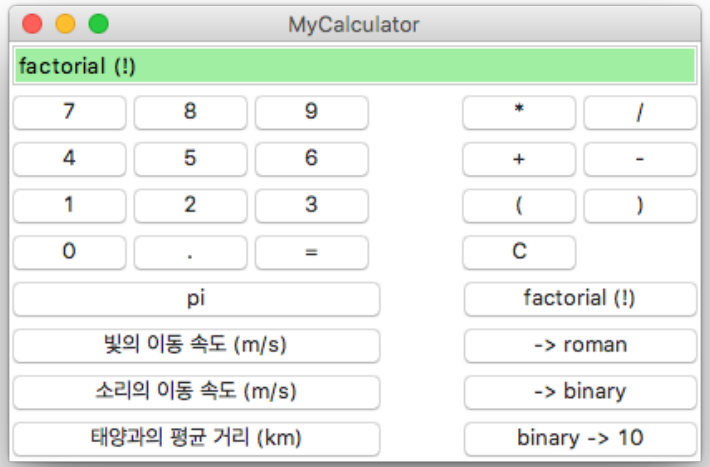
Programming in Python

16 – 계산기 알고리즘 덧붙이기 (2)

2016년 8월, 국민대학교 컴퓨터공학부

지금까지 만들었던 계산기



calc_functions.py

```
def  factorial(n):
        return "factorial (!)"
```

- 숫자 키패드 버튼을 눌러 숫자를 입력할 수 있다.
  - 키보드 입력도 가능
- '=' 버튼을 눌러 수식을 계산할 수 있다.
- 'C' 버튼을 눌러 수식 창을 초기화할 수 있다.
- 상수 버튼을 누르면 해당 상수가 표시창에 나타난다.
- 함수 버튼을 누르면 해당 함수의 설명이 표시창에 나타난다.

  → 함수 버튼을 누르면 표시창의 수에 함수를 적용해 계산한 값을 출력한다.

Factorial (!) 알고리즘

n! = n X (n - 1) X ⋯ X 1

5! = 5 X 4 X 3 X 2 X 1
= 120

이러한 계산을 위하여

가장 알맞은

프로그램 구문 구조는

무엇일까?

Factorial (!) 알고리즘 구현

```
def factorial(n):
    fact = 1
    while n > 1:
        fact *= n
        n -= 1

    return fact
```

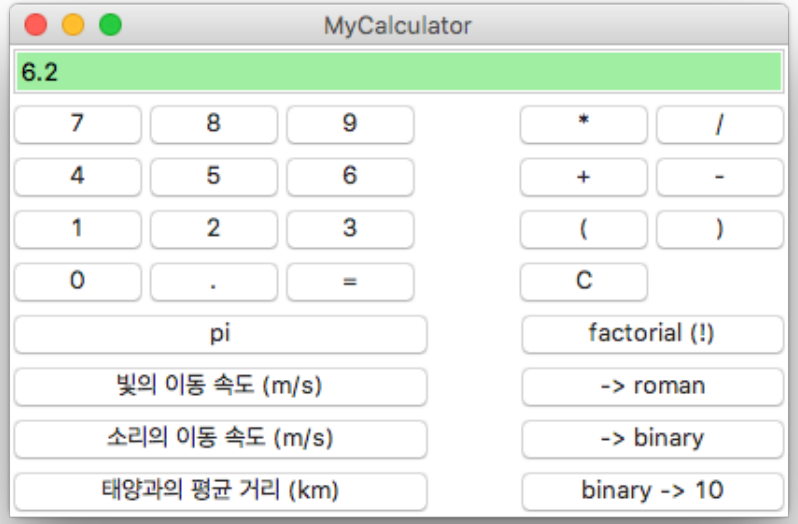
이렇게 하면

될까?

```
Exception in Tkinter callback
Traceback (most recent call last):
...
TypeError: unsupported operand type(s) for -=: 'str' and 'int'
```

