Python Programming

Interface, Basic Concept & String Type

송지수

Python Interface

Overview

✦사용자 인터페이스 (User Interface)

- 사용자가 컴퓨터를 조작하기 위한 환경

✦GUI (Graphical User Interface)

- 시각적으로 사용자와 컴퓨터 사이의 중개하는 환경
- 예) 윈도우즈의 바탕화면

✦CUI (Character User Interface)

- 화면의 문자를 통해 사용자와 컴퓨터 사이의 중개하는 환경
- 키보드로 명령을 내리고 화면의 창을 통해 결과를 확인
- 예) 윈도우의 명령 프롬프트, 리눅스의 쉘

Python Interface

Shell Prompt

✦쉘 (Shell)

- 컴퓨터와 사용자간의 CUI 인터페이스
- 사용자의 명령을 해석하여 이 명령을 수행함
  - 예) 리눅스의 Bash, 윈도우의 명령 프롬프트
- 파이썬은 인터프리터 쉘을 통해 명령을 입력하면 주어진 명령을 수행하고 결과를 즉시 보여줌

✦파이썬의 인터프리터 쉘 프롬프트 (>>>)

- 대화식 모드로 명령을 입력 받을 준비가 되어 있음을 의미함
- 사용자는 쉘 상에서 프로그램 명령을 한 줄씩 입력하고 실행하여, 결과를 한 행씩 테스트 해볼 수 있음
- 파이썬은 인터프리터 쉘을 통해 한 줄씩 실행도 가능하며, 에디터를 통해 여러 줄의 전체 프로그램 작성도 가능함
  - 프로그램은 한 줄씩 실행하며 마지막 줄이 실행되고 나면 프로그램은 종료함

Practice

Example 3-1, 3-2

```
>>> hello
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    hello
NameError: name 'hello' is not defined
>>> "hello"
'hello'
>>>  "hello"

SyntaxError: unexpected indent
```

Practice

Example 3-1, 3-2

```
>>> import calendar
>>> calendar.setfirstweekday(6)
>>> calendar.prmonth(2023, 9)
   September 2023
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```

Python Editor

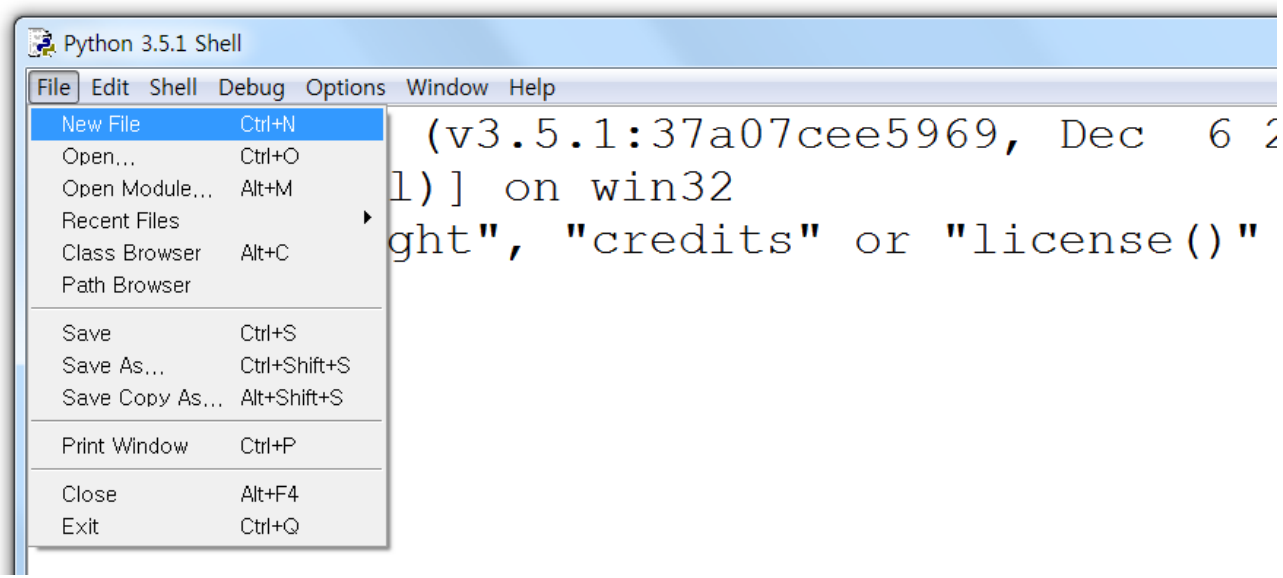
Script File

✦대화형 인터프리터의 경우 작성한 프로그램은 종료함과 동시에 사라짐

✦많은 라인의 코드를 작성하여 프로그램이 커질 경우 파일로 만들어서 수행함

- 파이썬 쉘 : 메뉴 – File – New File
- 파일 저장 : test.py (저장 경로 : C:₩Work₩Python)

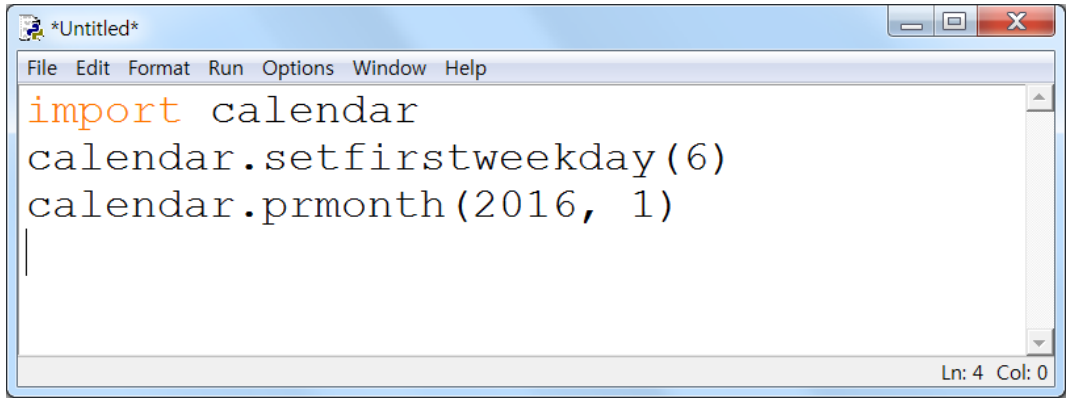
스크립트 파일로 실행
: 텍스트 파일에 프로그램을 작성하고
이것을 한꺼번에 일괄적으로 실행



Python Editor

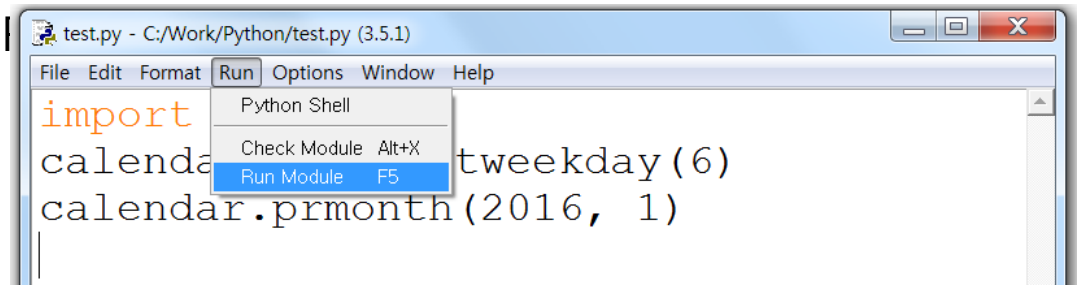
Script File

✦파이썬 코드의 작성



✦파이썬 코드의 실행

- 실행 : 메뉴 – R

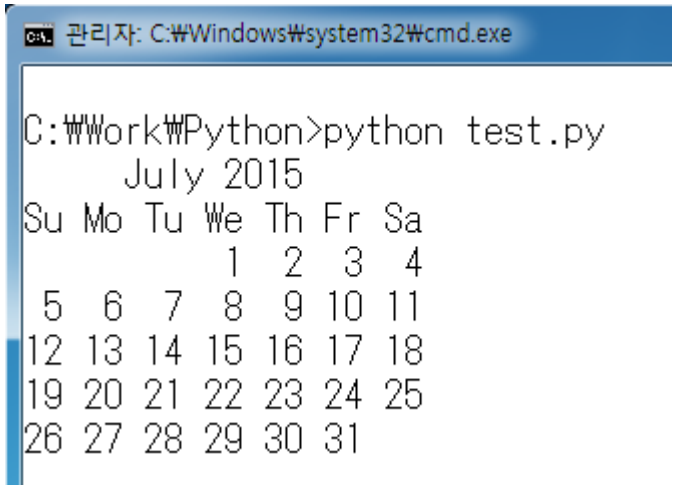


Python Editor

Script File

✦윈도우 도스 창을 이용하여 파이썬 프로그램 실행

- 프로그램이 저장된 폴더 명에서 Shift + 오른쪽 마우스 클릭
- 여기서 명령 창 열기
  - 반드시 관리자 권한으로 창이 열려야 함



Python Basic

String

✦문자열은 문자들의 모임으로 문자를 나열한 것을 말함

✦문자열 표현 방법

- ' ' 또는 " "로 문자들을 묶어서 표현하며 시작과 끝의 부호가 같아야 함
- 문자열 표현에 ' 와 " 두 가지 모두가 사용되는 이유
  - 문자열 내에 ' 또는 " 를 포함시킬 필요가 있을 때 구분하기 위함

✦문자열에서의 '+' 연산은 문자열 연결을 의미함(상수식)

- 문자열과 숫자는 서로 '+' 연산을 수행할 수 없음

