰 범위의 값을 사용할 때는 xrange를 사용하는 것이 좋음.

- **range(start, stop, step)**

: 등간격의 정수 리스트 자체가 필요할 사용하고 시퀀셜 자료를 인덱싱할 숫자들을 만들어낼 때도 쓸 모가 있으며, 매번 list를 생성함으로써 overhead가 발생하여 메모리 낭비를 초래할 수 있음.

xrange는 필요한 숫자만 생성함으로써 메모리 낭비를 줄일 수 있고 데이터가 많을 경우 성능 차이는 크게 발생함.

Python 2.7	Python 3.6
xrange와 range가 있음	xrange가 없어졌음.
<pre>test.py 1 for x in range(10): 2 print x, 3 print 4 for x in xrange(10): 5 print x, 6 print 7 for x in xrange(3,10): 8 print x, 9 print 10 for x in xrange(3,10,2): 11 print x,</pre> <p><terminated> test.py [C:\Python27\python.exe]</p> <p>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 3 4 5 6 7 8 9 3 5 7 9</p>	<pre>test.py 1 for x in list(range(10)): 2 print (x,end=" ") 3 print ("") 4 for x in range(10): 5 print (x,end=" ") 6 print ("") 7 for x in range(3,10): 8 print (x,end=" ") 9 print ("") 10 for x in range(3,10,2): 11 print (x,end=" ")</pre> <p><terminated> test.py [C:\ProgramData\Anaconda3\python.exe]</p> <p>0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 3 4 5 6 7 8 9 3 5 7 9</p>



문자열 분리 함수 : split

- 문자열을 특정 규칙으로 분리해서 리스트로 만들어줌. 2.7과 3.6의 사용법 모두 동일.

Python 2.7	Python 3.6
<pre>test 1 strData="010-444-1234" 2 Delimiter="-" 3 listStr=strData.split(Delimiter) 4 print listStr</pre> <pre>Console <terminated> test.py [C:\Python27#python.exe] ['010', '444', '1234']</pre>	<pre>test 1 strData="010-444-1234" 2 Delimiter="-" 3 listStr=strData.split(Delimiter) 4 print (listStr)</pre> <pre>Console <terminated> test.py [C:\ProgramData\Anaconda#python.exe] ['010', '444', '1234']</pre>

문자열 연결 함수 : join

- 리스트를 원하는 문자열로 구분지어 합쳐줌. 2.7과 3.6의 사용법 모두 동일.

Python 2.7	Python 3.6
<pre>test 1 listStr=["010", "444", "1234"] 2 Delimiter="-" 3 print Delimiter.join(listStr)</pre> <pre><terminated> test.py [C:\Python27#python.exe] 010-444-1234</pre>	<pre>test 1 listStr=["010", "444", "1234"] 2 Delimiter="-" 3 print (Delimiter.join(listStr))</pre> <pre><terminated> test.py [C:\ProgramData\Anaconda#python.exe] 010-444-1234</pre>