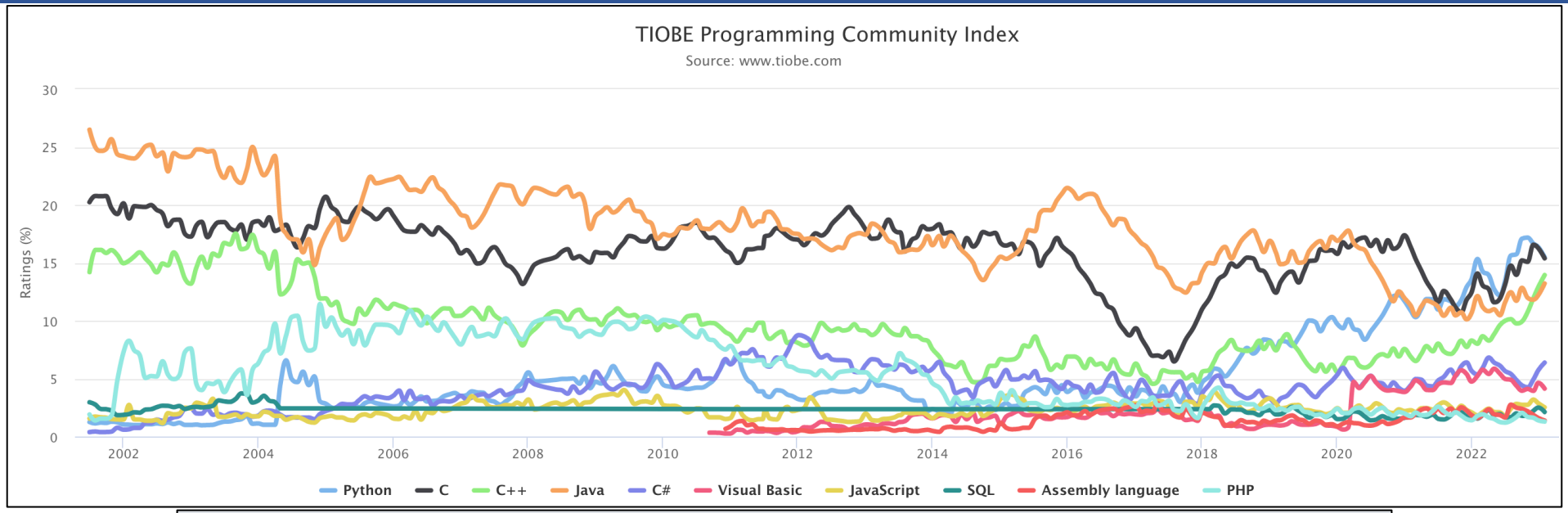
# 네트워크 프로그래밍
## - Python 기초 -

순천향대학교 사물인터넷학과

# TIOBE Programming Community Index (2023년 2월)




| Feb 2023 | Feb 2022 | Change | Programming Language | Ratings | Change |
|---|---|---|---|---|---|
| 1 | 1 | | Python | 15.49% | +0.16% |
| 2 | 2 | | C | 15.39% | +1.31% |
| 3 | 4 | ↑ | C++ | 13.94% | +5.93% |
| 4 | 3 | ↓ | Java | 13.21% | +1.07% |
| 5 | 5 | | C# | 6.38% | +1.01% |
| 6 | 6 | | Visual Basic | 4.14% | -1.09% |
| 7 | 7 | | JavaScript | 2.52% | +0.70% |
| **8** | **10** | ↑ | **SQL** | **2.12%** | **+0.58%** |
| 9 | 9 | | Assembly language | 1.38% | -0.21% |
| 10 | 8 | ↓ | PHP | 1.29% | -0.49% |

# 파이썬 프로그램의 구성 요소

- 파이썬 프로그램은 변수, 상수, 예약어, 연산자, 함수 등으로 구성
  - 예약어는 변수나 함수명으로 사용할 수 없음

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

  - 예약어 확인하기

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

# 파이썬 프로그램의 구성 요소

## ■ 식별자identifier

- 변수, 사용자 정의 함수, 클래스, 모듈 등의 이름

✓ 영문자, 숫자, 밑줄 (_), 유니코드문자를 사용할 수 있으나 숫자로 시작할 수는 없음
✓ 영문 대문자와 소문자는 구별됨 (예 : name과 Name은 다름)
✓ 예약어를 명칭으로 사용할 수는 없음
✓ 식별자의 첫 문자는 알파벳 문자 (ASCII 대/소문자 혹은 유니코드 문자) 이거나 밑줄(_) 이어야 함
✓ 한글도 유니코드 문자이므로 사용할 수는 있지만 가급적 사용하지 않는 게 좋음

## ■ 변수variable

- 변할 수 있는 값을 저장하는 공간
- 식별자 규칙에 따라 부여
- 가급적 소문자를 사용하고 의미를 알 수 있는 변수명 사용

# 파이썬 프로그램의 구성 요소

## ■ 변수의 종류

- 값을 저장하는 장소를 나타내는 이름
- 저장하는 값의 형type에 따라 분류

```
booleanVar = True          #부울형
intVar = 1                 #정수형
floatVar = 2.3             #실수형
strVar = "string"          #문자열
```

## ■ 연산자operator

- 연산을 나타내는 기호
- 산술연산자, 관계연산자, 논리연산자, 증가감소연산자, 비트연산자, 지정연산자

## ■ 상수constant

- 저장 값이 변하는 않는 장소를 나타내는 이름
- PI = 3.14, GRAVITY = 9.8
- C 언어의 const와 같은 기능은 없음

# 파이썬 프로그램의 구성 요소

## ■ 주석문comments

- # 문자부터 줄 끝까지 1줄로 이루어진 설명문
- 프로그램 실행과는 상관없이 소스 코드에 대한 설명을 위해 사용
- ''' 또는 """를 사용하여 여러 줄에 걸친 주석문 사용 가능

```
area = 1/2 * base * height    #삼각형의 면적

'''
These are multi-line
comments
'''
```

## ■ 함수

- 반복적으로 호출하여 사용 가능한 프로그램의 실행 단위
- 내장 함수, 사용자 함수: print(), myFunc()

# 기본 입출력 함수

## print() 함수

- 숫자나 문자열 등의 데이터나 수식을 화면에 출력할 때 사용

```
>>> print(10+20)                  #( ) 속의 연산 결과 출력
30
>>> print("사물인터넷")           #문자열 출력
사물인터넷
>>> print("abc " * 3)             #문자열 3번 출력
abc abc abc
>>> x = 25
>>> y = 32
>>> z = x + y
>>> print(x, y)                   #변수 x, y 값 출력
25 32
>>> print(x, '+', y, '=', z) #문자열과 변수의 조합 출력
25 + 32 = 57
```

# 기본 입출력 함수

```
# Separator 옵션사용
print('2023','03','02', sep='_')
print('daeheekim','sch.ac.kr', sep='@')
# format 사용
print('{} and {}'.format('You', 'Me'))
print("{0} and {1} and {0}".format('You', 'Me'))
print("{a} are {b}".format(a='You', b='Me'))
# %s: 문자 , %d: 정수, %f: 실수
print("%s's favorite number is %d" %('Daehee',7))
print("Test1: %5d, Price: %4.2f" %(776, 6534.123))
# %5d: 5자리의 숫자가 온다고 자릿수를 지정, %4.2f: 정수 4자리 소수 2자리 인 실수
print("Test1: {0:5d}, Price: {1:4.2f}".format(776, 6534.123))
print("Test1: {a:5d}, Price: {b:4.2f}".format(a=776, b=6534.123))
```

```
2023_03_02
daeheekim@sch.ac.kr
Welcome To the black parade piano notes
You and Me
You and Me and You
You are Me
Daehee's favorite number is 7
Test1:   776, Price: 6534.12
Test1:   776, Price: 6534.12
Test1:   776, Price: 6534.12
```

# 기본 입출력 함수

## input() 함수

- 키보드를 이용하여 숫자나 문자열 등의 데이터를 입력 받을 때 사용
- 입력된 정보는 문자열

```
>>> n = input('type a number : ')
type a number : 3
>>> n                    #문자열로 입력된다
'3'
>>> int(n)               #문자열을 정수로 변환
3
>>> float(n)             #문자열을 실수형으로 변환
3.0
>>> n = str(3)           #정수를 문자열로 변환
>>> n
'3'
```

# 기본 입출력 함수

## ■ 입출력 함수의 사용 예

- 문자열을 입력 받아 다른 문자열과 결합하여 출력

```
name = input('Name: ') ← 문자열 입력
print("Hello," + name)
```

- 두 정수를 입력 받아 곱셈 결과 출력

```
a = int(input('Number 1: ')) ← 문자열을 입력한 후 정수로 변환
b = int(input('Number 2: '))
print(a * b)
```

- 섭씨온도(C)를 입력 받아 화씨온도(F)로 바꾸어 출력

```
C = float(input("Type 섭씨온도 : ")) ← 입력 문자열을 실수로 변환
F = C*9/5 + 32          #온도 변환
print("화씨온도는", F)
```

# 자료형과 크기

## ■ 부울형(Boolean)

- 참(True) 또는 거짓(False) 중 하나의 값을 갖는 자료형(True, False)
- 비교 결과를 저장할 때 사용

```
>>> aVar = True
>>> bVar = False
>>> type(aVar)
<class 'bool'>

>>> a = 1
>>> b = 2
>>> c = a > b
>>> c
False
>>> d = a < b
>>> d
True
```

# 자료형과 크기

## ■ 정수형integer

- 소수점 없는 숫자 표현

```
>>> intVar = 5
>>> type(intVar)
<class 'int'>
```

- 여러 가지 정수 표현

```
1, 23, 3493             # 10진수
0b010101, 0b110010      # 2진수
0o1, 0o27, 0o6645       # 8진수
0x1, 0x17, 0xDA5        # 16진수
```

# 자료형과 크기

## ■ 실수형 float

```
>>> fVar = 1.0
>>> type(fVar)
<class 'float'>

# 여러 가지 실수 표현
0., 0.0, .0, 1., 1.0, 1e0, 1.e0, 1.0e0
```

## ■ 쉽게 식별할 수 있도록 _를 사용한 숫자의 표현

- 파이썬 3.6 부터 적용

```
>>> 100_000.000_0001, 0x_FF_FF, 0o7_777, 0b_1010_1010
(100000.0000001, 65535, 4095, 170)
>>> 0.00_10_27
0.001027
```