```
>>> "Hello World"
'Hello World'
>>> "Hello" + "World"
'HelloWorld'
>>> "He's my father"
"He's my father"
>>> insa = "Hi Everybody"
>>> insa
'Hi Everybody'
```

공백이 포함되지 않음

' 문자 포함 문자열

```
>>> "Hello" + "100"
'Hello100'
>>> "Hello" + 100
```

에러

문자열과 정수간의 연결

```
>>> number = 100
>>> insa = "Hello"
>>> insa + number
```

에러

Python Basic

Operation

✦간단한 수식 계산

- 인터프리터 쉘을 통해 간단한 수식 계산이 가능함
- 한 라인에 여러 구문이 올 경우에는 세미콜론 ';'을 사용함

첫 칸은 공백 없이 입력되어야 함

연산자	의미
+	더하기
–	빼기
*	곱하기
/	나누기
%	나눈 나머지
//	나눈 정수 몫
**	지수 승

```
>>> 5+2
7
>>> 5-2
3
>>> 5*2
10
>>> 5/2
2.5
>>> "5+2"
'5+2'
>>> 1+1; 2+2
2
4
```

```
>>> (2+3)*(4-6)
-10
>>> 2+3*4-6
>>> 2/3
0.6666666666666666
>>> 2//3
0
```

나눈 정수 몫을 결과로 가져옴

+ 연산은 숫자일 경우 덧셈이지만 문자열인 경우 문자열의 연결

Python Basic

Function

✦프로그램에서 자주 사용되는 특정 기능을 따로 함수로 만들어 정의함

- 주로 반복적으로 수행되는 코드 블록을 의미함

✦함수는 파이썬에서 미리 정의된 함수를 사용하거나, 사용자가 직접 함수를 정의하여 사용할 수 있음

✦함수의 사용법

수학에서 사용하는 개념과 비슷함

Function(argument)

Function : 함수이름

Argument : 함수에 입력하는 값


```
>>> pow(2, 3)  거듭제곱 : 2의 3승
8
>>> divmod(10, 3)
(3, 1)  10 나누기 3의 몫과 나머지를 동시에 구함
>>> x, y = divmod(10, 3)
>>> x       두 개의 변수에 각각 대입
3
>>> y
1
```

Python Basic

Function

✦help() 함수

- 인수에 해당하는 대상의 설명을 표시하는 내장 함수
- 대상은 함수, 객체, 클래스, 모듈, 패키지 등이 올 수 있음

```
>>> help(pow)
Help on built-in function pow in module builtins:

pow(...)
    pow(x, y[, z]) -> number

    With two arguments, equivalent to x**y.  With three arguments,
    equivalent to (x**y) % z, but may be more efficient (e.g. for ints).

>>> help(divmod)
Help on built-in function divmod in module builtins:

divmod(...)
    divmod(x, y) -> (div, mod)

    Return the tuple ((x-x%y)/y, x%y).  Invariant: div*y + mod == x.
```

Python Basic

Function

✦내장 함수

- 파이썬에서 기본적으로 제공되는 기본 함수 리스트
- 모듈을 읽어 들이지 않더라도 바로 사용할 수 있는 함수

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'Blocki
ngIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError
', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'Conne
ctionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentErro
r', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPoint
Error', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarni
ng', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError',
'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'NameError', 'Non
e', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'O
verflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupErr
or', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'Sto
pIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabEr
ror', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeEr
ror', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWar
ning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionErro
r', '_', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__',
'__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'boo
l', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex'
, 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval
', 'exec', 'exit', 'filter', 'float', 'format', 'frozenset', 'getattr', 'globals
', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubc
lass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', '
min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit'
, 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'st
aticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```

Python Basic

Variable

✦일반적인 프로그래밍에서의 변수

- 숫자나 문자와 같은 데이터를 기억장소에 보관해둘 필요가 있을 때 메모리 공간을 할당하여 이름을 붙인 것
- 일반적으로 변수는 데이터의 종류에 따라 사용 전에 미리 선언함
- 변수는 하나의 데이터를 보관하며, 이름을 통해 값을 읽거나 변경함

✦파이썬에서의 변수

- 프로그래머가 특별히 메모리 공간을 관리하지 않음
- 메모리 공간은 자동으로 할당되므로 변수를 선언하지 않음
- 변수는 실제 데이터(객체)를 참조하는 참조 변수의 개념
- '=' 대입 연산자를 이용하여 변수에 값을 대입함

```
>>> x = 100
>>> y = 200
>>> z = x + y
>>> z
300
```

Python Basic

Variable

✦type()
type()은 입력받은 객체의 자료형이 무엇인지 알려주는 함수.

✦sys.getrefcount()
객체에 참조 개수(명칭)가 몇 개 있는지 알려주는 함수.

✦id()
파이썬에서 메모리에 할당된 객체의 주소를 알려주는 함수.

Practice

Example 3-6

```
>>> number1 = 100
>>> number2 = 200
>>> hap = number1 + number2
>>> hap
300
>>> insa = "Hello"
>>> nation = "korea"
>>> word = insa + nation
>>> word
'Hellokorea'
>>> word = insa + number1
Traceback (most recent call last):
  File "<pyshell#41>", line 1, in <module>
    word = insa + number1
TypeError: Can't convert 'int' object to str implicitly
```

Python Basic

dir( )

✦하나의 객체에 어떤 메서드가 존재하는지 확인

✦dir() 함수는 인자에 해당하는 대상의 다양한 리스트를 표시함

- dir() : 현재 포함되어 있는 요소들을 구함
- dir(객체) : 객체가 가진 속성, 메소드들을 구함
- dir(클래스) : 클래스가 가진 속성들을 구함
- dir(모듈) : 모듈이 포함하고 있는 요소들의 이름들을 구함
- dir(패키지) : 패키지가 포함하고 있는 요소들의 이름을 구함

✦데이터를 포함하는 함수

Python Basic

Method

```
>>> a = 100
>>> dir(a)
['__abs__', '__add__', '__and__', '__bool__', '__ceil__', '__class__', '__delatt
r__', '__dir__', '__divmod__', '__doc__', '__eq__', '__float__', '__floor__', '_
_floordiv__', '__format__', '__ge__', '__getattribute__', '__getnewargs__', '__g
t__', '__hash__', '__index__', '__init__', '__int__', '__invert__', '__le__', '_
_lshift__', '__lt__', '__mod__', '__mul__', '__ne__', '__neg__', '__new__', '__o
r__', '__pos__', '__pow__', '__radd__', '__rand__', '__rdivmod__', '__reduce__',
 '__reduce_ex__', '__repr__', '__rfloordiv__', '__rlshift__', '__rmod__', '__rmu
l__', '__ror__', '__round__', '__rpow__', '__rrshift__', '__rshift__', '__rsub__
', '__rtruediv__', '__rxor__', '__setattr__', '__sizeof__', '__str__', '__sub__'
, '__subclasshook__', '__truediv__', '__trunc__', '__xor__', 'bit_length', 'conj
ugate', 'denominator', 'from_bytes', 'imag', 'numerator', 'real', 'to_bytes']

