# Python Programming

Tkinter

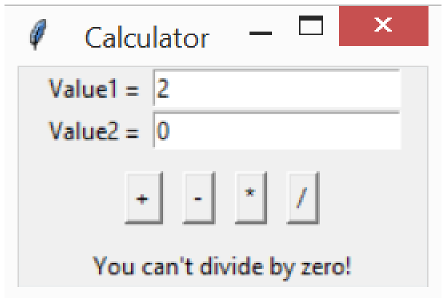
Mariusz Dzieńkowski

Institute of Computer Science
Lublin University of Technology
m.dzienkowski@pollub.pl

# Simple calculator

- Widgets used in the calculator app:
- 3 Label widtets: `Value1=` and `Value2=` and `Result=`
- 2 Entry widtets:
  - for the value of `e1`, after `Value1=`
  - for the value of `e2`, after `Value2=`
- 4 Button widgets, with the text `+, -, *, /` respectively

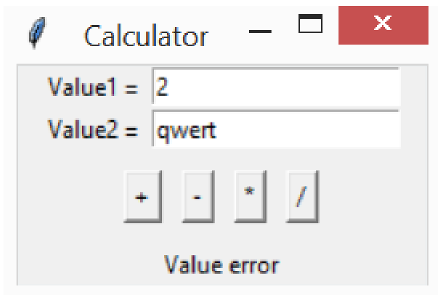
Clicking on a button will trigger the callback function `calc(self,op)` to calculate a result.


Calculator

Value1 = 2

Value2 = 0

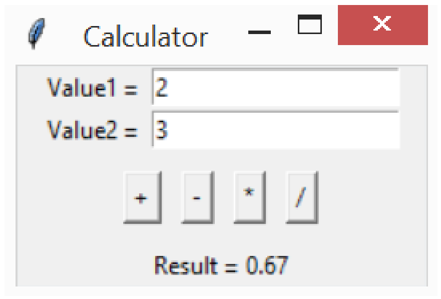
You can't divide by zero!


Calculator

Value1 = 2

Value2 = qwert

Value error


Calculator

Value1 = 2

Value2 = 3

Result = 0.67

## Simple calculator - solution

- The code for the GUI

```
from tkinter import *
class Calculator:
    def __init__(self,wdw):
        self.frame1 = Frame(wdw)
        self.frame1.pack()
        Label(self.frame1, text = 'Value1 = ').grid(row = 0, column = 0)
        self.entry1 = Entry(self.frame1)
        self.entry1.grid(row = 0, column = 1)
        self.label2 = Label(self.frame1, text = 'Value2 = ')
        self.label2.grid(row = 1, column = 0)
        self.entry2 = Entry(self.frame1)
        self.entry2.grid(row = 1, column = 1)
        self.frame2 = Frame(wdw)
        self.frame2.pack()
        self.frame3 = Frame(wdw)
        self.frame3.pack()
        self.label3 = Label(self.frame3, text = '')
        self.label3.grid(row = 2, column = 0)
        b=0
        for i in ['+','-','*','/']:
            Button(self.frame2, text=i, command =
                       lambda a=i: self.calc(a)).grid(row=3, column=b, padx=5, pady=10)
            b += 1
```

# Simple calculator - solution

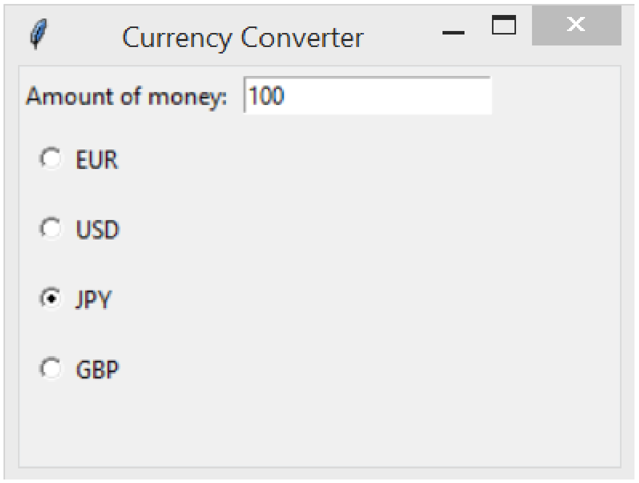
- The code for the GUI

```
    def calc(self,op):
        try:
            e1=float(self.entry1.get())
            e2=float(self.entry2.get())
            if op=='+':
                result = e1 + e2
            elif op=='-':
                result = e1 - e2
            elif op=='*':
                result = e1 * e2
            elif op=='/':
                result = e1 / e2
            self.label3['text']='Result = '+ format(result,'0.2f')
        except ValueError:
            self.label3['text']='Value error'
        except ZeroDivisionError:
            self.label3['text']="You can't divide by zero!"
```

```
def main():
    t = Tk()
    t.title('Calculator')
    c = Calculator(t)
    t.mainloop()

main()
```

