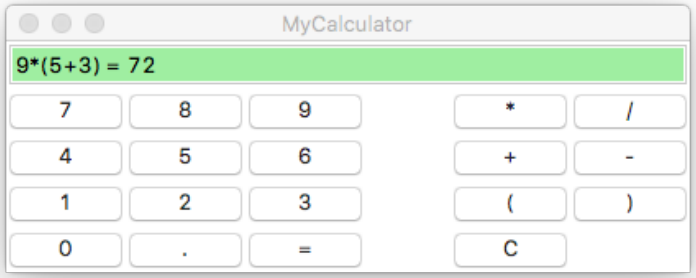
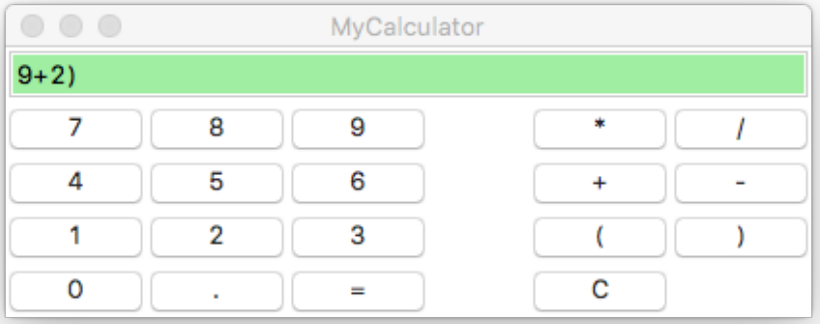
Programming in Python

15 – 계산기 알고리즘 덧붙이기 **(1)**

2016년 **8**월**,** 국민대학교 컴퓨터공학부

지금까지 만들었던 계산기





- 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.

발전시킬 방향:

1. 수식에 오류가 있는 경우를 처리
2. 몇 가지 상수를 입력할 수 있는 버튼을 추가
3. 몇 가지 함수를 계산할 수 있는 버튼을 추가

```
Exception in Tkinter callback
Traceback (most recent call last):
  File "…/Tkinter.py", line 1410, in __call__
    return self.func(*args)
  File "basic.py", line 46, in cmd
    click(x)
  File "basic.py", line 36, in click
    result = str(eval(display.get()))
  File "<string>", line 1
    9+2)
       ^
SyntaxError: unexpected EOF while parsing
```

Exercise (1) - 예외 처리

오류가 있는 수식이 입력된 채

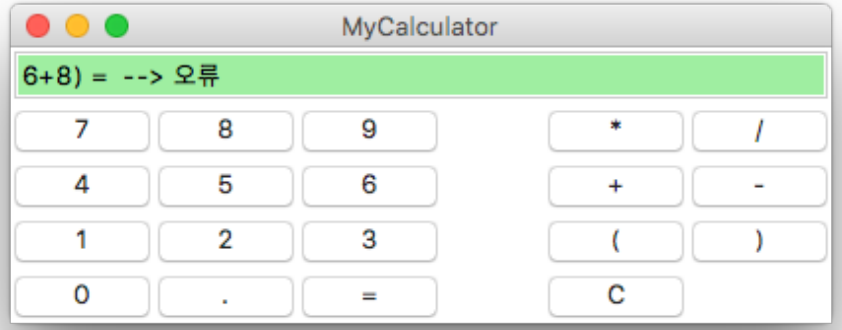
'=' 버튼이 눌렸을 때

오류 메시지를 수식 창에 출력하도록

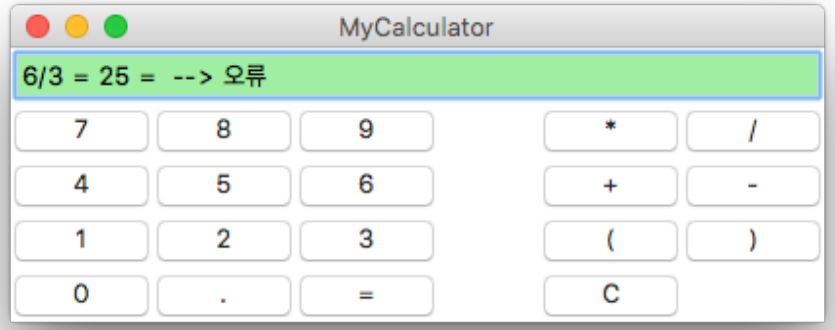
프로그램을 수정해 보자.