# 자료형과 크기

## ■ 문자열string

- 문자의 나열
- ' ' 또는 " "로 둘러 쌈

```
>>> strVar = "test"
>>> type(strVar)
<class 'str'>

'This is a literal string'
"This is another string"

"I'm a Python fanatic"          #"" 속에 작은 따옴표를 사용할 경우
'I"m a Python fanatic'          #'' 속에 큰 따옴표를 사용할 경우

'A not very long string \       #문장을 두 줄 이상으로 표시할 경우
that spans two lines'
```

# 연산자의 종류

## ■ 지정 연산자assignment operator

```
i = 3
i = i + 2
```

- 변수의 값을 지정하기 위해 사용(=)

| 연산자 | 설명 | 사용 예 |
|---|---|---|
| x + y | x와 y를 더함 | c = a + b |
| x - y | x에서 y를 뺌 | s = a - b |
| x * y | x와 y를 곱함 | m = a * b |
| x / y | x를 y로 나눔 | d = a / b |
| x // y | x를 y로 나눔(정수 나눗셈) | d = a // b |
| x % y | x를 y로 나눈 나머지 | r = a % b |
| x ** y | x의 y 제곱 | r = 2 ** 5 |

## ■ 산술 연산자arithmetic operator

- 사칙연산자: +, -, *, /
- 정수나눗셈: //
- 나머지 연산: %

```
a = 10
b = 3
c = a % b       ⇨ 1
d = a // b      ⇨ 3
e = a / b       ⇨ 3.3333333333333335
```

# 연산자의 종류

## ■ 관계 연산자relation operator

- 비교연산자
- 항의 대소관계 또는 동등관계 판정
- 연산결과는 True(참) 또는 False(거짓)

```
score = 95
print(score >= 90) ⇨ True
print(score < 80) ⇨ False
print(score == 90) ⇨ False
print(score != 80) ⇨ True
```

| 연산자 | 설명 | 사용 예 |
|---|---|---|
| x < y | x가 y보다 작으면 True | x=1; y=2; print(x<y) |
| x <= y | x가 y보다 작거나 같으면 True | x=1; y=2; print(x<=y) |
| x > y | x가 y보다 크면 True | x=1; y=2; print(x>y) |
| x >= y | x가 y보다 크거나 같으면 True | x=1; y=2; print(x>=y) |
| x == y | x와 y가 같으면 True | x=1; y=2; print(x==y) |
| x != y | x와 y가 다르면 True | x=1; y=2; print(x!=y) |

# 연산자의 종류

## ■ 논리 연산자logical operator

- 논리곱(and), 논리합(or), 논리부정(not)
- 논리 연산의 대상은 True 또는 False. 결과도 True 또는 False

```
>>> exam = 85
>>> report = 85
>>> exam >= 80 and report >= 80
True
>>> exam >= 90 or report >= 90
False
>>> not(exam < 90)
False
```

| 연산자 | 설명 | 사용 예 |
|---|---|---|
| x and y | x와 y가 모두 참일때만 연산결과가 참 | a=5; print(a > 0 and a < 10) |
| x or y | x와 y 둘 중 하나라도 참이면 연산 결과가 참 | a=5; print(a < -3 or a > 3) |
| not x | x의 부정. x가 참(거짓)이면 결과는 거짓(참). | a=False; print(not a) |

# 연산자의 종류

## ■ 멤버 연산자membership operator

- in
- 특정 문자열 또는 값이 문자열 또는 리스트에 속해 있는지 판별하는 연산자

```
>>> 'py' in 'python'
True
>>> 'ty' not in 'python'
True
```

## ■ 식별 연산자identity operator

- is : 양쪽 변수의 주소가 같은지 판단. if id(a) == id(b)
- is not : 양쪽 변수의 주소가 다른지 판단. if id(a) != id(b)

```
>>> a = 2
>>> b = 2
>>> id(a)
1921220752
>>> id(b)
1921220752
```

```
>>> id(a) == id(b)
True
>>> a is b
True
>>> a is not b
False
```

# 연산 오류

## ■ 타입 에러type error

- 연산할 수 없는 자료형을 연산하면 타입 에러 발생

```
age = 23
message = "Happy " + age + "rd Birthday!"
print(message)

Traceback (most recent call last):
File "birthday.py", line 2, in <module> ❶
message = "Happy " + age + "rd Birthday!"
TypeError: can only concatenate str (not "int") to str ❷

❶ 에러 발생 위치
❷ 에러 내용: str과 str만 연결할 수 있음
```

# 조건문

## ■ if문

- 조건을 만족할 때 정해진 문장을 실행

```
if Condition(조건) :
    Block(블록)
```

- 조건이 참(True)이면 블록을 실행하고 아니면 블록을 건너뜀

```
score = int(input("정수 입력:"))
if score >= 90 :                #끝에 :을 붙인다
    print("성적 : A")           #들여쓰기
    print("장학금 수여")

age_0 = 22
age_1 = 18
age = int(input('your age? '))
if age < age_0 and age > age_1: #복합 조건문
    print("당신의 나이는 18~22세 사이입니다")
```

# 조건문

## if-else 문

```
if Condition(조건) :
    Block1(블록1)
else :
    Block2(블록2)
```

- 조건이 참(True)이면 Block1을 실행하고 else: 는 무시
- 참(True)이 아니면 else: 다음 Block2를 실행

```
if score >= 90 :
    print("pass")
else :                  #끝에 :을 붙인다
    print("fail")       #들여쓰기
```

- 한 줄로 표현하기

```
res = 'pass' if score >= 90 else 'fail'
print(res)
```

# 조건문

- 평년과 윤년 구분하는 프로그램
  - 년도가 4로 나뉘어지며 100으로 나뉘어지지 않거나
  - 400으로 나뉘어지면 윤년임

```
year = int(input("Type a year :"))
if (year%4 == 0 and year%100 != 0) or (year%400) == 0 :
    print(year, "is leap year")
else :
    print(year, "is not a leap year")
```

# 조건문

■ 중첩 if 문

```
n = int(input("type int number : "))
if n > 0 :
    print("positive")
else :
    if n < 0 :
        print("negative")
    else :
        print("0")
```

```
n = int(input("type int number : "))
if n >= 0 :
    if n == 0 :
        print("0")
    else :
        print("positive")
else :
    print("negative")
```

# 조건문

## if-elif-else 문

- if 조건1이 참이면 블록1, if 조건1이 거짓이고 elif 조건2이 참이면 블록2, 조건이 모두 거짓이면 블록3이 실행

```
if Condition1(조건1) :
    Block1(블록1)
elif Condition2(조건2) :
    Block2(블록2)
else :
    Block3(블록3)
```

```
n = int(input("type int number : "))
if n > 0 :
    print("positive")
elif n < 0 :
    print("negative")
else :
    print("0")
```

# 조건문

## ■ 다중 elif 문

- if 또는 elif가 참이면 블록을 실행하고 나머지 elif나 else는 무시
- 다음 프로그램의 결과는 $4

```
age = 12
if age < 4:
    price = 0
elif age < 18:
    price = 4
elif age < 65:
    price = 10
else:
    price = 5

print("Your admission cost is $" + str(price) + ".")
```

# 반복문

## while 문

```
[초기화]
while Condition(조건) :
    Block(블록)
[다음 문장]
```

- 조건이 만족되면 블록을 실행하고 다시 조건을 조사
- 조건이 만족되지 않으면 다음 문장 실행

```
i = 1
sum = 0
while i <= 100 :          #끝에 :를 붙인다
    sum = sum + i         #들여쓰기
    i = i + 1

print("sum=", sum)        #블록이 끝나면 다시 들여쓰기를 원래 위치로
```

# while 연습문제

- 10진수를 입력 받아 2진수로 변환하여 출력하는 프로그램

```
n = int(input('Number: '))
result = ''

while n != 0:
    m = n % 2
    result = str(m) + result
    n = n // 2

print(result)
```

Number: 14
1110

# 반복문

## for 문

- 리스트 이용 시

```
for 변수 in [리스트] :
    블록
```

- [리스트]의 각 요소의 값을 하나씩 [변수]에 저장하고 블록을 실행
- [리스트]에 더 이상의 요소가 없을 때까지 반복

```
for i in [1, 2, 3] :  #끝에 :를 붙인다
    print("숫자", i)  #들여쓰기
```

- [리스트] 위치에 리스트 뿐만 아니라 튜플, 문자열과 같은 반복 객체를 사용할 수 있음

```
for i in (1, 2, 3):   # 튜플
    print("숫자", i)

for i in 'hello':     # 문자열
    print(i)
```

# 반복문

## for-range 문

```
for 변수 in range(start, stop, step) :
    블록
```

- 어떤 조건에 대해 연산을 반복할 때 사용
- range() 함수는 반복 객체를 생성하는 함수
- 변수 값을 start부터 시작하여 step 만큼 증가시키며 stop-1까지 실행
  step=1이면 생략

```
for i in range(1, 4, 1) : #1부터 시작하여 1씩 증가하여 4가 되면 종료
    print("Number", i)    #들여쓰기

Number 1
Number 2
Number 3
```

# 반복문

- for 문을 이용한 1부터 100까지의 합 구하기

```
sum = 0
for i in range(1, 101, 1) :
    sum = sum + i

print("sum = ", sum)
```

- $x^n$을 구하는 프로그램

```
x = float(input('Type x : '))
n = int(input('Type n : '))

prod = 1
for i in range(1, n+1):
    prod = prod * x

print(prod)
```

# 반복문

## ■ 중첩 반복문

- 내부 for 문이 먼저 실행되고 끝나면 외부 for 문이 실행됨
- 구구단 프로그램

```
for i in range(2, 10, 1) :
    for j in range(1, 10, 1):
        print("%d x %d = %d" %(i, j, i*j))
```

## ■ 무한 반복문

```
while True :
    print("last forever")
```

- 무한 반복문을 중지하려면 [Ctrl]+C를 누름

# 기타 제어문

## break 문

- 반복문을 빠져나올 때 사용

```
n = 10
while n >= -10 :
    if n == 0 :
        break           #n=0이면 while 블록을 빠져나온다
    inv = 1.0 / n
    print(inv)
    n = n - 1
```