Factorial (!) 알고리즘 구현

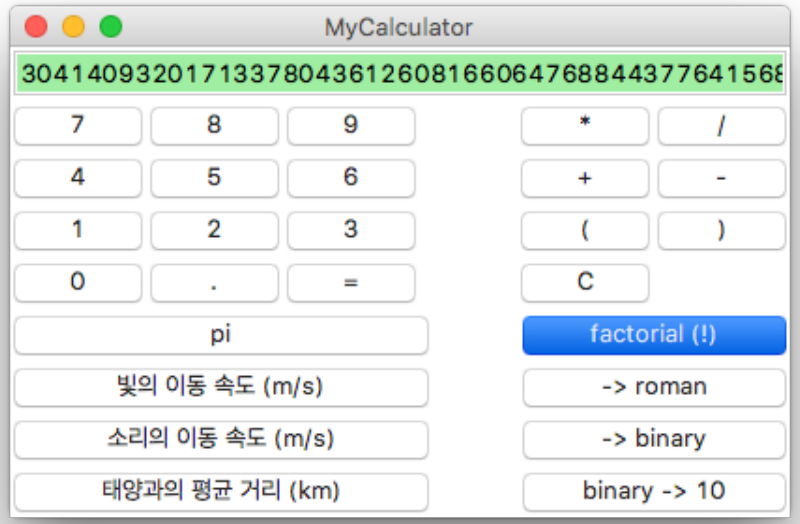
```
def factorial(x):
    n = int(x)
    fact = 1
    while n > 1:
        fact *= n
        n -= 1

    return fact
```



Exception in Tkinter callback
Traceback (most recent call last):
n = int(x)
ValueError: invalid literal for int() with base 10: '6.2'

Factorial (!) 알고리즘 구현

```
def factorial(x):
    try:
        n = int(x)
    except:
        return "--> 오류!"
    fact = 1
    while n > 1:
        fact *= n
        n -= 1

    return fact
```

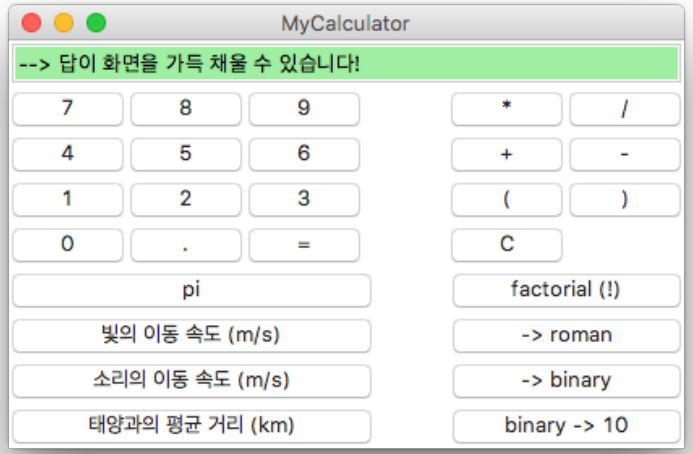


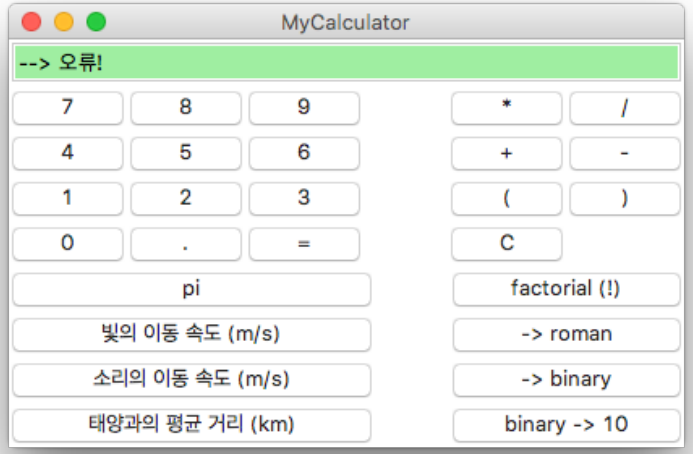
이런 상황도

방지해야겠다!

