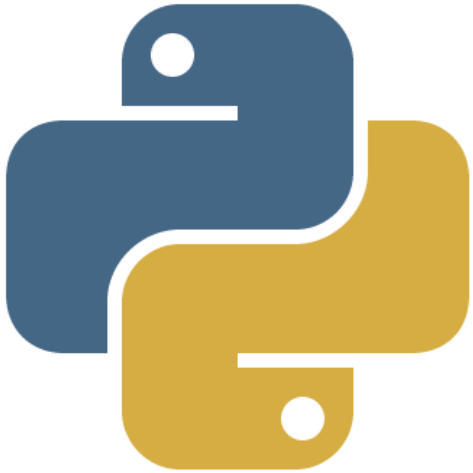
Programming in Python

10 – 간단한 계산기 구현

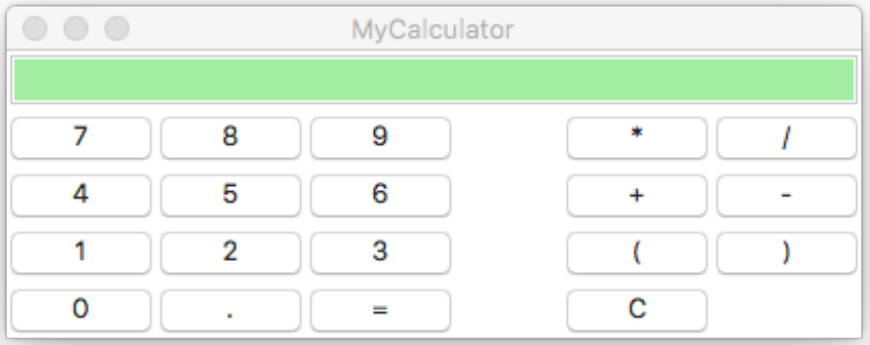


python

2016년 8월, 국민대학교 컴퓨터공학부

간단한 계산기 UI 의 설계

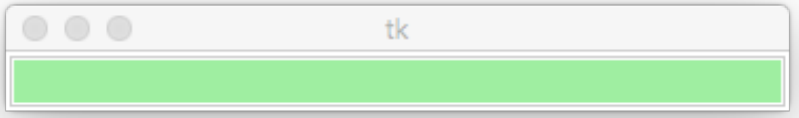
- 상단에는 텍스트 입력과 출력이 가능한 계산식 창이 있다.
  - Frame 과 Entry 이용
- 하단 왼쪽에는 0 부터 9 까지의 숫자와 소숫점, 그리고 = 이 붙은 버튼들이 나열된다.
  - Button 이용
- 하단 오른쪽에는 사칙연산 (덧셈, 뺄셈, 곱셈, 나눗셈) 과 괄호, 그리고 “C” 에 해당하는 버튼들이 나열된다.
  - “C” 는 상단 창의 내용을 모두 지우는 기능



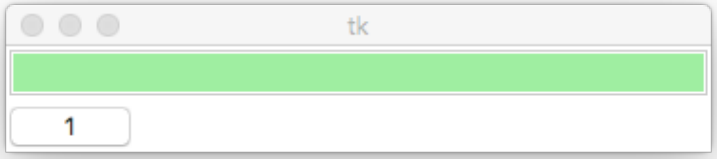
Tk 윈도우의 프레임 (Frames)

하나의 윈도우를 여러 개의 프레임으로 나누어, 각 프레임을 윈도우처럼 이용할 수 있음

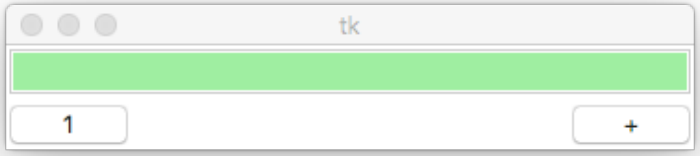
```
>>> top_row = Frame(window)
>>> top_row.grid(row=0, column=0,
                 columnspan=2, sticky=N)
>>> display = Entry(top_row, width=45, bg="light green")
>>> display.grid()
```




```
>>> num_pad = Frame(window)
>>> num_pad.grid(row=1, column=0, sticky=W)
>>> button1 = Button(num_pad, text="1", width=5) \
...                  .grid(row=0, column=0)
```