>>> str = "Hello"
>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '_
_eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs
__', '__gt__', '__hash__', '__init__', '__iter__', '__le__', '__len__', '__lt__'
, '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__r
epr__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subcl
asshook__', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', '
expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isd
ecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'issp
ace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'pa
rtition', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip
', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate
', 'upper', 'zfill']
```

Python Basic

String Output

✦print() 함수

- 표준 출력 화면에 문자열을 출력하는 방법
- 형식 : print("문자열1", "문자열", ...)
  - 여러 개의 인수를 ,(콤마)로 구분하여 사용할 수 있음
  - 출력 문자열 사이에 공백이 추가됨
- 이 함수는 줄 바꿈이 포함된 문자열을 출력함
- 코드의 문장이 길어질 경우 '₩'(역슬래시)를 통해 여러 라인으로 작성 가능

```
>>> print(100)
100
>>> print(3.14)
3.14
>>> print(a)
Traceback (most recent call
  File "<pyshell#17>", line
    print(a)
NameError: name 'a' is not
```

a는 변수가 아님

```
>>> print("Hello World")
Hello World
>>> print("Hello" + "World")
HelloWorld
>>> print("Hello", "World")
Hello World
>>> print("Hello" * 5)
HelloHelloHelloHelloHello
>>> print("Hello" + \
      "World")
HelloWorld
```

비교

공백 추가

문자열의 반복 연산

다음 줄이 하나로 이어짐

Python Basic

String Output

✦문자로 표현하기 힘든 특수 문자 표현

- ₩n : 줄 바꿈 문자를 표현
- ₩t : Tab 문자를 표현
- ₩₩ : ₩문자를 표현
- ₩' : ' 문자를 표현
- ₩" : " 문자를 표현

```
>>> print("Hello\nWorld\tKorea")
Hello
World   Korea
```

Practice

Example 3-8

```
>>> print(100); print("Hello")
100
Hello
>>> print("Hello" + "Python")
HelloPython
>>> print("Hello", "Python")
Hello Python
>>> print("Hello", "Welcome to Python", \
      "World")
Hello Welcome to Python World
>>> print("Hello" * 5)
HelloHelloHelloHelloHello
>>> number = 100; insa = "Hello"
>>> print(number, insa)
100 Hello
>>> print(number + insa)
Traceback (most recent call last):
  File "<pyshell#68>", line 1, in <module>
    print(number + insa)
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> print(a)
Traceback (most recent call last):
  File "<pyshell#69>", line 1, in <module>
    print(a)
NameError: name 'a' is not defined
```

Python Basic

String Output

✦만약 문자열 끝에 줄 바꿈을 포함하고 싶지 않은 경우

- print함수의 인수 end를 이용하여 마지막 출력 문자를 변경

✦만약 문자열 사이에 공백이 아닌 다른 문자로 구분하고 싶은 경우

- print함수의 인수 sep를 이용하여 구분 문자를 변경

print("문자열1", "문자열2", … , end="문자")
- 출력하는 문자열 마지막 끝에 줄 바꿈 대신 지정한 문자를 삽입

print("문자열1", "문자열2", … , sep="문자")
- 출력하는 문자열 각 항목간의 출력 문자를 지정한 문자로 변경

※ 참고 : print()를 통해 파일로 출력
text = open('out.txt', 'w')
print("hello", "world", file=text)
text = close()

```
>>> print("Hello"); print("World")
Hello
World
>>> print("Hello", end=" "); print("World")
Hello World
>>> print("Hello", "World", "Korea", sep=":")
Hello:World:Korea
```

문자열 끝에 공백 문자 삽입

각 문자열 사이를 콜른(:) 으로 구분

Practice

Example 3-9

```
>>> print("Hello", "python", "World")
Hello python World
>>> print("Hello", "python", "World", sep='#')
Hello#python#World
>>> print("Hello", "python", "World", sep='123')
Hello123python123World
>>> print("Hello"); print("Python")
Hello
Python
>>> print("Hello", end='!'); print("Python", end='?')
Hello!Python?
>>> print("Hello", end=''); print("Python", end='')
HelloPython
```

Python Basic

Comment

✦주석

- 주석은 프로그래머를 위한 것으로 프로그램 소스에 설명문을 넣을 때 사용
- 프로그램 문서화의 한 부분으로 사용
- 프로그램 수행에 영향을 미치지 않음
- '#' 으로 시작하는 문장은 '#' 부터 시작해서 그 줄 끝까지 주석 처리

```
>>> #5+3
```

✦여러 줄의 주석

- xxx.py 파이썬 파일에서 여러 라인을 주석 처리할 때 사용
- ''' ''' (삼중 따옴표) 또는 """ """ 로 시작 위치와 끝 위치를 지정
- 실제 주석은 아니지만 문자열에 대한 변수 명도 없으므로 아무 일도 하지 않는 문자열로 정의하여 주석처럼 사용

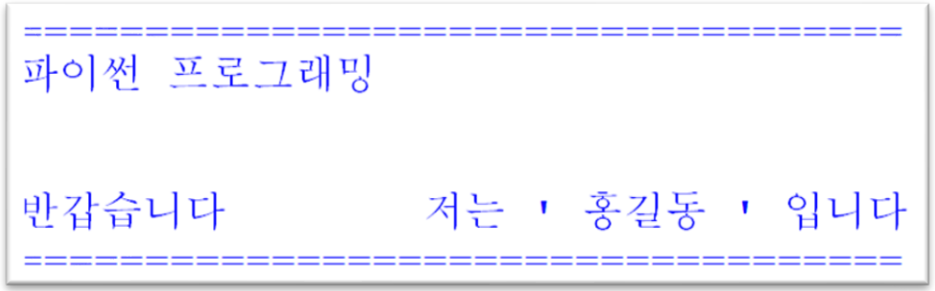
```
>>> # print("이 문장은 출력되지 않습니다")
>>> '''파이썬은 누구나 쉽게 코딩할 수 있는 프로그래밍 언어입니다
이제 부터 파이썬의 세계로 빠져볼까요'''
'파이썬은 누구나 쉽게 코딩할 수 있는 프로그래밍 언어입니다\n이제
부터 파이썬의 세계로 빠져볼까요'
>>>
```

Practice

Example 3-10

✦간단한 제목과 인사말 출력하기 (파이썬 파일로 코드 작성)

```
print("=" * 35)
#print("Python Programming")
print("파이썬 프로그래밍₩n₩n")
name = "홍길동"
print("반갑습니다₩t저는 ₩'", name, "₩' 입니다")
print("=" * 35)
```



Practice

Example 3-11

✦3명이 식사를 하고 나온 금액이 14500원이 나왔다. 팁으로 5%를 더 주고 각자 똑같이 얼마를 나타내야 하는가?

- 총 금액(cost) = 14500
- 팁(tip) = 0.05
- 비용(pay) = (cost + (cost * tip)) / 3

```
cost = 14500
tip = 0.05
pay = (cost + (cost * tip)) / 3
print("각자 내는 금액 :", pay, "원")
```

```
각자 내는 금액 : 5075.0 원
```

Python Basic

String Format

✦print() 함수를 이용한 출력 서식 지정

- 문자열의 출력을 좀더 세련되게 하기 위해 출력 서식을 지정함
- 문자열 내에 % 표시를 통해 문자열을 구성

출력서식	설명
%d %o %x	정수(10진수, 8진수, 16진수)의 표현 서식
%c	문자 한글자의 표현 서식
%f	소수점이 사용된 실수의 표현 서식
%s	한 글자 이상으로 표현된 문자열의 표현 서식

- 예 : (%.2f : 소수점 이하의 두 자리 수를 지정할 경우)