## continue 문

- 블록의 나머지 부분은 생략하고 다시 while 조건 조사

```
n = 10
while n >= -10 :
    if n == 0 :
        n = n - 1
        continue        #n=0이면 while 블록의 나머지 부분을 건너뜀
    inv = 1.0 / n
    print("n = %d, inv(n) = %0.2f"%(n, inv))
    n = n-1
```

# 제어문 예제

## ■ 숫자 맞추기 게임 프로그램

- 1~100 사이의 난수를 만들고 사용자가 1~100 사이의 수를 입력하여 맞추는 프로그램
- 5회까지 맞추지 못하면 종료

```
from random import randint
secret_num = randint(1,100)
num_guesses = 0          #시도 횟수
guess = 0                #예상 숫자

while guess != secret_num and num_guesses <= 4:
    guess = eval(input('Enter your guess (1-100): '))
    num_guesses = num_guesses + 1
    if guess < secret_num:
        print('더 큽니다.', 5-num_guesses, '회 남았습니다.\n')
    elif guess > secret_num:
        print('더 작습니다.', 5-num_guesses, '회 남았습니다.\n')
    else:
        print('맞았습니다!’)

if num_guesses==5 and guess != secret_num:
    print('당신이 졌습니다. 정답은 ', secret_num, '입니다')
```

# 파이썬 자료 구조

- 자료 구조data structure란?
  - 프로그램에서 자료를 조직, 관리, 저장하는 구조
  - 자료를 효율적으로 처리하고 저장하는 데 사용

- 파이썬에서 사용되는 기본 자료 구조
  - 문자열
  - 리스트
  - 튜플
  - 사전(딕셔너리)
  - 집합

# 문자열

## ■ 문자열의 구조

- 다수의 문자를 처리하기 위한 자료구조

**문자열 기본 구조**

문자열이름 = "글씨들" (또는 '글씨들')

(예) word = "Python" (word = 'Python')

---

문자열은 큰따옴표 또는 작은따옴표로 구분된다.

- ' ' 또는 " "로 둘러쌈: "abc", 'hello'
- '''(""") 부터 '''(""")까지 문자: 문자열이 두 줄 이상일 때 주로 사용

```
'''
Hello
Python
'''
```

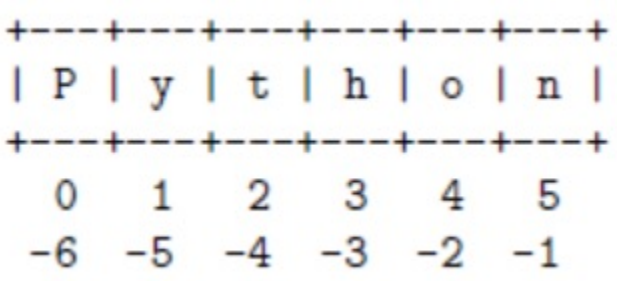
```
"""
Hello
Python
"""
```

# 문자열

- 문자열 요소(문자)를 개별적으로 사용하기: 인덱스 사용

```
>>> word = 'Python'
>>> word[0]    # 위치(첨자) 0의 문자. 인덱스는 0부터 시작
'P'
>>> word[5]    # 인덱스 5의 문자
'n'
```

```
>>> word = 'Python'
>>> word[-1]   # 마지막 문자. 마지막은 -1로 표시
'n'
>>> word[-6]   # 뒤에서 6번째 문자
'P'
```



# 문자열

## ■ 문자열 슬라이싱(잘라내기)

```
>>> word = 'Python'
>>> word[0:2]  # 첨자 0부터 1(=2-1)까지
'Py'
>>> word[2:5]  # 첨자 2부터 4(=5-1)까지
'tho'
>>> word[:2]   # 첨자 0부터 1까지의 문자
'Py'
>>> word[4:]   # 첨자 4부터 끝까지
'on'
>>> word[-2:]  # -1에서 -2까지 문자열
'on'
>>> word[2:]   # 2부터 끝까지
'thon'
>>> word[:2]+word[2:]          # 문자열 합치기
'Python'
```

# 문자열

## ■ 문자열 인덱싱과 슬라이싱 오류

```
>>> word[42]  # the word only has 6 characters
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

```
>>> word[4:42]          # ❶ 슬라이싱에서는 범위를 벗어나도 OK
'on'
>>> word[42:]           # ❷ 범위를 벗어난 슬라이싱
''
```

## ■ 문자열은 수정 불가

```
>>> word[0] = 'J'
...
TypeError: 'str' object does not support item assignment
```

# 문자열

- 슬라이싱을 이용하여 새로운 문자열 만들기

```
>>> word = 'Python'
>>> word2 = 'J' + word[1:]
>>> word2
'Jython'
```

- 문자열 길이 확인: `len()`

```
>>> s = 'smiles'
>>> len(s)
6
```

# 문자열 처리

- 인덱스를 이용하여 문자열의 문자를 역순으로 만들기

```
## 변수 초기화 부분 ##
outStr = ""    # 변수를 초기화(빈 문자열)
count, i = 0, 0

## 메인 코드 부분 ##
inStr = input("Type string : ")
count = len(inStr)    # 문자열 길이

for i in range(0, count) :
    outStr += inStr[count - (i + 1)] # 마지막 문자부터 추출하여 저장
print("Reversed string : %s" % outStr)
```

- 문자열에서 문자 조사하기: in/not in 연산자

```
>>> 'a' in 'cat'
True
>>> 'a' not in 'can'
False
```

# 문자열 처리

- 문자열과 메소드: dir(str), help(str.method이름)
  - 파이썬은 객체지향 언어이므로 객체object가 많이 사용됨
  - 객체는 변수와 함수를 묶은 것으로, 문자열도 객체
  - 객체와 관련된 함수를 메소드라 하며 객체.함수()로 호출한다
- 문자열 구성 파악하기

```
'123'.isdigit()                 # 문자열이 숫자인가?
'abcABC'.isalpha()              # 문자열이 문자인가?
'Ab122'.isalnum()               # 문자열이 문자+숫자 혼합인가?
'AB'.isupper()                  # 문자열이 대문자인가?
'ab'.islower()                  # 문자열이 소문자인가?
' '.isspace()                   # 문자열이 공백인가?
```

- 문자열을 대소문자로 변환하기

```
str = 'Python programming is easy!'
str.upper()                     # 대문자로 변환
str.lower()                     # 소문자로 변환
str.swapcase()                  # 대소문자 바꾸기
str.title()                     # 첫 글자만 대문자로 변환
```

# 문자열 처리

## ■ 문자열 찾기

```
>>> str = 'Python programming is easy!'
>>> str.count('i')    # 문자열에서 'i'의 개수
2
>>> str.find('o')     # 문자열에서 'o'의 처음 위치. 없으면 -1 반환
4
>>> str.rfind('o')    # 문자열에서 'o'가 나오는 가장 나중 위치
9
>>> str.index('on')   # 문자열에서 'on'의 처음 위치. 없으면 ValueError
4
```

## ■ 문자열 분리하기

```
>>> x = 'a bc'          # a와 b 사이에 공백이 있음
>>> x.split()           # 공백을 기준으로 분리
['a', 'bc']
>>> x.split('b')        # 문자열을 ( )속의 문자로 분리
['a ', 'c']             # 분리문자 'b'는 제외됨
```

# 문자열 처리

## ■ 문자열에서 공백 제거하기

```
>>> str = ' hello '
>>> str.strip()          # 양쪽에서 공백 제거
'hello'
>>> str                  # 원래 문자열은 변하지 않는다
' hello '
>>> str.rstrip()         # 오른쪽 공백 제거
' hello'
>>> str.lstrip()         # 왼쪽 공백 제거
'hello '
```

## ■ 문자열 결합하기

```
>>> '*'.join('hello')    # *와 문자열의 각 문자를 하나씩 결합한 문자열 생성
'h*e*l*l*o'
```

## ■ 문자열 채우기

```
>>> '12'.zfill(5)        # 문자열 '12'의 왼쪽을 0으로 채워 5자리로 만든다
'00012'
```

# 연습문제

- 문자열 “My name is 본인이름”에 대해 다음 물음에 답하라
  - 문자열의 문자수를 출력하라.
  - 문자열을 10번 반복한 문자열을 출력하라.
  - 문자열의 첫 번째 문자를 출력하라.
  - 문자열에서 처음 4문자를 출력하라.
  - 문자열에서 마지막 4문자를 출력하라.
  - 문자열의 문자를 거꾸로 출력하라.
  - 문자열에서 첫 번째 문자와 마지막 문자를 제거한 문자열을 출력하라.
  - 문자를 모두 대문자로 변경하여 출력하라.
  - 문자를 모두 소문자로 변경하여 출력하라.
  - 문자열에서 'a'를 'e'로 대체하여 출력하라.
    - ✓ 문자열.replace('a', 'e')

# 리스트

- 리스트는 여러 형type의 데이터를 하나의 이름으로 저장하는 자료 구조

리스트 기본 구조

```
리스트이름 = [항목1, 항목2, ...]
```

```
>>> odd = [1, 3, 5, 7, 9]     # [ ]를 이용하여 리스트 생성
>>> squares = [1, 4, 9, 16, 25]
>>> squares
[1, 4, 9, 16, 25]
>>> empty = []                 # 공백 리스트. empty = list()
>>> empty
[]
# 리스트 요소는 어떤 것이라도 가능
>>> a = [1, 'hello', 2, 'world', 1.23]
```

# 리스트

- 리스트의 인덱싱(요소 접근하기)과 슬라이싱(잘라내기)

```
>>> squares = [1, 4, 9, 16, 25]
>>> squares[0]          # 리스트 요소 추출하기(인덱싱은 0부터)
1
>>> squares[-1]         # 마지막 요소
25
>>> squares[0:3]        # 0부터(3-1)까지 잘라내기
[1, 4, 9]
>>> squares[-3:]        # 뒤에서 3번째부터 마지막까지 잘라내기
[9, 16, 25]
```

- 리스트 합치기

```
>>> squares + [36, 49, 64, 81, 100] # 리스트 합치기
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b       # 리스트 합치기
[1, 2, 3, 4, 5, 6]
```

# 리스트

## ■ 리스트의 요소 수정, 추가하기