Exercise (1) - 하나의 해결안

```
def click(key):
    if key == '=':
        try:
            result = str(eval(display.get()))
        except:
            result = " --> 오류"
        display.insert(END, " = " + result)
    elif key == 'C':
        display.delete(0, END)
    else:
        display.insert(END, key)
```



Exercise (2) - 새로운 계산식으로



아래 버튼을 차례대로 누른 경우의 화면:

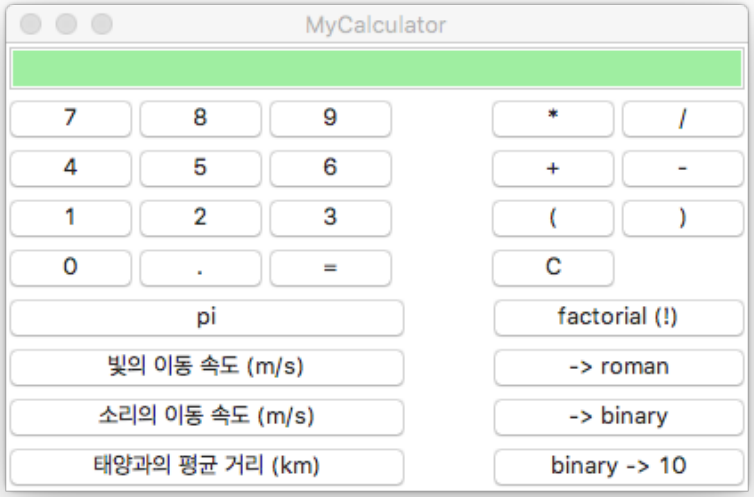
[6]
[/]
[3]
[=]
[5]
[=]

한번 수식을 계산한 결과를 얻은 뒤에는
다시 숫자 버튼을 누를 때
수식 창을 지우고
새로 시작하도록 해보자.

Exercise (2) - 하나의 해결안

```
def click(key):
    if key == '=':
        try:
            result = str(eval(display.get()))
        except:
            result = " --> 오류"
        display.insert(END, " = " + result)
    elif key == 'C':
        display.delete(0, END)
    else:
        if '=' in display.get():
            display.delete(0, END)
        display.insert(END, key)
```

계산기 알고리즘 추가 (안)



우선,

UI 부터

만들어 보자.

- 상수 버튼 모음과 함수 버튼 모음은 각각 프레임으로
  - 상수 버튼: 폭 22, sticky = W
  - 함수 버튼: 폭 13, sticky = E
- 한글 문자가 (주석이라 할지라도!) 포함된 .py 파일에는
  - #-*- coding: utf-8 -*-

참고 - 지난 번까지의 계산기 버튼 생성 코드

```
button_groups = {
        'num': {'list': num_pad_list, 'window': num_pad, 'width': 5, 'cols': 3},
        'op': {'list': operator_list, 'window': operator, 'width': 5, 'cols': 2},
}

for label in button_groups.keys():
        r = 0; c = 0
        buttons = button_groups[label]
        for btn_text in buttons['list']:
                def cmd(x=btn_text):
                        click(x)
                Button(buttons['window'],
                        text=btn_text,
                        width=buttons['width'],
                        command=cmd).grid(row=r, column=c)
                c = c + 1
                if c >= buttons['cols']:
                        c = 0
                        r = r + 1
```

UI 구성 코드

지난번 코드 개선 덕에
손쉽게 버튼들을 추가할 수 있다!

```
constants_list = [
    'pi',
    '빛의 이동 속도 (m/s)',
    '소리의 이동 속도 (m/s)',
    '태양과의 평균 거리 (km)'
]
```