Factorial (!) 알고리즘 구현

```
def factorial(x):
    try:
        n = int(x)
    except:
        return "--> 오류!"
    if n > 40:
        return "--> 답이 화면을 가득 채울 수 있습니다!"
    if n < 0:
        return "--> 오류!"
    fact = 1
    while n > 1:
        fact *= n
        n -= 1

    return fact
```





Factorial (!) - 이런 것은 누군가 이미 해 두었지 않을까?

```
def factorial_backup(x):
    from math import factorial as f
    try:
        n = int(x)
    except:
        return "--> 오류!"

    if n > 40:
        return "--> 답이 화면을 가득 채울 수 있습니다!"

    try:
        fact = f(n)
    except:
        fact = "--> 오류!"

    return fact
```

로마 숫자 함수 구현

수	로마자
1	I
5	V
10	X
50	L
100	C
500	D
1000	M

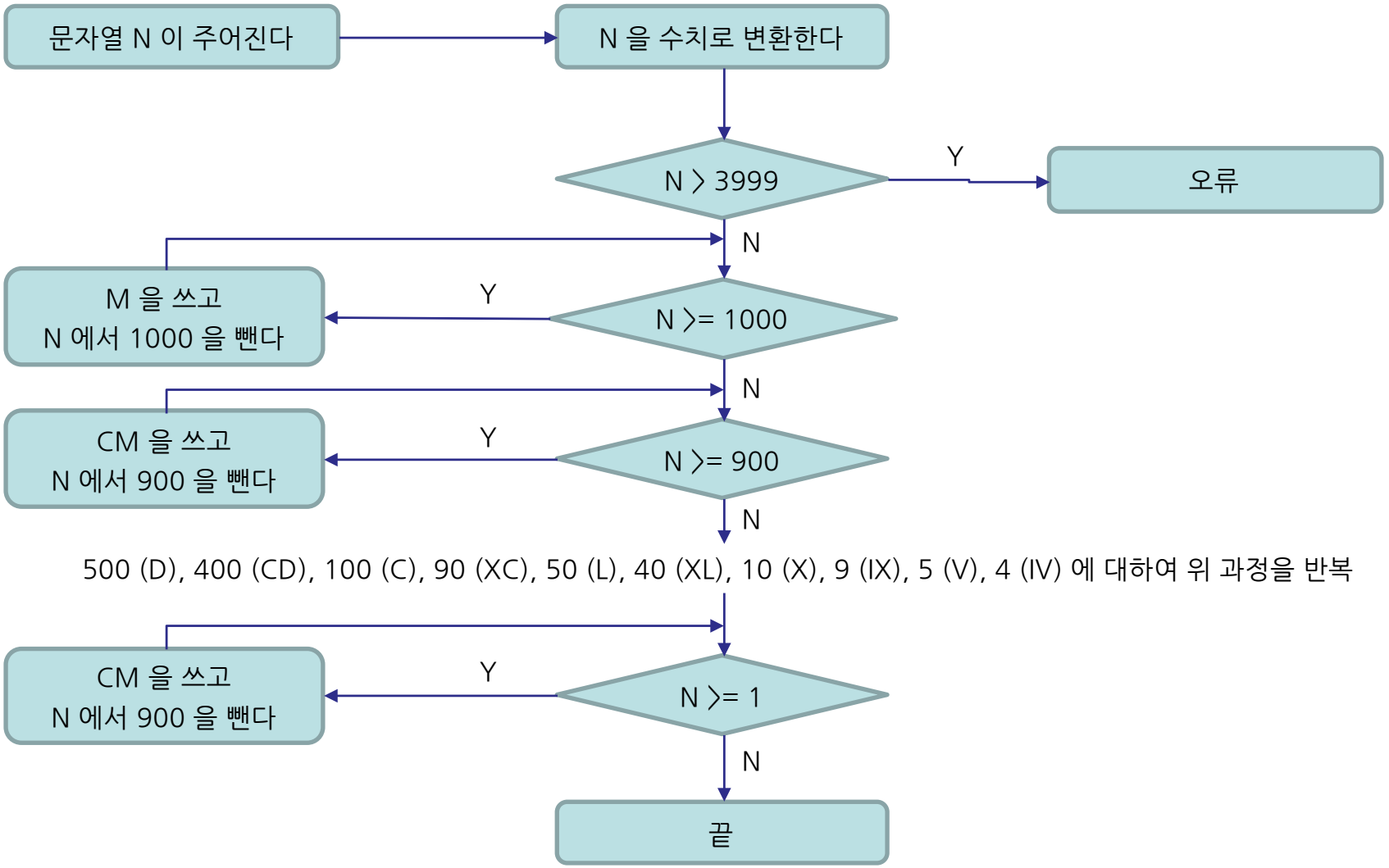
위 표에 있는 문자까지만 사용하면
얼마까지의 수를 표현할 수 있나요?