```
# ❶ 수정하기
>>> cubes = [1, 8, 27, 65, 125]
>>> 4 ** 3                    # the cube of 4 is 64, not 65!
64
>>> cubes[3] = 64             # 리스트 요소 수정하기
>>> cubes                     #수정된 리스트
[1, 8, 27, 64, 125]

# ❷ 마지막에 요소 추가하기
>>> cubes.append(216)         # 마지막에 리스트 요소 추가하기
>>> cubes.append(7 ** 3)      # 마지막에 리스트 요소 추가하기
>>> cubes
[1, 8, 27, 64, 125, 216, 343]
```

# 리스트

## ■ 리스트의 요소 수정, 제거하기

```
>>> letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
>>> letters
['a', 'b', 'c', 'd', 'e', 'f', 'g']

>>> letters[2:5] = ['C', 'D', 'E']  # 리스트 요소 수정
>>> letters
['a', 'b', 'C', 'D', 'E', 'f', 'g']

>>> letters[2:5] = []               # 리스트 요소 제거
>>> letters
['a', 'b', 'f', 'g’]

>>> letters[:] = []                 # 리스트 요소를 모두 제거
>>> letters
[]
```

# 리스트

## ■ 중첩 리스트

```
>>> a = ['a', 'b', 'c']
>>> n = [1, 2, 3]
>>> x = [a, n]
>>> x
[['a', 'b', 'c'], [1, 2, 3]] # 리스트 요소가 리스트임
>>> x[0]
['a', 'b', 'c']
>>> x[0][1]                  # 중첩 리스트 요소 지정
'b'
```

## ■ 2차원 리스트

```
list2d = [[1,2,3],           # 행렬을 나타낼 때 편리
          [4,5,6],
          [7,8,9]]
>>> list2d[0][0]
1
>>> list2d[2][2]
9
```

# 리스트

## ■ 내장 함수를 이용한 리스트 다루기

```
# 리스트 길이 알아내기
>>> letters = ['a', 'b', 'c', 'd']
>>> len(letters)
4

# 리스트 요소 합, 최소값, 최대값 구하기
>>> num = [1, 2, 3, 4, 5]
>>> sum(num)
15
>>> min(num)
1
>>> max(num)
5
```

# 리스트 메소드

■ 리스트 메소드(list.메소드이름)

| 함수명 | 사용 방법 | 설명 |
|---|---|---|
| del() | del(List[pos]) | List에서 pos 위치의 항목 삭제 |
| len() | len(List) | List 항목의 전체 개수를 센다 |
| sorted() | newList = sorted(List) | List 항목을 정렬해 newList 만듬 |
| append() | List.append(val) | List 맨 뒤에 val 항목 추가 |
| clear() | List.clear() | List의 내용을 지움 |
| copy() | newList = List.copy() | List의 내용을 newList에 복사 |
| count() | List.count(val) | List에서 val 값의 갯수를 센다 |
| extend() | List.extend(newList) | List 뒤에 newList를 추가 |
| index() | List.index(val) | val을 찾아 해당 첨자 반환 |
| insert() | List.insert(pos, val) | List의 pos 위치에 val 삽입 |
| pop() | List.pop() | List 맨 뒤에서 항목 삭제 |
| remove() | List.remove(val) | List에서 val 값 항목 삭제(여러 개면 첫번째만) |
| reverse() | List.reverse() | List 항목을 뒤집어 역순으로 만듬 |
| sort() | List.sort() | List 항목을 정렬 |

**사용법 확인하기**

```
>>>help(list.sort)
```

# 리스트 순서

■ 리스트 요소 순서 나열 (정렬)

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort()                    # 오름차순으로 배열
print(cars)

cars.sort(reverse=True)        # 역순 배열
print(cars)

print("Here is the original list:")
print(cars)
print("\nHere is the sorted list:")
print(sorted(cars))            # 오름차순으로 배열한 리스트를 따로 생성
print("\nHere is the original list again:")
print(cars)
```

# 리스트 순서

- 리스트 역순으로 출력하기

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
cars.reverse()        #요소 순서를 거꾸로 배열
print(cars)
```

- 리스트 관련 에러

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles[3])

Traceback (most recent call last):
File "motorcycles.py", line 3, in <module>
print(motorcycles[3])
IndexError: list index out of range # 인덱스 범위 초과
```

# 튜플 tuple

- 튜플은 리스트와 비슷하나 요소 값을 변경할 수 없음!
- 튜플 만들기

```
>>> t1 = (1,2,3)        # ( ) 속에 요소를 나열하여 만든다.
>>> t1
(1, 2, 3)

>>> t2 = 1,2,3          # ( ) 없이 항목을 나열해서 만들 수도 있다.
>>> t2
(1, 2, 3)

>>> t3 = 1,             # 요소가 한 개인 경우 ,를 붙인다.
>>> t3
(1, )
>>> t4 = tuple()        # 빈 튜플 생성
```

# 튜플

## ■ 튜플 요소 접근하기(인덱싱과 슬라이싱)

```
>>> t1 = (1,2,3)
>>> t1[1]
2
>>> t1[1:3]
(2,3)
```

## ■ 튜플 연산

```
>>> t1 = (1,2,3)
>>> t2 = ('a','b')
>>> t1+t2                  # 두 개의 튜플을 하나로 합치기
(1, 2, 3,'a','b')
>>> t1*2                   # 동일한 튜플을 하나 더 생성하여 합치기
(1,2,3,1,2,3)
```

# 튜플

## ■ 튜플과 리스트 상호 변환

```
>>> T1 = (1,2,3)
>>> L1 = list(T1)      # 튜플을 리스트로 변환
>>> L1.append(4)
>>> T1 = tuple(L1)     # 리스트를 튜플로 변환
>>> T1
(1, 2, 3, 4)
```

## ■ 튜플 언패킹

```
>>> a, b, c = (1, 2, 3)  # 튜플의 요소를 하나씩 변수에 할당. 언패킹
>>> a
1
>>> b
2
>>> c
3
```

# 딕셔너리 dictionary

- 키(key):값(value)로 구성되는 자료 구조
  - 키는 정수, 실수, 문자열, 튜플
  - 값은 임의의 데이터형
- 딕셔너리 생성

```
>>> height = {'Jun':174, 'Kim':170, 'Lee':165}     # {key:value}
>>> height
{'Jun': 174, 'Kim': 170, 'Lee': 165}

>>> test = dict()     # 빈 딕셔너리 생성, test = {}
>>> test
{}
```

- 딕셔너리 키 값 조사하기

```
>>> height['Kim']     # Dict[key]로 값 조사하기
170
>>> height['Lee']
165
```

# 딕셔너리

## ■ 딕셔너리 키 조사

```
>>> height = {'Jun':174, 'Kim':170, 'Lee':165}
>>> 'Kim' in height   # in 연산자로 키가 있는지 조사
True
```

## ■ 딕셔너리 요소 수정, 추가하기

```
>>> height['Lee'] = 180      # 요소 수정하기
>>> height
{'Jun': 174, 'Kim': 170, 'Lee': 180}

>>> height['Ihm'] = 168      # 요소 추가하기
>>> height
{'Jun': 174, 'Kim': 170, 'Lee': 180, 'Ihm': 168}
```

# 딕셔너리

- 딕셔너리의 키, 값, 요소를 뽑아내기

```
>>> height.keys()     # 키 뽑아내기
dict_keys(['Jun', 'Kim', 'Lee', 'Ihm'])     # iterable이 반환됨

>>> height.values()                         # 값 뽑아내기
dict_values([174, 170, 165, 168])

>>> height.items()                          # 요소 분리하기
dict_items([('Jun', 174), ('Kim', 170), ('Lee', 165), ('Ihm',
168)])

# 뽑아낸 키를 리스트로 변환하기
>>> dict_keys = height.keys()
>>> keys_list = list(dict_keys)
>>> keys_list
['Jun', 'Kim', 'Lee', 'Ihm']
```

# 딕셔너리

## for 문 사용법

- 딕셔너리에 대해 for 문을 실행하면 key값이 할당
- value, (key, value)로 반복하기 위해서는 values(), items() 사용

```
a = {'kim': [4, 1, 17], 'lee': 170, 'park': 85, 'choi': 'IoT'}
for key in a:
    print(key)

print()
for key in a.keys():
    print(key)

print()
for val in a.values():
    print(val)

print()
for key, val in a.items():
    print('key={}, value={}'.format(key, val))
```

# 딕셔너리

## ■ 정렬하기

- sorted() 함수 사용
- key와 value에 대해서 정렬할 수 있음

```
>>> fruits = {'melon':2, 'banana':1, 'plum':0, 'pear':2, 'apple':1}
>>> sorted(fruits)
['apple', 'banana', 'melon', 'pear', 'plum']

>>> sorted(fruits.keys())
['apple', 'banana', 'melon', 'pear', 'plum']

>>> sorted(fruits, reverse=True)
['plum', 'pear', 'melon', 'banana', 'apple']

>>> fruits.items()
dict_items([('melon', 2), ('banana', 1), ('plum', 0), ('pear', 2), ('apple', 1)])

>>> sorted(fruits.items())
[('apple', 1), ('banana', 1), ('melon', 2), ('pear', 2), ('plum', 0)]

>>> sorted(fruits.items(), key=lambda t: t[1])
[('plum', 0), ('banana', 1), ('apple', 1), ('melon', 2), ('pear', 2)]
```

# 딕셔너리

## ■ 딕셔너리 요소 제거하기

```
>>> height.pop('Ihm')   #'lhm' 요소 제거
168
>>> height
{'Jun': 174, 'Kim': 170, 'Lee': 165}
```

## ■ 딕셔너리 지우기

```
>>> height.clear()       # 요소를 지우기. 빈 딕셔너리로 만든다
>>> height
{}
```

# 집합 set

- { } 속에 요소들이 중복되지 않고 순서없이 모아 놓은 자료 구조
- 집합의 생성

```
numbers = {1,2,3}
name = set(['Kim', 'Lee', 'Park'])
```

- 빈 집합 생성, 개수 확인, 요소 추가, 요소 제거하기

```
>>> empty = set()                # 빈 집합 생성
>>> numbers = {1,2,3}
>>> len(numbers)
3
>>> numbers.add(4)               # 요소 추가
>>> numbers
{1, 2, 3, 4}
>>> numbers.remove(4)            # 요소 제거
>>> numbers
{1, 2, 3}
```