```
>>> age = 25
>>> print("당신의 나이는 %d입니다" %age)
당신의 나이는 25입니다

>>> print("%d : %f" %(10, 3.14159))
10 : 3.141590
```

두 개의 값을 차례로 서식으로 출력할 때

Python Basic

String Format

✦사용 예

- print("My age is %d" %25)
  - My age is 25
- print("My weight is %fkg" %65.3)
  - My weight is 65.300000kg
- print("My weight is %6.2fkg" %65.3)
  - My weight is 65.30kg
- print("My name is %s, age is %d" %("홍길동", 25))
  - My name is 홍길동, age is 25
- name = " 홍길동 " ; age = 25
- print(" My name is %s, age is %d " %(name, age))
  - My name is 홍길동, age is 25

Python Basic

String Format

✦format() 메서드를 이용한 출력 서식 지정

- 하나의 값을 주어진 출력 서식의 문자열로 변환
- 주로 데이터 한 개에 대한 서식 지정을 위해 많이 사용
- 예 : print(format(3.14159, ".2f"))
  - 두 번째 인수의 .2f에서 %를 사용하지 않음

```
>>> person = "{0:5s}의 나이는 {1:4d}".format("홍길동", 25)
>>> print(person)
홍길동  의 나이는   25
```

{0}와 {1}은 메서드의 첫 번째와 두 번째 인수를 각각 의미함

Practice

Example 3-12

✦200km의 거리를 시간당 80km의 속도로 운전할 때 걸리는 시간 구하기

```
distance = 200
speed = 80
time = distance / speed
print("%dkm의 거리를 시간당 %dkm로 운전할 때 걸리는 시간은 %d시간 입니다" %(distance, speed, time))
```

```
200km의 거리를 시간당 80km로 운전할 때 걸리는 시간은 2시간 입니다
```

Python Basic

String Input

✦input() 함수를 이용한 표준 입력

- 표준 입력(키보드)으로부터 값을 입력 받는 방법
- 형식 : variable = input("문자열")
- 함수의 인수는 입력 받기 전 표시할 문자열을 의미함
- 입력 받은 문자열은 변수에 저장하여 프로그램에서 사용 가능
- 파이썬 v2에서는 raw_input("문자열")을 사용함

✦주의 사항

- 입력 받은 데이터는 문자열로 인식됨
- 입력 받은 데이터가 정수나 실수와 같은 숫자라면 계산에 활용할 수 없음
- 숫자 입력의 경우 정수나 실수로 데이터를 형 변환 하여야 함

```
>>> name = input("이름을 입력하세요 : ")
이름을 입력하세요 : 홍길동
>>> print("당신의 이름은", name, "입니다")
당신의 이름은 홍길동 입니다
```

Python Basic

Type Casting

✦데이터의 형 변환

- 데이터는 정수, 실수, 문자열 등의 다양한 형식이 존재하며, 서로 다른 데이터 형끼리 연산을 수행할 경우 형 변환을 필요로 함

✦print() 함수를 수행할 때 유의해야 할 형 변환

- print()에서 숫자 형과 문자열을 '+'로 연결할 경우 오류 발생
- 숫자 형 데이터를 문자열로 형 변환을 수행함
- str() 함수 : 숫자 형 혹은 다른 형태의 데이터 형을 문자열로 형 변환

```
>>> print("Hello" + 2015)
Traceback (most recent call last):
  File "<pyshell#38>", line 1, in <module>
    print("Hello" + 2015)
TypeError: Can't convert 'int' object to str implicitly
>>> print("Hello" + str(2015))
Hello2015
```

Python Basic

Type Casting

✦input() 함수를 수행할 때 유의해야 할 형 변환

- input()을 통해 입력된 데이터는 문자열 형태임
- 입력된 숫자 문자열을 숫자 형으로 사용하기 위해서는 형 변환이 필요
- int() 함수 : 숫자로 구성된 문자열 데이터를 정수 형 숫자로 변환
- float() 함수 : 숫자로 구성된 문자열 데이터를 실수 형 숫자로 변환

```
>>> print("100" + "3.14")
1003.14
>>> print(int("100") + float("3.14"))
103.14

>>> number = int(input("수를 입력 : "))
수를 입력 : 10
>>> print(number+3)
13
```

Practice

Example 3-13

```
>>> name = input("이름을 입력하세요 : ")
이름을 입력하세요 : 홍길동
>>> print("당신의 이름은", name, "입니다")
당신의 이름은 홍길동 입니다
>>>
>>> age = input("나이를 입력하세요 : ")
나이를 입력하세요 : 25
>>> print("당신의 나이는", age, "입니다")
당신의 나이는 25 입니다
>>>
>>> birth = input("태어난 년도를 입력하세요 : ")
태어난 년도를 입력하세요 : 1993
>>> age = 2016 - birth
Traceback (most recent call last):
  File "<pyshell#35>", line 1, in <module>
    age = 2016 - birth
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

Practice

Example 3-15

✦가로 세로 길이를 입력 받아 사각형의 면적 구하기

```
width = input("가로 길이 : ")
height = input("세로 길이 : ")
area = int(width) * int(height)
print("직사각형의 넓이 : " + str(area))
```

```
가로 길이 : 10
세로 길이 : 15
직사각형의 넓이 : 150
```

Practice

Example 3-16

✦3가지 잔돈 금액을 입력 받아 잔돈의 합 구하기

```
fivehundred = int(input("500원 개수 : "))
hundred = int(input("100원 개수 : "))
fifty = int(input("50원 개수 : "))
money = 500 * fivehundred + 100 * hundred + 50 * fifty
print("잔돈 합 : " + str(money) + "원")
```

```
500원 개수 : 3
100원 개수 : 4
50원 개수 : 2
잔돈 합 : 2000원
```

Practice

Example 3-17

✦화씨 온도를 입력하여 섭씨 온도로 변환하기

✦변환 수식 : C = 5 / 9 * (F – 32)

```
fah = float(input("화씨 온도를 입력 : "))
cel = 5 / 9 * (fah - 32)
print("화씨 온도 %.1f의 섭씨 온도는 %.1f입니다" %(fah, cel))
```

```
화씨 온도를 입력 : 83.2
화씨 온도 83.2의 섭씨 온도는 28.4입니다
```

Practice

Example 3-18

✦국어, 영어, 수학 점수를 입력 받아 총점과 평균 구하기

```
kor = input("국어 점수 : ")
eng = input("영어 점수 : ")
math = input("수학 점수 : ")

total = int(kor) + int(eng) + int(math)
average = total / 3

print("총합 : %d" %total)
print("평균 : %.2f" %average)
```

```
국어 점수 : 90
영어 점수 : 85
수학 점수 : 88
총합 : 263
평균 : 87.67
```

String

Overview

✦문자열은 문자들의 모임으로 문자를 순차적으로 나열하는 데이터 형

✦‘ ’ 또는 “ ”로 문자들을 감싸서 표현함

- "Welcome to Python", 'Welcome to Python'

✦숫자도 인용 부호로 감싸면 문자열이 되며

- "20170801"

✦‘ 와 “ 두 가지를 사용하는 이유

- 문자열 내의 ‘ 또는 “ 를 구분하기 위함
- "He's my father"

✦여러 줄의 문자열을 표현할 경우 ‘’’ ’’’ 또는 “”” “”” 을 사용함

- 여러 줄의 주석을 처리할 경우 유용함
- '''파이썬은 누구나 쉽게 배울 수 있는 프로그래밍 언어입니다. 이제부터 파이선의 세계로 빠져볼까요?'''