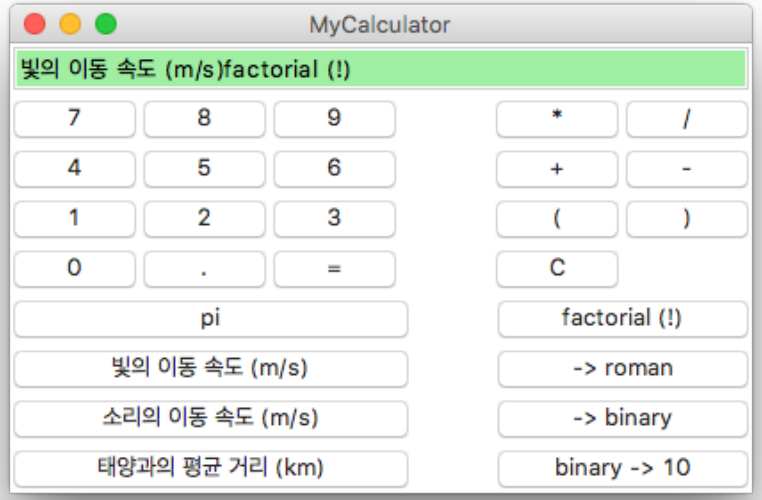
```
functions_list = [
    'factorial (!)',
    '-> roman',
    '-> binary',
    'binary -> 10',
]
```

```
# constants frame
constants = Frame(window)
constants.grid(row=3, column=0, sticky=W)

# functions frame
functions = Frame(window)
functions.grid(row=3, column=1, sticky=E)
```

```
button_groups = {
    'num': {'list': num_pad_list, 'window': num_pad, 'width': 5, 'cols': 3},
    'op': {'list': operator_list, 'window': operator, 'width': 5, 'cols': 2},
    'consts': {'list': constants_list, 'window': constants, 'width': 22, 'cols': 1},
    'funcs': {'list': functions_list, 'window': functions, 'width': 13, 'cols': 1},
}
```

UI 는 구성하였으나…



상수	값
pi	3.141592654
빛의 이동 속도 (m/s)	300000000
소리의 이동 속도 (m/s)	330
태양과의 평균 거리 (km)	149597887.5

- 위 그림은 상수 버튼 하나와 함수 버튼 하나를 차례로 누른 결과
- 버튼이 눌리면, (= 과 C 를 제외하면) 버튼 텍스트를 창에 표시하도록 했으니 당연한 결과임
- 올바른 기능:
  - 상수 버튼을 누르는 경우 해당 상수가 표시되도록
  - 함수 버튼을 누르는 경우 현재 창에 있는 값에 함수 적용하여 계산하도록

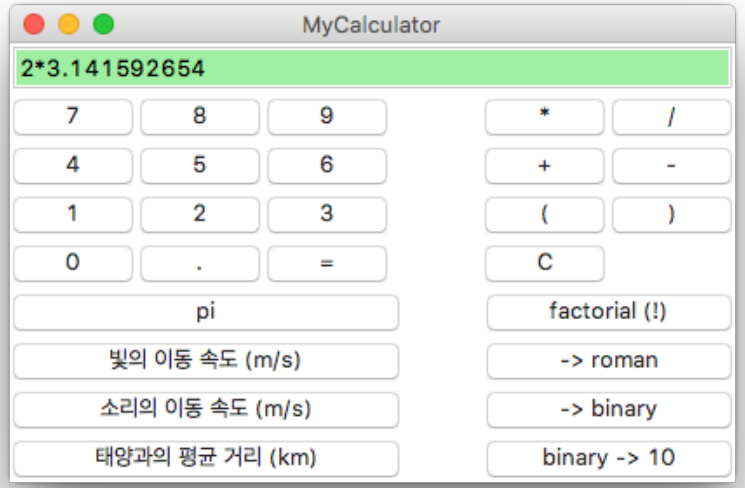
상수 쪽이 쉬우니

이것부터 만들어 보자.

상수 버튼 기능을 구현

이 코드는 어디에 들어가야 맞을까?

```
elif key == constants_list[0]:
        display.insert(END, "3.141592654")
elif key == constants_list[1]:
        display.insert(END, "300000000")
elif key == constants_list[2]:
        display.insert(END, "330")
elif key == constants_list[3]:
        display.insert(END, "149597887.5")
```



상수 쪽을 처리했으니, 이제는

함수 버튼에 대한 처리도

추가해 보자.