# 집합

## ■ 집합의 연산

```
>>> setA={1,2,3}
>>> setB={2,3,4}
>>> setA & setB          # 교집합. setA.intersection(setB)
{2, 3}

>>> setA | setB          # 합집합 . setA.union(setB)
{1, 2, 3, 4}

>>> setA - setB          # 차집합. setA에서 setB 요소를 뺀 나머지.
                         # setA.difference(setB)
{1}
```

# 집합

## ■ 부분 집합 연산

```
>>> setA = {1, 2, 3}
>>> setB = {1, 2, 3}
>>> setA == setB                #두 개의 집합이 같은가?
True

>>> setA = {1, 2, 3, 4, 5, 6}
>>> setB = {1, 2, 3}
>>> setB < setA                 #집합 B가 집합 A의 부분집합인가?
True

>>> setB.issubset(setA)         #집합 B가 집합 A의 부분집합인가?
True
```

# 집합 메소드

| Method | Description |
|---|---|
| **Nonmutating** | |
| *S*.copy() | Returns a shallow copy of *S* (a copy whose items are the same objects as *S*'s, not copies thereof), like set(*S*) |
| *S*.difference( *S1* ) | Returns the set of all items of *S* that aren't in *S1* |
| *S*.intersection( *S1* ) | Returns the set of all items of *S* that are also in *S1* |
| *S*.issubset( *S1* ) | Returns True when all items of *S* are also in *S1*; otherwise, returns False |
| *S*.issuperset( *S1* ) | Returns True when all items of *S1* are also in *S*; otherwise, returns False (like *S1*.issubset(*S*)) |
| *S*.symmetric_difference( *S1* ) | Returns the set of all items that are in either *S* or *S1*, but not both |
| *S*.union( *S1* ) | Returns the set of all items that are in *S*, *S1*, or both |
| **Mutating** | |
| *S*.add( *x* ) | Adds *x* as an item to *S*; no effect if *x* was already an item in *S* |
| *S*.clear() | Removes all items from *S*, leaving *S* empty |
| *S*.discard( *x* ) | Removes *x* as an item of *S*; no effect when *x* was not an item of *S* |
| *S*.pop() | Removes and returns an arbitrary item of *S* |
| *S*.remove( *x* ) | Removes *x* as an item of *S*; raises a KeyError exception when *x* was not an item of *S* |

# 집합 예제

- 1부터 45까지의 수 중에서 6개를 선택하여 로또 번호를 만드는 프로그램

```
import random
pick = set()                        # 빈 집합

while len(pick) < 6:
    n = random.randint(1,45) # 랜덤 넘버 생성
    if n not in pick:
        pick.add(n)                 # 집합에 요소 추가

print(pick)
print(sorted(pick))                 # 순서대로 나열된 집합 출력
```

# 함수

- 함수란?
  - 이름을 붙여 반복적으로 사용할 수 있는 프로그램 부분
  - `print()`, `input()`, `int()`, `str()`, `sorted()` 등
- 함수를 사용하는 이유
  - 프로그램 안에서 중복된 코드를 제거
  - 복잡한 프로그래밍 작업을 더 간단한 작업들로 분해할 수 있음
  - 함수는 한번 만들어지면 다른 프로그램에서도 재사용 가능
  - 함수를 사용하면 가독성(프로그램의 이해)이 증대되고, 유지 관리가 쉬워짐
- 파이썬의 함수는 변수처럼 사용 가능
  - 변수에 할당: `a = func`
  - 다른 함수에 인자로 전달 가능: `func1(func)`
  - return 문으로 반환 가능: `return func`
- 함수의 종류
  - 내장 함수
  - 사용자 정의 함수

# 함수의 정의와 호출

## ■ 사용자 정의 함수의 작성

```
def 함수이름(매개변수) :          # 함수의 이름과 매개변수 정의
    문장1
    문장2
    ...
    return 값  ▶함수 종료시 반환 값. 들여쓰기가 끝나는 부분까지 함수
```

## ■ 함수의 정의와 호출 예

```
def welcome(name) :              # 함수의 정의
    print("Hello, ", name)      # return 문이 없어도 됨. None 반환

welcome('John')          # 함수의 호출. 인자 'John'이 매개변수로 전달됨
```

# 함수의 반환

## ■ 반환 값이 없는 함수

```
def prtStr(str) :      # str을 매개변수라 함
    print("%s" %str)
    return             # 반환 값이 없이 함수가 종료. 생략가능

str = "Wecome to Python"
prtStr(str)            # 매개변수를 전달하여 함수 호출
```

## ■ 값을 반환하는 함수

```
def squareArea(s) :
    area = s * s
    return area        # area 값이 반환되고 종료

a = squareArea(5)      # 5를 인자(argument)라 함. 반환 값이 a에 저장
b = squareArea(7)
print("한변의 길이가 %d인 정사각형의 넓이는 %d" %(5, a))
print("한변의 길이가 %d인 정사각형의 넓이는 %d" %(7, b))
```

# 함수의 예

- 1~n까지의 합을 계산하는 함수

```
def sum(n):              # 매개변수 n을 전달받는다
    hab = 0
    for i in range(1, n+1): # 1~n까지 반복
        hab = hab + i # 합에 새로운 수를 더한다
    return hab          # 반환값=sum

a = sum(50)    # 함수를 호출하여(인자=50) 1부터 50까지의 합을 계산
b = sum(1000)  # 함수를 호출하여(인자=50) 1부터 1000까지의 합
print(a)
print(b)
```

Q. 반지름을 전달하면 원의 면적을 반환하는 cir_area(r) 함수와 원의 둘레를 반환하는 cir_circum(r) 함수를 작성하라. 이들 함수를 이용하여 반지름이 3.5cm인 원의 면적과 둘레를 소수점 아래 첫 자리까지 구하라.

(`import math` 실행 후, `math.pi` 사용)

# 함수와 변수

■ 지역 변수local variable와 전역 변수global variable

- 지역 변수: 함수 내에서만 값이 유효
- 전역 변수: 프로그램 내에서 값이 유효
- 다음 프로그램에서 v는 지역 변수

```
# 함수 선언부
def func1() :
    v = 10      # 함수 func1()안의 지역 변수. func1 내에서만 유효
    print("func1()의 v = %d" % v)

def func2() :
    v = 20      # 함수 func2()안의 지역 변수. func2 내에서만 유효
    print("func2()의 v = %d" % v)

# 메인 코드 부분
func1()
func2()
```

# 전역 변수

## ■ 전역 변수

```
# 함수 선언부
def func1() :
    v = 10     # 함수 func1()안의 지역 변수. 지역 변수가 우선 사용
    print("func1()의 v = %d" % v)

def func2() :
    print("func2()의 v = %d" % v) # 함수 func2()안의 전역 변수

# 전역변수 선언부
v = 20  # 전역 변수이므로 프로그램 내에서 유효

# 메인 코드 부분
func1() # 지역 변수 값인 10을 출력. 지역 변수가 없을 때 전역 변수 사용
func2() # 전역 변수 값인 20을 출력
```

# 전역 변수

## ■ 전역 변수의 선언

```
# 함수 선언부
def func1() :
    v = 10      # 함수 func1()안의 지역 변수
    print("func1()의 v = %d" % v)

def func2() :
    global v    # v를 전역 변수로 선언
    v = 30
    print("func2()의 변경된 전역 v = %d" % v)

v = 20          # 함수 밖에서 정의 되었으므로 전역 변수

# 메인 코드 부분
func1()         # 10을 출력
func2()         # 30을 출력
print(v)        # 30을 출력. 전역변수 v가 변경됨
```

# 함수의 매개변수

- 매개변수의 기본값 지정(기본 매개변수)

```
def prtMesg(message, count=1):   # count의 기본값이 1로 지정
    print (message * count)
prtMesg('default')               # message='default'. count는 기본값 사용
prtMesg('Hi! ', 5)               # message와 count 인자 전달
```

```
def greet(name, msg="Nothing new?"):
    print("Hi! ", name + ', ' + msg)
greet("Abe") # name 만 전달
greet("Bob", "Good morning!") # name과 msg 전달
```

- 일반(위치) 매개변수의 값을 전달하지 않으면 오류

```
>>> prtMesg(count=3)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    prtMesg(count=3)
TypeError: prtMesg() missing 1 required positional argument: 'message'
```

# 함수의 매개변수

- 가변 매개변수: 매개변수의 개수가 가변일 때
  - 매개변수가 *arg로 표시되면 튜플로 처리

```
def total(*numbers):
    sum = 0
    for n in numbers:    # 전달 받은 매개변수를 튜플로 처리
        sum += n
    return sum
print (total(1,2,3))     # 3개의 인자
print (total(1,2,3,4,5)) # 5개의 인자
```

  - 매개변수가 **keywords로 표시되면 딕셔너리로 처리

```
def dicPresident(**keywords):
    for i in keywords.keys():
        print("%s : %d-th president" %(i, keywords[i]))

dicPresident(Kennedy=35, Obama=44, Trump=45) # (key=value)로 전달
```

# 함수의 매개변수

## 키워드를 이용한 매개변수 전달

- 매개변수의 위치가 아니라 키워드를 이용하여 매개변수 전달

```
def func(a, b=2, c=3): # a: 위치 매개변수, b, c: 키워드 매개변수
    print('a=', a, 'b=', b, 'c=', c)
    return a + b + c

print(func(4, 5))      # 인자 위치에 의해 a=4, b=5로 전달
print(func(5, c=7))    # 키워드 c를 이용한 전달. a=5, c=7, b는 기본값
```

## pass 키워드

- 함수 몸체의 작성을 잠시 보류할 때 사용
- 문법적 오류를 피하고 아무 것도 실행하지 않을 때 사용

```
def func(arg):
    pass
```

# 람다(lambda) 함수

## ■ 람다(무명) 함수

- 이름이 없는 한 줄짜리 함수
- `lambda parameters: expr` # lambda 함수의 구조
  - ✓ parameters는 매개변수, expr는 결과가 return되는 수식
- `lambda x, y: x + y` # 매개변수 x, y를 전달받아 x + y를 리턴

```
sum = lambda x, y: x+y          # lambda는 함수를 반환
print( "정수의 합 : ", sum( 10, 20 )) # 10+20을 반환
print( "정수의 합 : ", sum( 20, 20 )) # 20+20을 반환
```