String

Unicode

✦파이썬에서 모든 문자열은 기본적으로 유니코드로 표현

✦유니코드(Unicode)

- http://www.unicode.org/
- 각 나라별 언어를 모두 표현하기 위해 나온 코드 체계
- 모든 문자를 2byte로 표현하므로 최대 65,536자 표현 가능
  - 아스키코드(ASCII) : 1byte로 표현
- 유니코드에 기반을 둔 인코딩 방식 : UTF-8, UTF-16, UTF-32
- 파이썬에서는 '₩u'를 사용하여 유니코드 문자를 표현

✦문자 표현과 관련한 문자열 함수와 메서드

- ord("문자") → 문자의 코드 값을 반환
- chr(코드값) → 문자 코드에 대한 문자를 반환
- 문자열.encode("인코딩방식") → 문자열을 지정된 바이트 열로 인코딩
- 바이트.decode("인코딩방식") → 바이트를 문자열로 변환

Practice

Example 7-1

```
>>> import sys
>>> sys.stdin.encoding
'cp949'
>>> sys.stdout.encoding
'cp949'
>>> ord("A")
65
>>> chr(65)
'A'
>>> hex(ord("파"))
'0xd30c'
>>> chr(0xd30c)
'파'
>>> '\ud30c'
'파'
>>> python = "파이썬"
>>> python_bytes = python.encode("utf-8")
>>> python_bytes
b'\xed\x8c\x8c\xec\x9d\xb4\xec\x8d\xac'
>>> type(python_bytes)
<class 'bytes'>
>>> python_bytes.decode("utf-8")
'파이썬'
```

String

Operator

✦문자열 연산자

- '+' 연산은 문자열 연결을 의미
- '*' 연산은 문자열 반복을 의미
- '₩' 연산은 현재 라인과 다음 라인을 연결

```
>>> print("Hello", "World")
Hello World
>>> print("Hello " + "World")
Hello World
>>> print("Hello" * 3)
HelloHelloHello
>>> print("Hello" + \
      "World")
HelloWorld
>>> str1 = "Hello World"
>>> str2 = "Hi Everybody"
>>> print(str1, str2)
Hello World Hi Everybody
>>> print("Hello\nWorld\nEverybody")
Hello
World
Everybody
```

```
>>> "korea" > "japan"
True
>>> message = "korea fighting"
>>> "f" in message
True
>>> "korea" in message
True
>>> "japan" not in message
True
```

String

Operator

✦문자열과 숫자는 '+' 연산자로 연결할 수 없음

- 숫자를 문자열로 형 변환을 수행하여 동일한 종류의 데이터 형으로 연산함

```
>>> "korea" + 2017   문자열과 숫자는 연결할 수 없음
Traceback (most recent call last):
  File "<pyshell#77>", line 1, in <module>
    "korea" + 2017
TypeError: must be str, not int
>>> "korea" + str(2017)
'korea2017'
```

※ 함수 : str(객체)
숫자 객체를 출력할 수 있는 문자열로 형 변환

✦in 키워드

✦데이터 내부에 포함된 요소의 존재를 확인

- 요소가 포함되어 있을 경우 True
- 요소가 없으면 False를 반환

✦not in 키워드

- 요소가 내부에 포함되어 있지 않음을 확인함

```
>>> message = "Hello Python"
>>> "P" in message
True
>>> "a" in message
False
>>> "Hello" in message
True
>>> "world" in message
False
```

String

Output

✦문자열의 출력

- print()를 사용하여 문자열 출력
  - print() 함수의 %d, %f, %s와 같은 서식문자를 사용하여 출력 문자열의 다양한 출력 형식을 지정할 수 있음
- 사용 예
  - name = "홍길동"; age = 25
  - print("My name is %s, age is %d" %(name, age))
  - print("Hello", "Python", "World", end='#')
    - end 인수를 사용하여 마지막 문자를 다른 문자로 변경가능
  - print("Hello", "Python", "World", sep='#')
    - sep 인수를 사용하여 출력할 문자열 사이에 다른 문자를 삽입
  - Print("%d" % (5+1))
    - 서식에 해당 값이 수식일 때 괄호로 감싸는 것을 권장

String

Formatting

✦문자열 포매팅

- 문자열 내에 어떤 값을 넣어서 새로운 문자열을 만드는 것
- 문자열의 출력 서식을 지정하여 문자열 양식을 만들 수 있음

✦서식 문자를 활용한 출력 서식 지정

- 문자열 내에 % 표시를 통해 변수 값을 삽입하여 문자열을 구성함

서식문자	의미	서식문자	의미
%d	정수형 서식	%3d	필드 폭을 3칸 확보, 오른쪽 정렬
%o	8진수 정수형 서식	%-3d	필드 폭을 3칸 확보, 왼쪽 정렬
%x	16진수 정수형 서식	%.1f	소수 1자리까지 실수 출력
%f	실수형 서식	%5.2f	5개의 공간에 소수점 2자리까지
%e	지수 E 표기식 실수형 서식	%%	%표시 출력
%c	문자 서식		
%s	문자열 서식		

String

Formatting

```
>>> print("%d : %f : %e" %(5, 3.14159, 3.14159))
5 : 3.141590 : 3.141590e+00
>>> print("%d : %f : %e" %(5, 3.14159, 314.159))
5 : 3.141590 : 3.141590e+02
>>> print("%3d : %.2f : %s" %(3, 3.14159, "Hello"))
  3 : 3.14 : Hello
>>> print("10진수 %d, 8진수 %o, 16진수 %x" %(27, 27, 27))
10진수 27, 8진수 33, 16진수 1b

>>> name = "홍길동"
>>> age = 25
>>> weight = 72.3
>>> print("당신의 이름은 %s, 나이는 %d, 몸무게는 %.1f 입니다" %(name, age, weight))
당신의 이름은 홍길동, 나이는 25, 몸무게는 72.3 입니다
```

String

Input

✦문자열의 입력

- input() 함수를 통해 입력 받은 문자열을 변수에 저장하여 사용
- 단 입력 받은 데이터는 문자열로 처리되므로 계산에 활용할 수 없음

✦데이터의 형 변환

- 입력된 데이터를 연산이 가능한 정수나 실수의 데이터로 형 변환
- int() : 문자열 형태의 숫자를 정수형으로 변환 (반올림을 하지 않음)
- float() : 문자열 형태의 숫자나 정수를 실수로 변환
- str() : 숫자 데이터를 문자열로 반환
- 사용 예
  - 변수 = int(input("문자열"))

Practice

Example 7-2

✦임의의 문자열을 입력받아 문자와 숫자의 개수 구하기

```
nchar = 0; ndigit = 0; nother = 0
message = input("문자열을 입력 : ")

for letter in message:
    if 'A' <= letter <= 'Z' or 'a' <= letter <= 'z':
        nchar += 1
    elif '0' <= letter <= '9':
        ndigit += 1
    else:
        nother += 1

print("문자 : %d개" %nchar)
print("숫자 : %d개" %ndigit)
print("기타 : %d개" %nother)
```

```
문자열을 입력 : hello1234$%^&%korea
문자 : 10개
숫자 : 4개
기타 : 5개
```

Practice

Example 7-3

✦입력된 문자열에서 공백을 구분하여 단어로 출력하기

```
message = input("문자열을 입력 : ")
word = ""

for letter in message:
    if letter == " ":
        print(word)
        word = ""
        continue
    word += letter

print(word)
```

```
문자열을 입력 : hello korea fighting
hello
korea
fighting
```

데이터 전처리
: 코드에 맞게 데이터를 수정.