- 함수의 계산 알고리즘은 별도의 모듈로 작성
  - 모듈 이름: calc_functions
  - factorial()
  - to_roman()
  - to_binary()
  - from_binary()

함수 버튼 기능을 구현 (인터페이스만)

```
import calc_functions
...
elif key == functions_list[0]:
        n = display.get()
        display.delete(0, END)
        display.insert(END, calc_functions.factorial(n))
elif key == functions_list[1]:
        n = display.get()
        display.delete(0, END)
        display.insert(END, calc_functions.to_roman(n))
elif key == functions_list[2]:
        n = display.get()
        display.delete(0, END)
        display.insert(END, calc_functions.to_binary(n))
elif key == functions_list[3]:
        n = display.get()
        display.delete(0, END)
        display.insert(END, calc_functions.from_binary(n))
```

calc_functions.py

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

def to_roman(n):
        return "-> roman"

def to_binary(n):
        return "-> binary"

def from_binary(n):
        return "binary -> 10"
```

Exercise - 문제점 발견하기

지금까지 작성한 계산기 프로그램에서

불만스러운 점은 무엇일까?

(함수 계산 기능이 동작하지 않는 점 빼고)

Exercise - 문제점 발견하기

발견한 문제점: click() 함수에서 불필요한 코드의 반복이 많다.

- 어떤 점에서 문제가 있는 것일까?
  - 비슷비슷한 코드를 일일이 입력하려니 귀찮다.
  - 나중에 코드를 읽거나 고치기가 번거롭다.
  - 프로그램 구조를 수정하려 하면 어떤 일을 해야 할까?
  - UI 를 수정하면 어떤 일을 해야 할까?
    - 또다른 상수, 함수 버튼의 추가
    - 버튼의 순서를 변경

Exercise - 문제점 해결하기

- 상수, 함수의 정의는 별도의 모듈로 분리한다.
  - 상수: constants.py
  - 함수: functions.py

- 눌린 키에 따라 계산기의 동작을 서로 달리 하려 할 때, 이용할만한 Python 데이터 타입은 무엇일까?
  - 상수에 대해서는 눌린 키로부터 무엇을 얻어와야 할까?
  - 함수에 대해서는 눌린 키로부터 무엇을 얻어와야 할까?

- import 는 어떤 식으로 하는 것이 좋을까?

- click() 함수에서 눌린 키가 상수/함수 버튼에 해당하는지는 어떻게 판단할 수 있을까?

Exercise - 하나의 해결안

constants.py

```
# -*- coding: utf-8 -*-

constants_list = [
        'pi',
        '빛의 이동 속도 (m/s)',
        '소리의 이동 속도 (m/s)',
        '태양과의 평균 거리 (km)'
]

constant_values = {
        'pi': '3.141592654',
        '빛의 이동 속도 (m/s)': '300000000',
        '소리의 이동 속도 (m/s)': '330',
        '태양과의 평균 거리 (km)': '149597887.5',
}
```

```
from constants import constants_list, constant_values
```

click()

```
elif key in constants_list:
        display.insert(END, constant_values[key])
```

조금 더 낫게 할 수는?

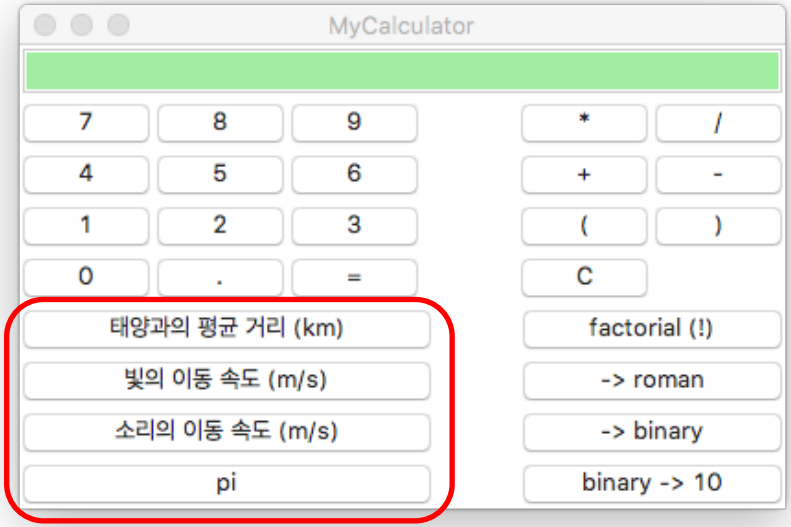
Exercise - 하나의 해결안, 조금 더 낫게

constants.py