- 두 개 이상의 값을 리턴하려면 튜플, 리스트 사용
- `lambda x, y: (x + y, x * y)` # (x+y, x*y) 튜플을 반환
- 람다 함수의 호출

```
>>> (lambda x, y: x + y)(1, 2)
3
```

# 재귀 함수

## ■ 재귀 함수recursion function

- 함수 내에서 자신을 호출하는 함수
- 함수가 종료되는 조건을 반드시 포함해야 함(상수 반환문)
- factorial을 구하는 재귀 함수

```
def fact(n):
    if n==0:   # 재귀 함수 종료 조건. n=0이면 종료
        return 1
    else:
        return n*fact(n-1)   # n! = n x (n-1)!

print(fact(5))
```

# 내장 함수

## any

- any(*seq*)
- seq 요소 중 True가 있으면 True 반환

```
>>> any([1,0,0])
True
>>> any([1,-1,-5])
True
>>> any([0,0,0])
False
```

## bin, oct, hex

- bin(x), oct(x), hex(x)
- 10진 정수 x에 대한 2진수, 8진수, 16진수 문자열 반환

```
>>> bin(12)
'0b1100'
```

# 내장 함수

## chr

- chr(*code*)
- 0~255까지의 code에 대한 문자
- chr(0x41) → 'A'

## dir

- dir(*[obj]*)
- obj의 모든 속성(변수와 메소드) 이름을 보여줌

## enumerate

- enumerate(*iterable*,*start*=0)
- 번호(index)와 iterable 요소 쌍을 iterable로 반환

```
>>> names = ['Corey', 'Chris', 'Dave', 'Travis','Smith']
>>> for index, name in enumerate(names, start=1):
            print(index, name)
1 Corey
2 Chris
3 Dave
4 Travis
5 Smith
```

# 내장 함수

## eval

- eval(*expr*,[*globals*[,*locals*]])
- 문자열로 표시된 *expr*의 계산 결과를 반환

```
>>> eval('1+2*3')
7
```

## int ⇔ bin, oct, hex

- int(str, base=10)
- 문자열 str을 2진수, 8진수, 10진수로 변환

```
>>> int('123')
123
>>> int('1010', 2)
10
```

## filter

- filter(*func*, *seq*)
- seq의 각 요소에 대해 func를 호출하여 결과 True인 요소만 모아서 반환

```
>>> list(filter(lambda x: len(x)>2, ['this', 'is', 'a', 'test']))
['this', 'test'] # 리스트 요소 중에서 길이가 2보다 큰 요소만 모음
```

# 내장 함수

## map

- map(*func, seq, ...*)
- seq의 각 요소에 대해 func를 호출하여 실행하고 결과를 반환

```
def Squares(n):
    return n**2
numbers = [1,3,5,9]
#Squares 함수에 numbers 요소를 하나씩 대입
print(list(map(Squares, numbers)))
```

## ord

- ord(*ch*)
- ch에 대한 ASCII 코드 반환
- ord('A') → 65

## repr

- repr(*obj*)
- obj를 문자열로 변환
- repr(b'0011') → "b'0011'"

# 내장 함수

## round

- round(x, n=0)
- 실수 x를 소수점 아래 n자리로 반올림하여 반환

```
>>> round(12.345, 1)
12.3
>>> round(12.345, 2)
12.35
```

## zip

- zip(*iterables): 동일한 개수로 이루어진 자료형을 묶어주는 함수
- 튜플 쌍으로 이루어진 리스트 반환

```
names = ['Peter Parker', 'Clark Kent', 'Wade Wilson', 'Bruce Wayne']
heroes = ['Spiderman', 'Superman', 'Deadpool', 'Batman']
for name, hero in zip(names, heroes):
    print(f'{name} is actually {hero}')
```

# 모듈

- 모듈이란?
  - 함수, 변수와 클래스들을 모아 놓은 파일(라이브러리)
- 모듈을 사용하는 이유
  - 프로그램의 재사용이 가능
- 모듈의 종류
  - 표준 모듈(https://docs.python.org/3/py-modindex.html)
  - 서드 파티(3rd party) 모듈(https://pypi.org/)
  - 사용자 정의 모듈
- 파이썬 모듈의 특징
  - 파이썬은 오픈 소스를 지향하므로 많은 사람들이 개발하여 공개한 수많은 모듈이 존재
  - 오픈 모듈은 무료로 사용 가능
  - 4차산업(빅데이터, 인공지능, 사물인터넷) 관련 오픈 모듈 존재

# 모듈 불러오기

- 서드 파티 모듈 설치
  - c:~〉 python –m pip install *modulename* [--upgrade]
- 모듈 불러오기 (import)
  - 프로그램 외부에 존재하는 모듈을 프로그램 내부로 불러와서 사용하기 위해 필요한 명령
- import *module_name*
  - 예) >>> import socket
    - ✓ socket 모듈에 포함된 속성(변수, 함수, 클래스)를 사용할 수 있음
    - ✓ dir(socket): socket 모듈에 들어 있는 변수, 함수, 클래스를 보여줌
    - ✓ dir(socket.socket): socket 모듈의 socket 클래스에 포함된 변수, 메소드를 보여 줌

```
>>> import socket
>>> socket.gethostname() # 모듈이름.속성 형태로 호출
'DESKTOP-1VE7RCV'
>>> dir(socket)
>>> dir(socket.socket)
```

# 모듈 불러오기

- from *module_name* import *class or function*
  - 예) from socket import gethostname
  - import된 class 또는 functons를 사용할 수 있음

```
>>> from socket import gethostname
>>> gethostname() # 메소드를 직접 호출
'DESKTOP-1VE7RCV'
```

- from *module_name* import *
  - 예) from socket import *
  - socket 모듈의 모든 속성을 사용할 수 있음

```
>>> from socket import *
>>> gethostname() # 메소드를 직접 호출
'DESKTOP-1VE7RCV'
```

# 모듈 불러오기

- import *module_name* as *alias*
  - 모듈 이름이 긴 경우 단축이름으로 불러오기

```
>>> import socket as so # 모듈을 단축이름으로 불러오기
>>> so.gethostname()    # 단축이름으로 속성 호출
'DESKTOP-1VE7RCV'
```

- 모듈 탐색 경로
  - 불러오는 모듈은 현재 디렉토리, sys.path(환경변수)에 지정된 디렉토리 순서로 찾는다

```
>>> import os, sys
>>> os.getcwd()      # 현재 디렉토리 표시
>>> sys.path         # 환경변수에 지정된 디렉토리
```

# 사용자 정의 모듈

## ■ 사용자 정의 모듈 만들기

- 모듈은 일반 프로그램과 같이 작성
- 다음 파일을 myModule.py로 저장

```
def sum(n):
    sum = 0
    for i in range(1, n+1):
        sum = sum + i
    return sum

def power(x, n):
    prod = 1
    for i in range(1, n+1):
        prod = prod * x
    return prod

if __name__ == '__main__':    #직접 호출하면 실행됨
    print(sum(5))
    print(power(2, 3))
```

# 사용자 정의 모듈

## ■ 사용자 정의 모듈의 사용

```
import myModule                 #모듈을 불러온다. 파일이름에서 .py 제외
hab = myModule.sum(10)          #모듈의 sum 함수 실행
pwr = myModule.power(3, 3)      #모듈의 power 함수 실행
print(hab, pwr)
```

## ■ if __name__ == '__main__':의 기능

- 모듈을 독립된 프로그램으로 사용할 때만 실행됨
- 모듈의 기능을 테스트하기 위해 사용됨

```
if __name__ == '__main__': #모듈로 사용될 때는 실행되지 않음
    print(sum(5))
    print(power(2, 3))
```

# 유용한 모듈

## ■ 파이썬 내장 모듈

- 파이썬 인터프리터와 함께 제공되는 모듈
- https://docs.python.org/ko/3/ 에서 내장 모듈에 대한 자세한 설명을 볼 수 있음

## ■ os 모듈

- 운영 체제의 기본 기능을 이용할 수 있음

```
>>> import os
>>> os.system('calc')    # 계산기 프로그램을 불러옴. C:~>calc와 동일
>>> os.getcwd()          # 현 작업 디렉터리를 알 수 있다
'C:\\Users\\dhkim\\AppData\\Local\\Programs\\Microsoft VS Code'\
>>> os.chdir("C:\\temp") # 현 작업 디렉터리를 바꾼다
>>> os.getcwd()
'C:\\temp'
>>> os.listdir("C:\\")   # 모든 파일이름을 알려준다
>>> os.mkdir("ttt")      # 서브 디렉터리를 만든다
```

# 유용한 모듈

## ■ sys 모듈

- 파이썬 인터프리터에 관한 정보를 알 수 있음

```
>>> import sys
>>> sys.version  # 파이썬 버전정보
'3.11.2 (v3.11.2:878ead1ac1, Feb  7 2023, 10:02:41) [Clang 13.0.0 (clang-
1300.0.29.30)]'

>>> sys.prefix   # 파이썬 설치 경로
'C:\\Users\\dhkim\\AppData\\Local\\Programs\\Python\\Python311'

>>> sys.path     # 파이썬이 모듈을 찾는 경로
['', 'C:\\Python311-64\\Lib\\idlelib', 'C:\\Python311-64\\python311.zip',
'C:\\Python311-64\\DLLs', 'C:\\Python311-64\\lib', …

>>> sys.stdin.readline() # 표준 입력 장치. file 객체의 속성과 메소드 사용
hello       ← 입력값
'hello\n'   ← 출력값
>>> sys.stdout.write('Hello') # 표준 출력 장치. file 객체의 속성과 메소드 사용
Hello
```

# 유용한 모듈

## sys.argv 속성

- 명령 라인의 매개변수 전달받음

```
#echo.py
import sys

for i in range(1, len(sys.argv)):
    print(sys.argv[i])
```

```
C:>python echo.py 1st 2nd 3rd

1st     ◀sys.argv[1]
2nd     ◀sys.argv[2]
3rd     ◀sys.argv[3]
```

# 유용한 모듈

## ■ random 모듈

- 난수random number를 만들 때 사용

```
>>> import random
>>> random.randint(1,5)          # 1~5 사이의 임의 정수 생성
4

>>> random.random()              # 0~1 사이의 실수 생성
0.33925056274636767

>>> random.randrange(1, 10, 1)   # 1~9 사이에서 step=1로 난수 발생
9

>>> random.choice('hello')       # 'hello'에서 임의로 하나의 문자 반환
'e'
```