Practice

Example 7-4

✦주민번호를 입력하여 “-” 문자 없애기

```
idnumber = input("주민번호를 입력 : ")
newid = ""

if "-" in idnumber:
    print('"-" 문자를 제거합니다')
    for letter in idnumber:
        if letter != "-":
            newid += letter
else:
    print("올바르게 입력")

print("ID : " + newid)
```

```
주민번호를 입력 : 701214-1192343
"-" 문자를 제거합니다
ID : 7012141192343
```

Practice

Example 7-5

✦입력된 문자열에서 특수문자를 제거하여 출력하기

```
message = input("문자열을 입력 : ")
special = '''~!@#$%^&*-+=|?/₩()[]{}<>"':;'''
new_message = ""

for letter in message:
    if letter in special:
        continue
    new_message += letter

print("특수문자를 제거한 후 : %s" %new_message)
```

```
문자열을 입력 : hello<world>1234!!korea@
특수문자를 제거한 후 : helloworld1234korea
```

String

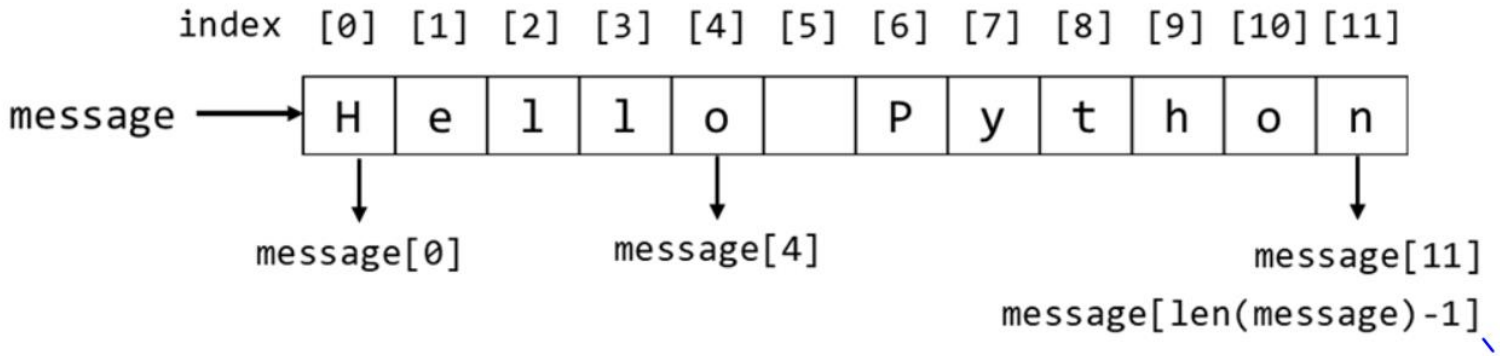
Indexing

✦문자열은 순차적인 자료형으로 인덱스를 통해 각 문자에 접근함

✦인덱스(index)

- 연속된 데이터의 요소를 구별하기 위해 사용하는 번호 (대상의 위치)
- 인덱스는 항상 0부터 시작함, 마지막 인덱스 : 전체개수-1
- 인덱스 정보를 사용하여 문자열의 문자 위치를 바로 알 수 있음
- 만약 인덱싱을 통해 문자열의 일부를 변경할 경우 에러를 발생함

```
message = "Hello Python"
```



Practice

Example 7-6

```
>>> message = "Hello Python"
>>> len(message)
12
>>> print(message[0], message[6], message[len(message)-1])
H P n
>>> message[0] = "K"
Traceback (most recent call last):
  File "<pyshell#31>", line 1, in <module>
    message[0] = "K"
TypeError: 'str' object does not support item assignment
>>> name = "홍길동"
>>> name[2]
'동'
```

각 요소의 변경은 불가능

※ 함수 : len(문자열 객체)

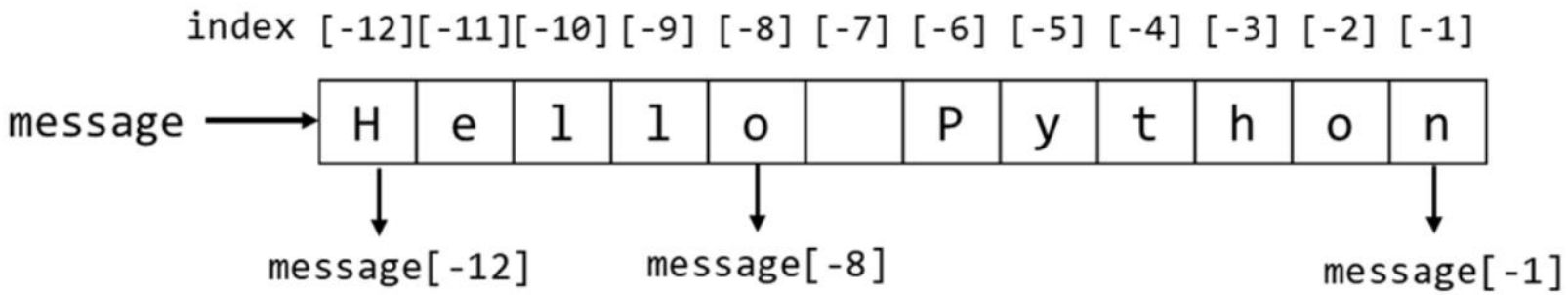
문자열 객체의 길이를 구함

String

Indexing

✦인덱스가 역방향 경우

- 문자열의 인덱스를 음수로 사용
- 문자열의 뒤에서 읽으며, 마지막 글자가 인덱스가 -1이 되어 앞으로 감소함




```
>>> message = "Hello Python"
>>> print(message[-1], message[-6])
n P
```

Practice

Example 7-7

✦문자열의 중간에 있는 문자를 출력하기

```
word = input("문자열을 입력 : ")
length = len(word)

if length % 2 == 1:
    letter = word[length // 2]
    print("중간 글자 : %s" %letter)
else:
    letter1 = word[(length // 2) - 1]
    letter2 = word[(length // 2)]
    print("중간 글자 : %s" %(letter1+letter2))
```

```
문자열을 입력 : good
중간 글자 : oo

문자열을 입력 : afternoon
중간 글자 : r
```

Practice

Example 7-8

✦입력된 문자열을 거꾸로 읽기

```
message = input("문자열 입력 : ")
length = len(message)
reverse = ""

for letter in range(0, length):
    reverse += message[length-(letter+1)]

print("반대 문자열 : " + reverse)
```

```
문자열 입력 : hello python
반대 문자열 : nohtyp olleh
```

Practice

Example 7-9

✦메뉴를 입력 받고 계속 수행 여부를 판단하는 프로그램

```
while True:
    print("1.New, 2.Load, 3.Save, 4.Exit")
    menu = int(input("메뉴 입력 : "))

    if menu == 1:
        print("New Game")
    elif menu == 2:
        print("Load Game")
    elif menu == 3:
        print("Save Game")
    elif menu == 4:
        print("종료합니다")
        answer = input("종료 하시겠습니까(y/n)? ")
        if answer.lower()[0] == "y":
            break
```

```
1.New, 2.Load, 3.Save, 4.Exit
메뉴 입력 : 4
종료합니다
종료 하시겠습니까(y/n)? yes
```

Practice

Example 7-10

✦입력된 문자열에서 특수 문자를 제거하여 출력하기

- while문과 인덱싱을 사용하여 프로그램 하기

```
message = input("문자열을 입력 : ")
special = '''~!@#$%^&*-+=|?/₩()<>"':;'''
new_message = ""

index_m = 0
while index_m < len(message):
    index_s = 0
    while index_s < len(special):
        if message[index_m] == special[index_s]:
            break
        index_s += 1
    else:
        new_message += message[index_m]

    index_m += 1

print("특수문자를 제거한 후 : %s" %new_message)
```

```
문자열을 입력 : hello<world>1234!!korea@
특수문자를 제거한 후 : helloworld1234korea
```

String

Slicing

✦슬라이싱

- 문자열에서 문자열의 일부를 분리해서 부분 문자열을 만드는 것
- 인덱스 범위 내에서 순차 자료의 일부를 추출하는 것
  - 슬라이싱의 결과에서 마지막 인덱스 요소는 포함되지 않음
- 결과는 원래의 자료형과 같으며, 문자열은 새로운 객체로 인식
- Str 객체는 = 뒤에 슬라이싱을 사용해도 새로운 객체 x

stringObject[start:end:step]

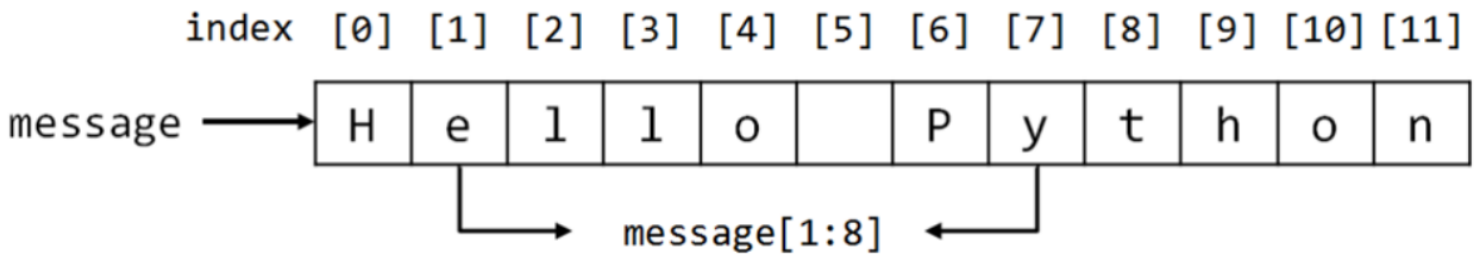
str_obj : 문자열 객체이름
start : 시작 인덱스
end : 마지막 인덱스