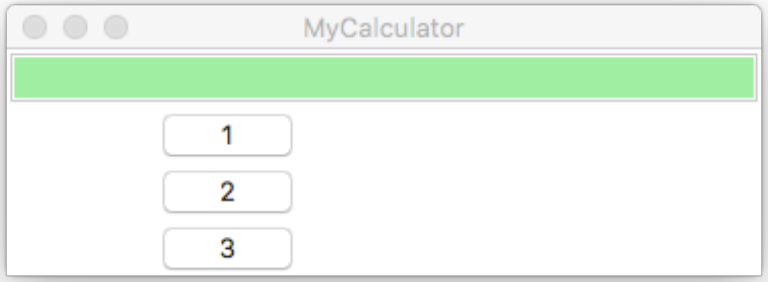

```
>>> operator = Frame(window)
>>> operator.grid(row=1, column=1, sticky=E)
>>> button_plus = Button(operator, text="+", width=5) \
...                       .grid(row=0, column=0)
```



숫자 패드의 UI 구성

```
def click1():
    display.insert(END, "1")
Button(window, text="1", width=5, command=click1).grid(row=1, column=0)

def click2():
    display.insert(END, "2")
Button(window, text="2", width=5, command=click2).grid(row=2, column=0)

def click3():
    display.insert(END, "3")
Button(window, text="3", width=5, command=click3).grid(row=3, column=0)
```

이보다는 나은 방법이
분명히 있을 것!



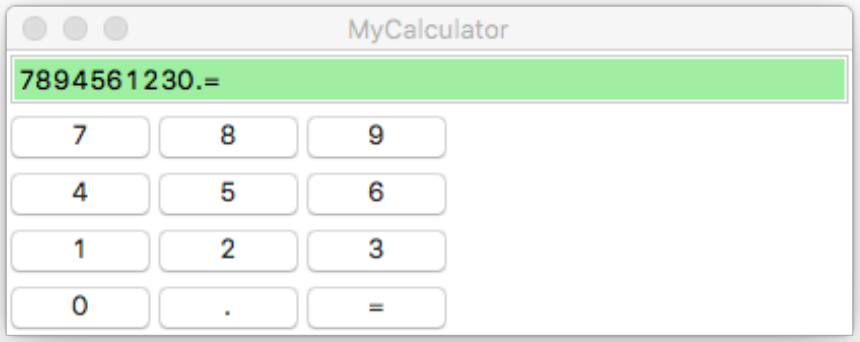
순환문을 이용한 숫자 버튼 생성

```
num_pad_list = [
    '7', '8', '9',
    '4', '5', '6',
    '1', '2', '3',
    '0', '.', '='
]
```

```
def click(key):
    display.insert(END, key)

r = 0; c = 0
for btn_text in num_pad_list:
        Button(num_pad, text=btn_text, width=5, command=click(btn_text)) \
                .grid(row=r, column=c)
        c += 1
        if c > 2:
                c = 0
                r += 1
```

문제점:

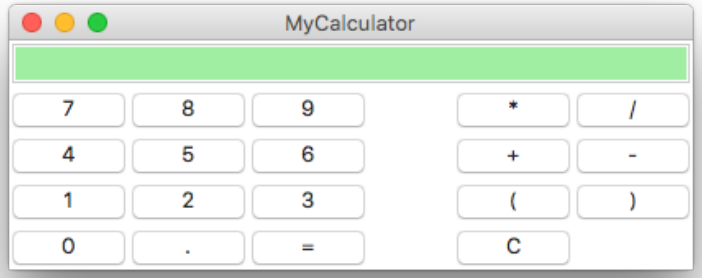
1. 버튼을 누를 때 아무런 반응이 없다.
2. 프로그램을 처음 실행하면 7894561230.= 이 표시된다.