# 유용한 모듈

## math 모듈

- 수학적 계산을 위한 모듈

```
>>> import math
>>> dir(math) # math 모듈의 함수와 상수
['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos',
'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign',
'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs',
'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot',
'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log',
'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin',
'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

>>> math.pi # pi(π) 상수값
3.141592653589793

>>> math.sin(math.pi/6) # sin 함수
0.4999999999999994

>>> math.log10(2) # 로그 함수
0.3010299956639812
```

# 유용한 모듈

## time 모듈

- 시간을 문자열로 표시하는 모듈

```
>>> import time
>>> time.time()          # 1970.1.1 0시부터 경과한 초
1677578405.326278
>>> time.asctime()       # 현재 날짜와 시간
'Tue Feb 28 19:00:24 2023'
```

## calendar 모듈

- 달력을 출력하는 모듈

```
>>> import calendar
>>> print(calendar.month(2023, 2))
   February 2023
Mo Tu We Th Fr Sa Su
       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
```

# 유용한 모듈

## enum 모듈

- 열거형 상수를 정의하기 위한 모듈

```
from enum import Enum
class Color(Enum):
    RED = 1      # name = value
    GREEN = 2
    BLUE = 3

#Color의 속성: Color.RED, Color.GREEN, Color.BLUE
#속성은 name과 value를 가진다
#속성의 value를 변경할 수 없다
print(Color.RED.name, Color.RED.value)

Color.RED = 5    #error 발생. 상수이므로 변경할 수 없음

for x in Color:  #Color의 모든 속성 출력
    print(x)
```

# 객체지향 프로그래밍

- 소프트웨어도 객체들로 구성될 수 있다는 생각을 기반으로 한 프로그래밍 방법
- 객체지향 프로그램은 객체의 상태(속성)와 동작(함수)을 나타내는 클래스를 작성하고, 이로부터 객체를 생성(인스턴스화; Instantiation)하여 프로그램을 작성
- 클래스는 사용자 정의 자료 구조
- 클래스의 동작을 나타내는 함수를 메소드method라 함
- 클래스는 다른 클래스로부터 상속받을 수 있음. 상속 클래스는 부모 클래스가 갖지 않는 속성과 메소드를 추가로 가질 수 있음
- 객체는 클래스의 속성과 메소드를 가짐
- 객체는 일반 변수처럼 사용
  - 함수의 인자로 사용되고, 함수에서 객체를 반환할 수도 있음
  - 튜플이나 딕셔너리의 요소로 사용될 수 있음

# 클래스 정의

## ■ 클래스 정의

```
class 클래스이름(부모 클래스) : # class 키워드로 정의
    클래스 몸체
```

❶ 클래스는 class 키워드로 정의됨. 부모 클래스는 없으면 생략
❷ 클래스 몸체는 변수(속성)와 메소드로 구성

## ■ 클래스 객체

- 클래스를 인스턴스화instantiation하여 만들어진 것을 객체object라 함
- 객체는 클래스의 속성과 메소드에 접근할 수 있음

```
>>> class Test:                    # 클래스 정의
        name = "홍길동"             # 클래스 속성
>>> t = Test()                     # 객체 생성
>>> print(t.name) # 객체에서 클래스 속성 접근(ObjName.Prop)
홍길동
```

# 클래스 정의

## ■ 클래스의 예

```
class Car :
    def __init__(self, color, speed): # 초기화 메소드. self는 객체를 나타냄
        self.color = color            # 인스턴스 변수 정의
        self.speed = speed

    def speedUp(self, v):             # 메소드의 첫 번째 매개변수는 self
        self.speed = self.speed + v
        return self.speed

    def speedDown(self, v):
        self.speed = self.speed - v   # 다른 메소드에서 정의된 변수 접근
        return self.speed
```

- *self.variable*로 정의된 변수(인스턴스 변수)는 클래스 전체에서 유효함
- *self*가 없는 변수는 메소드 내에서만 유효한 지역변수임

# 클래스 초기화 함수

## ■ 초기화 함수(__init__())

- 객체를 생성할 때 자동실행 메소드
- 인스턴스 변수(*self.var*)를 초기화할 때 사용
  - ✓ 초기화 필요가 없으면 생략 가능
- *self*를 첫 번째 매개변수로 가지며, *self*는 클래스 객체를 나타냄
- 인스턴스 변수는 클래스 내부의 모든 메소드에서 접근 가능
- 객체를 생성할 때 자동 실행할 메소드를 호출할 수도 있음

```
class Car :
    def __init__(self, color, speed):
        self.color = color # 인스턴스 변수 self.color 초기화
        self.speed = speed # 인스턴스 변수 self.color 초기화
```

# 클래스 객체

- Car 클래스 객체를 이용한 프로그램
  - Car 클래스 객체 mycar 생성
  - Car 클래스의 color와 speed 속성 접근
  - Car 클래스의 speedup()과 speeddown() 메소드 호출

```
mycar = Car('Black', 60)          #Car class 객체 생성
print('색상: ', mycar.color, '속도: ', mycar.speed)        #속성 접근

mycar.color = "Red"               #속성 변경도 가능
print('색상: ', mycar.color)

mycar.speedUp(10)                 #speedup() method 호출
print('속도: ', mycar.speed)

mycar.speedDown(20)               #speeddown() method 호출
print('속도: ', mycar.speed)
```

# 클래스 상속 inheritance

## ■ 클래스 상속

- 객체지향프로그램의 장점은 상속을 통해 코드를 재사용할 수 있음
- 상속은 기존에 존재하는 클래스로부터 속성과 메소드를 이어받고 자신이 필요한 기능을 추가하는 기법
- 상위 클래스: 부모 클래스 또는 super class
- 하위 클래스: 자식 클래스 또는 sub class
- 상속의 구현

```
class subclass(superclass):
```

- 파이썬 프로그램의 모듈은 대부분 클래스로 구현되어 있고 자신의 프로그램에서는 모듈에 포함된 부모 클래스를 상속받아 구현됨

```
class Handler(serversocket.BaseRequestHandler):
```

  - ✓ serversocket 모듈의 BaseRequestHandler 클래스를 상속받아 Handler 클래스 구현

# 클래스 상속

## ■ 클래스 상속

- People 클래스에서 상속 받아 Teacher 클래스 정의
- 부모 클래스 호출: super().method(args)/People.method(self, args)

```
class People :
    def __init__(self, age=0, name=None):
        self.__age = age
        self.__name = name

    def introMe(self):
        print("Name :", self.__name, "age :", str(self.__age))

class Teacher(People) :
    def __init__(self, age=0, name=None, school=None) :
        super().__init__(age, name) # 부모 클래스의 속성 할당. self 없음
        self.school = school        # 자신의 인스턴스 변수 추가

    def showSchool(self):
        print("My School is ", self.school)
```

# 클래스 상속

## ■ 클래스 상속

- 상속 메소드 호출

```
>>> p1 = People(29, "Lee")          ❶ People 객체
>>> p1.introMe()                    ❷ People.introMe() 호출
Name : Lee age : 29


>>> t1 = Teacher(48, "Kim", "HighSchool") ❸ Teacher 객체
>>> t1.introMe()                          ❹ People.introMe() 호출
Name : Kim age : 48


>>> t1.showSchool()                 ❺ Teacher.showSchool() 호출
My School is  HighSchool
```

❹에서 Teacher 클래스에는 introMe()가 정의되어 있지 않으나  부모 클래스인 People의 introMe()를 상속 받았으므로 t1에서 introMe() 호출 가능
❺에서 Teacher 클래스에 추가된 showSchool() 메소드 호출

# 메소드 오버라이딩

## ■ 메소드 오버라이딩Method Overriding

- 자식클래스에서 부모클래스의 메소드 중 필요한 것을 수정해 다시 정의하는 것
- Student 클래스에서 People 클래스의 introMe() 메소드를 재정의하여 사용

```
class People :
    def __init__(self, age=0, name=None):
        self.__age = age
        self.__name = name
    def introMe(self):
        print("Name :", self.__name, "age :", str(self.__age))

class Student(People) :
    def __init__(self, age=0, name=None, grade=None):
         super().__init__(age, name)
         self.__grade = grade
    def introMe(self): # 부모 클래스 메소드를 재정의(오버라이딩)
        super().introMe()
        print("Grade : ", self.__grade)
```

# 메소드 오버라이딩

- 오버라이딩 메소드 호출

```
p1 = People(29, "Lee")
p1.introMe()        # People의 introMe() 호출

s1 = Student(17, "Park", 2)
s1.introMe()        # Student의 introMe() 호출

Name : Lee age : 29
Name : Park age : 17 Grade: 2
```

# 클래스 분리

## ■ 클래스 분리

- 클래스 크기가 너무 커지면 두 개의 클래스로 분리하고 한 클래스의 객체를 다른 클래스의 속성으로 사용하면 하나로 된 클래스와 같은 효과를 가지게 할 수 있음

```
class Car():
    # to represent a car
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_car_name(self):
        long_name = str(self.year) + ' ' + self.make
+ ' ' + self.model
        return long_name.title()

class ElectricCar(Car):
    def __init__(self, make, model, year):
        # Initialize attributes of the parent class.
        super().__init__(make, model, year)
        self.battery = Battery() ❶
```

```
class Battery():
    # to model a battery for an electric car.
    def __init__(self, battery_size=70):
        # Initialize the battery's attributes.
        self.battery_size = battery_size

    def describe_battery(self):
        # a statement describing the battery
size.
        print("This car has a " +
str(self.battery_size) + "-kWh, ! battery.")

my_tesla = ElectricCar('tesla', 'model s', 2016)
print(my_tesla.get_car_name())
my_tesla.battery.describe_battery() ❷
```

❶에서 클래스 Battery를 속성으로 사용
❷ my_tesla 객체에서 Battery 클래스의 describe_battery() 호출

# 다형성 polymorphism

- 전달된 인자에 따라 함수 또는 연산의 기능이 달라지는 기능
  - 2 + 3 → 5
  - '2' + '3' → '23'
  - 'Hello ' + 'World' → 'Hello World'

```
class Korean(object):
    def greeting(self):
        print("안녕하세요")

class American(object):
    def greeting(self):
        print("Hello")

def sayhello(people):
    people.greeting()

Kim = Korean()              # Korean 클래스 객체
John = American()           # American 클래스 객체
sayhello(Kim)               # 동일한 함수인데 인자에 따라 출력이 다름(안녕하세요)
sayhello(John)              # John을 전달하면 Hello
```