인덱싱과 마찬가지로 인덱스를 음수로 사용할 수 있음,
이때 문자열은 뒤에서부터 읽음

String

Slicing

☐ message = "Hello Python"



- message[0:5] → "Hello"
- message[1:8] → "ello Py"
- message[2:10:2] → "loPt"
- message[-1:-7:-1] → "nohtyP"
- message[-3:-10:-2] → "hy l"
- message = message[0:6] + "World" → "Hello World"

Practice

Example 7-11

```
>>> message = "Hello Python World"
>>> print(message[0:6], message[13:18])
Hello  World
>>> print(message[3:16:3])
lPh r
>>> print(message[-7:-13:-1])
nohtyP
>>> message[0:5] = "Korea"
Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    message[0:5] = "Korea"
TypeError: 'str' object does not support item assignment
>>> message = "Korea" + message[5:18]
>>> print(message)
Korea Python World
```

String

Slicing

✦슬라이싱의 인덱스 생략

- 슬라이싱의 시작 위치를 생략하면 처음부터 라는 의미
- 슬라이싱의 마지막 위치를 생략하면 끝까지 라는 의미

✦사용 예

- message = "Hello Python"
- message[:5] → "Hello"
- message[6:] → "Python"
- message[:] → "Hello Python"
- message[::2] → "HloPto"
- message[::-1] → "nohtyP olleH"

Practice

Example 7-12

✦입력한 주민번호에서 생년월일 추출하여 출력하기

```
import sys
idnumber = input("주민번호를 '-' 없이 입력 : ")

if "1" <= idnumber[6] <= "2":
    year = "19" + idnumber[:2] + "년"
elif "3" <= idnumber[6] <= "4":
    year = "20" + idnumber[:2] + "년"
else:
    print("잘못 입력")
    sys.exit()

month = idnumber[2:4] + "월"
day = idnumber[4:6] + "일"

print(year + month + day)
```

```
주민번호를 '-' 없이 입력 : 8912041182134
1989년12월04일
```

String Method

Function

✦문자열 관련 내장 함수

- len() : 문자열의 길이를 구함
- max(), min() : 문자열에서 가장 큰 요소와 가장 작은 요소 구하기
- 참고
  - 다음 장에 나오는 리스트, 튜플, 집합, 사전에도 적용 가능

```
>>> message = "Hello Python"
>>> len(message)
12
>>> max(message)
'y'
>>> min(message)
' '
```

String Method

Replace

✦문자열 변환하기

- 문자열.upper() : 문자열을 대문자로 바꿈
- 문자열.lower() : 문자열을 소문자로 바꿈
- 문자열.swapcase() : 문자열의 대문자와 소문자를 서로 바꿈
- 문자열.capitalize() : 문자열의 첫 문자만 대문자로 바꿈
- 문자열.title() : 문자열의 각 단어의 첫 글자만 대문자로 변환
- 문자열.expandtabs() : 문자열의 탭 문자를 공백 문자로 바꿈

```
>>> message = "hello PYTHON"
>>> message.upper()
'HELLO PYTHON'
>>> message.lower()
'hello python'
>>> message.swapcase()
'HELLO python'
>>> message.capitalize()
'Hello python'
>>> message.title()
'Hello Python'
```

String Method

Search

✦문자열 검색하기

- 문자열.count(찾을 문자열, 찾을 위치)
  - 문자열에서 인수로 지정한 문자열이 몇 번 나오는지 개수를 셈
- 문자열.replace(찾을 문자열, 바꿀 문자열)
  - 첫 번째 인수에서 지정한 문자열을 찾아서 두 번째 인수에서 지정한 문자열로 바꿈

```
>>> message = "Hi Good Python, Good Korea"
>>> message.count("Good")
2
>>> message.replace("Good", "Hello")
'Hi Hello Python, Hello Korea'
```

String Method

Search

✦문자열 검색하기

- 문자열.index(찾을 문자열, 찾을 위치)
  - 문자열에서 인수로 지정한 문자나 문자열의 처음 나타나는 인덱스를 반환
- 문자열.rindex(찾을 문자열, 찾을 위치)
  - 뒤에서 부터 검색하여 처음 나타나는 인덱스 값을 반환

```
>>> message = "Hi Good Python, Good Korea"
>>> message.index("Good")
3
>>> message.index("Good", 8)
16
>>> message.rindex("Good")
16
>>> message.rindex("Good", 20)
Traceback (most recent call last):
  File "<pyshell#127>", line 1, in <module>
    message.rindex("Good", 20)
ValueError: substring not found
```

※ 리스트 자료 형의 경우
test = [1, 2, 3, 4]
test.index(2)
결과 : 1

String Method

Search

✦문자열 검색하기

- 문자열.find(찾을 문자열, 찾을 위치)
  - 문자열에서 인수로 지정한 문자열의 위치를 앞에서 인덱스를 반환

위치는 0부터 찾기

- 문자열.rfind(찾을 문자열, 찾을 위치)
  - 뒤에서 부터 검색하여 처음 나타나는 위치의 인덱스 값을 반환

```
>>> message = "Hi Good Python, Good Korea"
>>> message.find("Good")
3
>>> message.find("Good", 8)
16
>>> message.rfind("Good")
16
>>> message.rfind("Good", 20)
-1
```

String Method

Search

✦문자열 검색하기

- 문자열.startswith(찾을 문자열, 찾을 위치)
  - 문자열에서 인수로 지정한 문자열로 시작하는지를 판단하여 부울형으로 반환
- 문자열.endswith(찾을 문자열, 찾을 위치)
  - 문자열에서 인수로 지정한 문자열로 끝나는지를 판단하여 부울형으로 반환

```
>>> message = "Hello Python"
>>> message.startswith("Hello")
True
>>> message.startswith("Hello", 1)
False
>>> message.endswith("Python")
True
>>> message.endswith("Python", 7)
False
```

String Method

Arrangement

✦문자열의 정렬과 채우기

- 문자열.center(전체 자릿수, 문자)
  - 지정된 수만큼 전체 자릿수를 잡아서 문자열을 가운데로 정렬
  - 두 번째 인수를 지정하면 지정된 문자로 공백을 채움
- 문자열.ljust(전체 자릿수, 문자) : 문자열을 왼쪽 정렬
- 문자열.rjust(전체 자릿수, 문자) : 문자열을 오른쪽 정렬
- 문자열.zfill(전체 자릿수) : 지정된 수만큼 전체 자릿수를 잡아서 오른쪽으로 붙여 쓰고 왼쪽 빈 공간을 0으로 채움

```
>>> message = "Hello Python"
>>> message.center(20)
'    Hello Python    '
>>> message.center(20, "#")
'####Hello Python####'
>>> message.ljust(20, "#")
'Hello Python########'
>>> message.rjust(20, "#")
'########Hello Python'
>>> message.zfill(20)
'00000000Hello Python'
```

String Method

Strip

✦문자열의 문자 제거

- 문자열.strip(문자)
  - 문자열의 좌우에 지정된 문자를 모두 제거
  - 인수에 아무것도 지정하지 않으면 공백을 제거함
  - 참고로 문자열 중간의 문자를 제거하기 위해서는 replace() 메서드를 활용
- 문자열.lstrip(문자) : 문자열의 왼쪽 문자를 제거
- 문자열.rstrip(문자) : 문자열의 오른쪽 문자를 제거

```
>>> message = "    Hello Python    "
>>> message
'    Hello Python    '
>>> message.strip()
'Hello Python'
>>> message = "----Hello-Python----"
>>> message.lstrip("-")
'Hello-Python----'
>>> message.rstrip("-")
'----Hello-Python'
>>> message.strip("-")
'Hello-Python'
```