어떤 수의 오른쪽에 쓰여진 수는 더해지고,
왼쪽에 쓰여진 수는 빼진다.

3 까지는 I 를 거듭 쓴다.
4는 5 - 1 이므로 V 의 왼쪽에 I 를 쓴다.

예: 248 = CCXLVIII

로마 숫자 함수 구현 - 알고리즘 설계



로마 숫자 함수 구현

```
def to_roman(x):

    try:
        n = int(x)
    except:
        return "--> 오류!"

    if n >= 4000:
        return "--> 범위를 넘어섭니다."

    result = ""

    while n >= 1000:
        result += "M"
        n -= 1000
    while n >= 900:
        result += "CM"
        n -= 900
    while n >= 500:
        result += "D"
        n -= 500
    while n >= 400:
        result += "CD"
        n -= 400
    while n >= 100:
        result += "C"
        n -= 100
    while n >= 90:
        result += "XC"
        n -= 90
    while n >= 50:
        result += "L"
        n -= 50
    while n >= 40:
        result += "XL"
        n -= 40
    while n >= 10:
        result += "X"
        n -= 10
    while n >= 9:
        result += "IX"
        n -= 9
    while n >= 5:
        result += "V"
        n -= 5
    while n >= 4:
        result += "IV"
        n -= 4
    while n >= 1:
        result += "I"
        n -= 1

    return result
```

코드의 반복이 심하다.
이후에, 백만까지의 로마 숫자를 표현하도록
프로그램을 고치면 어떻게 될까?

로마 숫자 함수 구현

순서쌍과 사전을 이용해 보자!

```
numberBreaks = (1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1)
letters = {
          1000: "M", 900: "CM", 500: "D", 400: "CD",
           100: "C",  90: "XC",  50: "L",  40: "XL",
            10: "X",   9: "IX",   5: "V",   4: "IV",
             1: "I"
}
```

로마 숫자 함수 구현

```
def to_roman(x):

    try:
        n = int(x)
    except:
        return "--> 오류!"

    if n >= 4000:
        return "--> 범위를 넘어섭니다."

    numberBreaks = (1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1)
    letters = {
       1000: "M", 900: "CM", 500: "D", 400: "CD",
        100: "C",  90: "XC",  50: "L",  40: "XL",
         10: "X",   9: "IX",   5: "V",   4: "IV",
          1: "I"
    }

    result = ""
    for value in numberBreaks:
        while n >= value:
            result = result + letters[value]
            n = n - value

    return result
```

Exercise - 10진수와 2진수간 변환 함수 구현

가장 쉬운 방법은

무엇일까?

Exercise - 하나의 해결안

```
def to_binary(x):

    try:
        n = int(x)
    except:
        return "--> 오류!"

    return bin(n)[2:]


def from_binary(x):

    try:
        return int(x, 2)
    except:
        return "--> 오류!"
```

Quiz

- 올바른 수식이 입력된 상태에서 함수 버튼을 누르면 제대로 동작하지 않는다.
  - 이것을 개선할 수 있는 방법은?

Quiz - 정답

```
elif key in functions_list:
        val = str(eval(display.get()))
        display.delete(0, END)
        display.insert(END, function_map[functions_list.index(key)][1](val))
```

Exercise - 알고리즘 개선 (로마 숫자 변환)

불필요한 반복이 있지는 않은가?
numberBreaks 의 원소와 letters 의 키가 동일해야 하는데…
(상수 버튼 추가할 때의 constants 의 경우와 유사)

```
numberBreaks = (1000, 900, 500, 400, 100, 90, 50, 40, 10, 9, 5, 4, 1)
letters = {
        1000: "M", 900: "CM", 500: "D", 400: "CD",
         100: "C",  90: "XC",  50: "L",  40: "XL",
          10: "X",   9: "IX",   5: "V",   4: "IV",
           1: "I"
}


result = ""
    for value in numberBreaks:
        while n >= value:
            result = result + letters[value]
            n = n - value
```