순환문을 이용한 연산자 버튼 생성

```
operator_list = [
    '*', '/',
    '+', '-',
    '(', ')',
    'C'
]
```

```
r = 0; c = 0
for btn_text in operator_list:
        Button(operator, text=btn_text, width=5, command=click) \
                .grid(row=r, column=c)
        c += 1
        if c > 1:
                c = 0
                r += 1
```




```
Exception in Tkinter callback
Traceback (most recent call last):
  File "/Library/Frameworks/Python.framework/Versions/2.7/lib/python2.7/lib-tk/Tkinter.py", line 1410, in __call__
    return self.func(*args)
TypeError: click() takes exactly 1 argument (0 given)
```

함수 인자 기본값을 이용한 문제 해결

```
>>> def func(x="default text"):
...     print(x)
...
>>> func("two")
two

>>> func(3)
3

>>> func()
default text
```

함수 func() 에 x 인자가 전달되지 않으면
x = "default text" 로 간주한다는 뜻

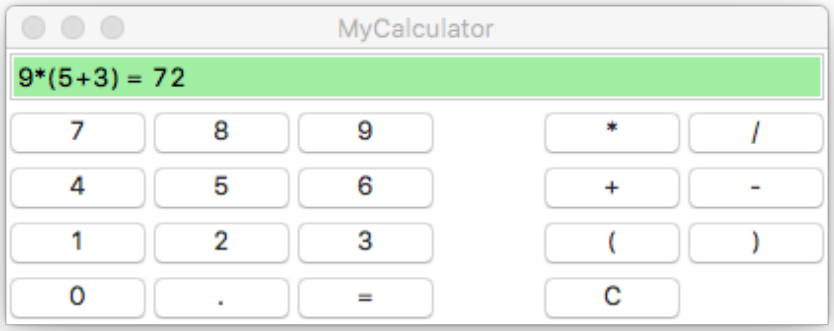
```
>>> def f(x):
...     return x + 1
...
>>> f
<function f at 0x1018a1230>

>>> def f(x):
...     return x * 2
...
>>> f
<function f at 0x1018a12a8>
```

```
r = 0; c = 0
for btn_text in num_pad_list:
        def cmd(x=btn_text):
                click(x)
        Button(num_pad, text=btn_text, width=5, command=cmd) \
                .grid(row=r, column=c)
        c += 1
        if c > 2:
                c = 0
                r += 1
```

이젠 버튼을 누르면 숫자나 연산 기호가 표시됩니다!
(그러나, 계산기 기능은 안됩니다. 프로그램하지 않았으므로)

계산기 기능 완성 - Exercise

- 다른 버튼들과 달리, (표시창에 버튼에 해당하는 내용을 추가) 아래 두 버튼은 기능이 있음
  - = 버튼: 현재까지 만들어진 수식의 값을 계산하여 표시창에 덧붙임
  - C 버튼: 표시창의 내용을 모두 지움
- 힌트: 수식의 계산을 어떻게?
  - eval() 함수를 떠올린다.



Exercise - 정답

```
def click(key):

    if key == '=':
        result = str(eval(display.get()))
        display.insert(END, " = " + result)

    elif key == 'C':
        display.delete(0, END)

    else:
        display.insert(END, key)
```

발전 과제

- 숫자 버튼을 생성하는 코드와 연산 기호 버튼을 생성하는 코드를 비교해 본다.
  - 매우 비슷하다 - 그렇다면 한 벌의 코드로 이용하는 쪽이 좋겠다.
  - (앞으로 두 세트의 버튼을 더 추가하게 됨)

- 숫자 버튼과 연산 기호 버튼에 대해 각각 정보를 기록해 두고 그것을 이용하자.
  - 버튼 텍스트가 기록된 리스트 (num_pad_list, operator_list)
  - 어느 윈도우 (프레임) 에 생성할지
  - 버튼의 폭 (지금은 5 로 동일하지만, 앞으로 추가할 버튼들은 다르므로)
  - 한 행에 몇 개의 버튼을 표시할지 (숫자는 3, 연산 기호는 2)
  - 이것을 기록하기에 적절한 데이터 타입은 무엇인가?

발전 과제 - 하나의 해결안

```
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
```

Q & A