String Method

Check

✦문자열의 검사

- 문자열의 구성을 분석하여 결과를 True, false로 반환
- 문자열.isalpha()
  - 문자열이 문자로만 구성되어 있는지 검사
- 문자열.isdigit()
  - 문자열이 숫자로만 구성되어 있는지 검사
- 문자열.isalnum()
  - 문자열이 문자 또는 숫자로만 구성되어 있거나 문자와 숫자가 같이 구성되어 있으면 True를 반환

```
>>> message = "Hello"
>>> message.isalpha()
True
>>> message = "Hello World"
>>> message.isalpha()
False
>>> message = "12345"
>>> message.isdigit()
True
>>> message = "12.345"
>>> message.isdigit()
False
>>> message = "Hello1234"
>>> message.isalnum()
True
>>> message = "Hello 1234!!!"
>>> message.isalnum()
False
```

String Method

Check

✦문자열의 검사

- 문자열의 구성을 분석하여 결과를 True, false로 반환
- 문자열.islower() : 문자열이 소문자로만 구성되어 있는지 검사
- 문자열.isupper() : 문자열이 대문자로만 구성되어 있는지 검사
- 문자열.isspace() : 문자열이 오직 공백으로 구성되어 있는지 검사
- 문자열.istitle() : 문자열의 첫 글자가 대문자인지 검사

```
>>> message = "hello python!!"
>>> message.islower()
True
>>> message.isspace()
False
>>> message = "Hello Python"
>>> message.istitle()
True
>>> message = "HEllo Python"
>>> message.istitle()
False
```

String Method

Split

✦문자열의 분리

- 문자열.split()
  - 문자열에서 부분문자열을 특정 구분문자로 분리하여 리스트로 반환

```
>>> message = "Hello Python Korea"
>>> message.split()
['Hello', 'Python', 'Korea']
```

- 문자열.splitlines()
  - 문자열에서 줄 바꿈이 포함된 문자열을 행 단위로 분리

```
>>> message = "Hello Python"
>>> message.splitlines()
['Hello Python']
>>> message = "Hello Python\nKorea Fighting"
>>> message.splitlines()
['Hello Python', 'Korea Fighting']
```

String Method

Join

✦문자열의 결합

- 구분문자.join(문자열 또는 리스트)
  - 인수에서 지정한 문자열이나 리스트의 각 항목들에 구분 문자를 중간에 삽입하여 하나의 문자열을 만들어 반환

```
>>> message = "Hello Python"
>>> ":".join(message)
'H:e:l:l:o: :P:y:t:h:o:n'
>>> person = ["홍길동", "25", "korea"]
>>> "-".join(person)
'홍길동-25-korea'
>>> message = "Let's go Python World"
>>> ":".join(message.split())
"Let's:go:Python:World"
```

Practice

Example 7-26

✦비행코드를 입력하여 정보 얻기 (NYC : 뉴욕, ICN : 인천, NRT : 도쿄)

- 코드 : 출발지역, 비행기번호, 도착지역을 공백 없이 붙여서 구성

```
code = input("비행 코드 입력 : ")
code = code.upper()

if code.startswith("NYC"):
    print("출발 : 뉴욕", end="")
elif code.startswith("NRT"):
    print("출발 : 도쿄", end="")
elif code.startswith("ICN"):
    print("출발 : 인천", end="")
else:
    print("코드가 없습니다")

print(" <-> ", end="")
if code.endswith("NYC"):
    print("도착 : 뉴욕")
elif code.endswith("NRT"):
    print("도착 : 도쿄")
elif code.endswith("ICN"):
    print("도착 : 인천")
else:
    print("코드가 없습니다")
```

```
비행 코드 입력 : NRTKE232ICN
출발 : 도쿄 <-> 도착 : 인천
```

Practice

Example 7-27

✦'*'로 다이아몬드 그리기

```
width = int(input("다이아몬드의 폭을 입력 : "))
star = "*"
loop = width-(width//2)

if width % 2 == 1:
    for count in range(loop):
        print(star.center(width))
        star += "**"

    for count in range(loop-1, 0, -1):
        star = "*" * (count * 2 - 1)
        print(star.center(width))
else:
    print("홀수를 입력하세요")
```

```
다이아몬드의 폭을 입력 : 11
     *
    ***
   *****
  *******
 *********
***********
 *********
  *******
   *****
    ***
     *
```

Practice

Example 7-28

✦임의의 문자열을 입력 받아 문자와 숫자의 개수 구하기

- 단 문자열 검사 메서드를 활용

```
nchar = 0; ndigit = 0; nother = 0
message = input("문자열을 입력 : ")

for letter in message:
    if letter.isalpha():
        nchar += 1
    elif letter.isdigit():
        ndigit += 1
    else:
        nother += 1

print("문자 : %d개" %nchar)
print("숫자 : %d개" %ndigit)
print("기타 : %d개" %nother)
```

```
문자열을 입력 : hello1234$%^&%korea
문자 : 10개
숫자 : 4개
기타 : 5개
```

Practice

Example 7-29

✦이름과 나이를 입력 받아 문자인지 숫자인지 검사하는 프로그램

```
while True:
    name = input("이름 입력 : ")
    if not name.isalpha():
        print("문자만 입력 가능합니다")
        continue

    age = input("나이 입력 : ")
    if not age.isdigit():
        print("숫자만 입력 가능합니다")
        continue
    else:
        age = int(age);
        break

    print("이름 : %s, 나이 : %d" %(name, age))
```

```
이름 입력 : 홍길동123
문자만 입력 가능합니다
이름 입력 : 홍길동
나이 입력 : 25a
숫자만 입력 가능합니다
이름 입력 : 홍길동
나이 입력 : 25
이름 : 홍길동, 나이 : 25
```

Practice

Example 7-30

✦비밀번호를 입력하여 문자는 '#'으로 숫자는 '*'으로 표시하여 출력하기

```
s_passwd = input("비밀번호 입력 : ")
h_passwd = s_passwd
for count in range(len(s_passwd)):
    if s_passwd[count].isdigit():
        h_passwd = h_passwd.replace(s_passwd[count], "*")
    if s_passwd[count].isalpha():
        h_passwd = h_passwd.replace(s_passwd[count], "#")

print("비밀번호 : " + s_passwd)
print("숨겨진 비밀번호 : " + h_passwd)
```

```
비밀번호 입력 : hello1234korea5678
비밀번호 : hello1234korea5678
숨겨진 비밀번호 : #####****#####****
```

Practice

Example 7-31

✦이름과 비밀번호를 체크하여 처리하는 프로그램

- 단 비밀번호는 숫자나 문자만 오면 안 됨
- 동일한 숫자나 문자는 연속으로 올 수 없음

```
name = input("이름 입력 : ")
name.strip()
name = name.replace(" ", "")

passwd = input("비밀 번호를 입력 : ")
for count in range(0, len(passwd)-1):
    if passwd[count] == passwd[count+1]:
        print("연속된 문자가 존재합니다")
        break;

else:
    if passwd.isalpha() or passwd.isdigit():
        print("문자와 숫자를 섞어야 합니다")
    else:
        print("정상처리 되었습니다")
        print("이름 : %s, 비밀번호 : %s" %(name, passwd))
```

```
이름 입력 : 홍길동
비밀 번호를 입력 : 1234
문자와 숫자를 섞어야 합니다

이름 입력 : 홍길동
비밀 번호를 입력 : hong1111
연속된 문자가 존재합니다

이름 입력 :  홍 길 동
비밀 번호를 입력 : hong1234
정상처리 되었습니다
이름 : 홍길동, 비밀번호 : hong1234
```

Practice

Example 7-32

✦점수를 차례로 입력하여 총합 구하기

```
score = input("점수를 차례로 (-)로 입력 : ")

total = 0
for item in score.split("-"):
    total += int(item)

print("총점 : %d" %total)
```

```
점수를 차례로 (-)로 입력 : 100-98-85-78
총점 : 361
```