```
# -*- coding: utf-8 -*-

constant_values = {
        'pi': '3.141592654',
        '빛의 이동 속도 (m/s)': '300000000',
        '소리의 이동 속도 (m/s)': '330',
        '태양과의 평균 거리 (km)': '149597887.5',
}

constants_list = list(constant_values.keys())
```



문제점 발견!
상수 버튼의 순서를
원하는 대로 유지할 수가 없다.
(왜?)

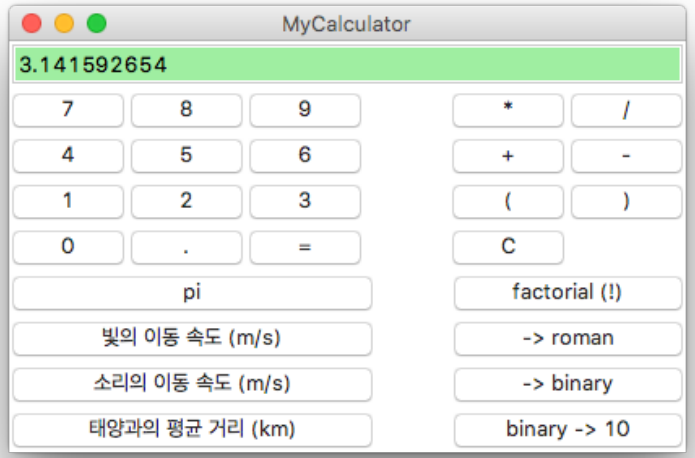
Exercise - 하나의 해결안, 약간 발전한 방향

constants.py

```
# -*- coding: utf-8 -*-

constant_values = [
        ('pi', '3.141592654'),
        ('빛의 이동 속도 (m/s)', '300000000'),
        ('소리의 이동 속도 (m/s)', '330'),
        ('태양과의 평균 거리 (km)', '149597887.5'),
]

constants_list = [x[0] for x in constant_values]
```




```
 elif key in constants_list:
            display.insert(END, constant_values[constants_list.index(key)][1])
```

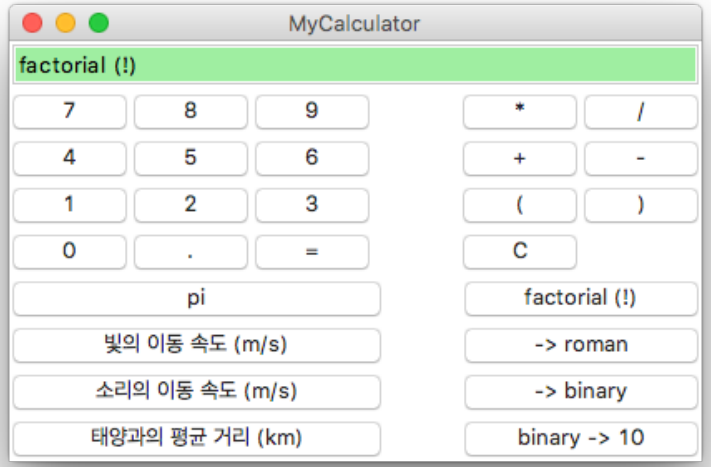
Exercise - 함수에 대해서도 해결

functions.py

```
from calc_functions import *

function_map = [
        ('factorial (!)', factorial),
        ('-> roman', to_roman),
        ('-> binary', to_binary),
        ('binary -> 10', from_binary),
]

functions_list = [x[0] for x in function_map]
```




```
from functions import functions_list, function_map
```

click()

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

Q & A