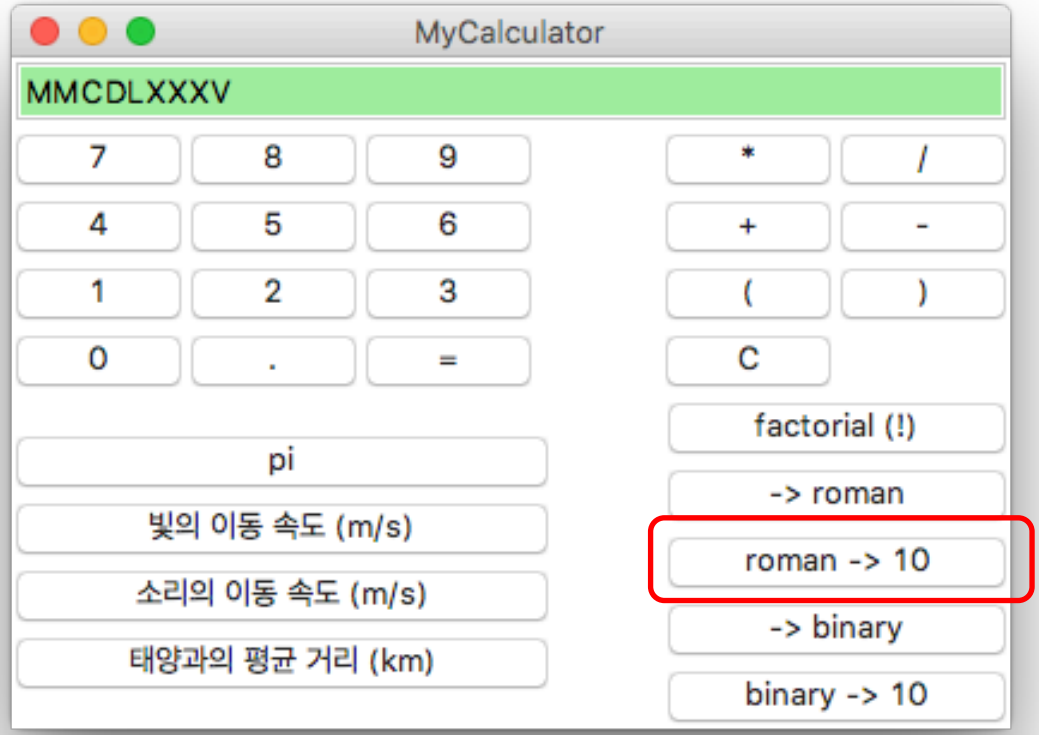
순서쌍을 이용해보자!

```
romans = (
        (1000, "M"), (900, "CM"), (500, "D"), (400, "CD"),
        (100, "C"), (90, "XC"), (50, "L"), (40, "XL"),
        (10, "X"), (5, "V"), (4, "IV"), (1, "I")
    )
```

Exercise - 하나의 해결안

```
def to_roman(x):

    try:
        n = int(x)
    except:
        return "--> 오류!"

    if n > 4999:
        return "--> 범위를 넘어섭니다."

    romans = (
        (1000, "M"), (900, "CM"), (500, "D"), (400, "CD"),
        (100, "C"), (90, "XC"), (50, "L"), (40, "XL"),
        (10, "X"), (5, "V"), (4, "IV"), (1, "I")
    )

    result = ""
    for value, letters in romans:
        while n >= value:
            result = result + letters
            n = n - value

    return result
```

Exercise - 로마 숫자를 10진수로 변환하는 코드 추가



힌트:

문자열을 왼쪽부터 스캔하면서
앞에서 이용했던 로마 숫자 테이블과
매치되는 것이 있는지 살펴보고
매치되는 값을 누적한다.

올바르지 않은 문자열이
입력될 수 있음에 주의!

Exercise - 하나의 해결안

```
def from_roman(x):

    romans = (
        (1000, "M"), (900, "CM"), (500, "D"), (400, "CD"),
        (100, "C"), (90, "XC"), (50, "L"), (40, "XL"),
        (10, "X"), (5, "V"), (4, "IV"), (1, "I")
    )

    result = 0
    for value, letters in romans:
        while x[:len(letters)] == letters:
            result += value
            x = x[len(letters):]

    if len(x) == 0:
        return str(result)
    else:
        return "--> 오류!"
```

Q & A