# object 클래스

- 자식 클래스를 만들 때 부모 클래스를 명시적으로 지정해 주지 않으면 object 클래스를 부모 클래스로 사용
  - `class className:  == class calssName(object):`
- object 클래스는 가장 상위 클래스
- object 클래스의 메소드
  - `__new__()` 메소드는 객체가 생성될 때 자동으로 호출됨
  - 그 다음 이 메소드는 객체를 초기화하려고 `__init__()` 메소드를 호출
    - ✓ 보통은 새 클래스에 정의된 데이터 필드를 초기화하기 위해 `__init__()` 메소드만 오버라이드 하면 됨
  - `__str__()` 메소드는 객체의 클래스 이름과 객체의 메모리 주소로 구성된 객체에 대한 설명을 문자열로 반환

# object 클래스

## object 클래스 예제

```
class People(object) :
    def __init__(self, age=0, name=None):
        self.__age = age
        self.__name = name

    def __str__(self):   # __str__() 재정의
        return "info -- Name : "+self.__name+" age :"+str(self.__age)

p1 = People(29, "John")
print(p1)

info—Name: John age : 29
```

# 예외

## ■ 예외(exception)란?

- 프로그램 실행 도중에 오류가 발생하면 생성되는 특별한 객체
  - ✓ print(1/0)을 실행하면 ZeroDivisionError 예외 발생
- 예외가 발생하면 파이썬은 프로그램 실행을 중단
- 예외처리란 예외가 발생했을 때 프로그램 실행을 중단하지 않고 어떻게 처리할 것인지 알려주는 프로그램 부분

- 예외의 종류
  - ✓ ZeroDivisionError 0으로 나눌 때
  - ✓ IndexError 인덱스가 범위를 벗어날 때
  - ✓ FileNotFoundError 없는 파일을 열려고 할 때
  - ✓ ValueError 원하는 값을 입력 받지 못할 때
  - ✓ NameError 정의되지 않은 변수를 사용할 때
  - ✓ TypeError 데이터형이 맞지 않는 연산을 할 때

# 예외

■ 여러가지 예외

```
>>> 5/0
ZeroDivisionError: division by zero

>>> a = [0,1,2]
>>> a[3]
IndexError: list index out of range

>>> fp = open("Nofile.txt","r")
FileNotFoundError: [Errno 2] No such file or directory: 'Nofile.txt’

>>> int(input('Type float number or string : '))
Type float number or string : 1.23
ValueError: invalid literal for int() with base 10: '1.23’

>>> 1 + var*2
NameError: name 'var' is not defined

>>> "str" + 1
TypeError: can only concatenate str (not "int") to str
```

# 예외처리

## ■ 예외처리

- try-except/try-except-else 문

```
try:
    a = 1 / b ❶
except ZeroDivisionError:
    print("0으로 나눌 수 없습니다") ❷
else:
    print(a) ❸
```

- ❶ a=1/b를 실행하고 ZeroDivisionError 예외가 발생하면 ❷ 문장 실행
- ZeroDivisionError가 아닌 다른 예외가 발생하면 프로그램 중단
- 예외가 발생하지 않으면 ❸ 문장 실행
- else: 블록은 생략 가능
- except ZeroDivisionError: 에서 ZeroDivisionError를 생략하면 모든 예외에 대해 ❷ 문장 실행

# 예외처리

## ■ 예외를 알 수 없는 경우

```
try:
    a = int(input("Type a Number: "))❶
except Exception as e: ❷
    print("예외가 발생했습니다", e) ❸
else:
    print(a) ❹
```

- ❶을 실행하고 예외가 발생하면 ❸ 문장 실행
- 예외의 종류는 모르지만 예외가 발생할 이유를 알고 싶을 때 ❷와 같이 사용. e는 예외와 관련된 메시지 저장
- 예외가 발생하지 않으면 ❹ 문장 실행

# 예외처리

## ■ 복합 try-except 문

```
import sys
try:
    fp = open('sample.txt')
    sl = fp.readline()
    value = int(sl.strip())
except OSError as err:
    print("OS 오류: ", err) ❶
except ValueError:
    print("정수로 변환할 수 없습니다") ❷
except:
    print(알 수 없는 오류가 발생하였습니다") ❸
```

- try: 블록 실행 결과 OSError 예외가 발생하면 ❶문장 실행
- ValueError 예외가 발생하면 ❷문장 실행
- 기타 예외가 발생하면 ❸문장 실행

# 예외처리

## try-except-finally 문

```
try:
    result = x / y ❶
except ZeroDivisionError:
    print("0으로 나눌 수 없습니다") ❷
except:
    print(result) ❸
finally:
     print("예외처리가 끝났습니다") ❹
```

- ❶문장을 실행하고 ZeroDivisionError 예외가 발생하면 ❷ 문장 실행
- 기타 예외가 발생하면 ❸ 문장 실행
- 예외 발생에 관계없이 ❹ 문장 실행. 따라서 ❹문장은 항상 실행됨

# 연습문제

- 3명 이상 친구 이름 리스트를 작성하고 다음 내용을 프로그램 하라.
  - insert()로 맨 앞에 새로운 친구 추가
    - ✓ 사용법: list.insert(pos, val)
      - pos 위치에 val을 삽입
  - insert()로 3번째 위치에 새로운 친구 추가
  - append()로 마지막에 친구 추가

- 리스트 [1, 2, 3]에 대해 다음과 같은 처리를 하라.
  - 두 번째 요소를 17로 수정
  - 리스트에 4, 5, 6을 추가
  - 첫 번째 요소 제거
  - 리스트를 요소 오름차순 정렬하기
  - 리스트를 요소 내림차순 정렬하기
  - 인덱스 3에 25넣기

# 연습문제

- 다음과 같은 게임 프로그램을 작성하라.
  - 플레이어가 처음에 $50을 가지고 있다. 동전을 한 번 던져서 앞면(1) 또는 뒷면(2)이 나온다. 맞추면 $9을 따고 틀리면 $10을 잃는다. 플레이어가 돈을 모두 잃거나 $100이 되면 게임이 종료된다.
  - 동전을 던져서 나오는 수는 다음 문장을 이용하라.

```
from random import randint
coin = randint(1,2) #1 또는 2를 임의로 발생
```

- 두 수의 최대 공약수는 두 수를 나누어 떨어지는 가장 큰 수이다. 예를 들어 (16, 24)의 최대 공약수는 8이다. 두 수를 입력 받아 다음 알고리즘에 의해 최대 공약수를 구하는 프로그램을 작성하라.
  - 큰 수를 작은 수로 나눈 나머지를 구하라
  - 큰 수를 작은 수로 대체하고 작은 수는 나머지로 대체하라
  - 작은 수가 0이 될 때까지 이 과정을 반복하라. 마지막 큰 수가 최대 공약수이다.

# 연습문제

- 문자 'a'가 들어가는 단어를 키보드에서 입력 받아 첫 번째 줄에는 'a'까지의 문자열을 출력하고 두 번째 줄에는 나머지 문자열을 출력하는 프로그램을 작성하라.

```
Your word: Buffalo
Buffa
lo
```

- 숫자를 문자열로 변화하는 방법은 str(num)을 이용한다. str(12) → '12'가 된다. 반대로 문자열을 숫자로 변환하려면 int(string)을 이용한다. int('12') → 12가 된다. 이를 이용하여 1부터 1000까지의 숫자의 각 자리수의 합을 모두 구하라. 예를 들어 234 → 2+3+4=9가 된다.

```
[Hint]
sum = 0
for s in '234':
    sum += int(s)
```

- for 루프를 이용하여 다음과 같은 리스트를 생성하라.
  - 0~49까지의 수로 구성되는 리스트
  - 0~49까지 수의 제곱으로 구성되는 리스트

# 연습문제

■ 아래 내용에 대한 프로그램(1개)을 작성하라.

```
days = {'January':31, 'February':28, 'March':31, 'April':30,
        'May':31, 'June':30, 'July':31, 'August':31,
        'September':30, 'October':31, 'November':30,
        'December':31}
```

- 사용자가 월을 입력하면 해당 월에 일수를 출력하라.
- 알파벳 순서로 모든 월을 출력하라.
- 일수가 31인 월을 모두 출력하라.
- 월의 일수를 기준으로 오름차순으로 (key-value) 쌍을 출력하라.
- 사용자가 월을 3자리만 입력하면 월의 일수를 출력하라.(Jan, Feb 등)

# 연습문제

- 아래 내용에 대한 프로그램(1개)을 작성하라.

```
d=[{'name':'Todd', 'phone':'555-1414’, ‘email':'todd@mail.net’},
   {'name':'Helga', 'phone':'555-1618’, 'email':'helga@mail.net’},
   {'name':'Princess', 'phone':'555-3141', 'email':''},
   {'name':'LJ', 'phone':'555-2718', 'email':'lj@mail.net'}]
```

  - 전화번호가 8로 끝나는 사용자 이름을 출력하라.
  - 이메일이 없는 사용자 이름을 출력하라.
  - 사용자 이름을 입력하면 전화번호, 이메일을 출력하라. 이름이 없으면 '이름이 없습니다'라는 메시지를 출력하라.

- 다음과 같이 구성되는 문자열을 구분 문자(&, =)로 분리하여 딕셔너리로 반환하는 함수를 포함하는 프로그램을 작성하라.

예) 문자열 'led=on&motor=off&switch=off'이고 구분 문자가 '&', '='일 때

```
{'led':'on', 'motor':'off', 'switch':off'} 반환
```

# 연습문제

- 다음 Person 클래스를 상속 받는 Employee 클래스를 정의하라. Employee 클래스에 employeeID 속성을 추가하고 getID() 메소드를 정의하라. getID() 메소드는 employeeID를 반환하는 메소드이다. 최종적으로 Employee 클래스를 이용하여 Employee("IoT", 65, 2018)로 생성된 객체의 이름, 나이, ID를 출력하는 프로그램을 작성하라.

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def getName(self):
        print(self.name)

    def getAge(self):
        print(self.age)
```

# Thank you

Questions?

Contact: daeheekim@sch.ac.kr