# Currency Conventer

- Widgets used in the app:
  - 1 Label widtet: `Amount of money:`
  - 1 Entry widtet - for the value of `e1`, after `Amount of money:`
  - A group of Radiobutton widgets, with the text `EUR, USD, JPY, GBP`
  - Clicking on a Radiobutton will trigger the callback function `fun()` to calculate a result and display information by means of messagebox.



Currency Converter

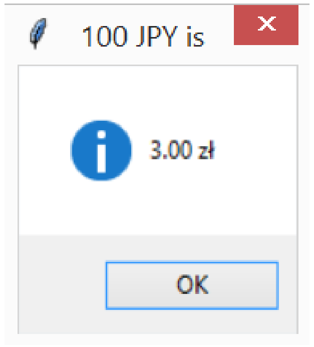
Amount of money: 100

[ ] EUR

[ ] USD

[✓ JPY]

[ ] GBP



100 JPY is

3.00 zł

# Currency Converter - solution

- The code for the GUI

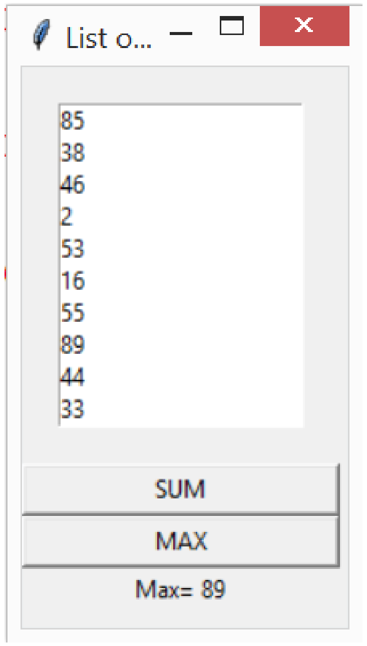
```python
from tkinter import *
from tkinter import messagebox
from random import randint

class Rb:
    def __init__(self,root):
        label=Label(root,text='Amount of money:')
        label.grid(row = 0, column = 0)
        e1 = IntVar()
        e1.set('')
        entry1 = Entry(root, textvariable=e1)
        entry1.grid(row = 0, column = 1, padx = 5, pady = 5, sticky = W)
        self.var = IntVar()
        for text, value in [('EUR',1),('USD',2),('JPY',3),('GBP',4)]:
            Radiobutton(root,text=text, value=value, variable=self.var,
                        command = lambda a=e1,b=text: self.fun(a,b)).grid(
                            row = value, column = 0, padx = 5, pady = 5, sticky = W)
```

# Currency Converter - solution

- The code for the GUI

```
    def fun(self,am,t):
        i = self.var.get()
        if(i==1):
            zl=4.29
        elif (i==2):
            zl=3.74
        elif (i==3):
            zl=0.03
        else:
            zl=4.78
        messagebox.showinfo(str(am.get())+' '+t+' is ',format(zl*am.get(),'0.2f')+' zł')

def main():
    t = Tk()
    t.title('Currency Converter')
    t.geometry("300x200")
    rb = Rb(t)
    t.mainloop()

main()
```

# List of Numbers



85
38
46
2
53
16
55
89
44
33

Max= 89

- Widgets used in the app:
  - 1 Label widtet: `Sum =` or `Max =`
  - 1 Listbox - for 10 of random values (0-99)
  - 2 Buttons widgets, with the text `SUM`, `MAX` respectively
  - Clicking on a button will trigger a callback function `sum()` or `max()` to calculate a result and display information by means of messagebox.

# List of Numbers - solution

- The code for the GUI

```python
from tkinter import *
from random import randint

class Sb:
    def __init__(self,root):
        lb = Listbox(root)
        lb.grid(row=0,column=0,padx = 17, pady = 17)
        for item in range(10):
            lb.insert(END, randint(0,99))
        btn1 = Button(root, text='SUM', command = lambda a=lb: self.sum(a))
        btn1.grid(row=1,column=0, sticky=W+E)
        btn2 = Button(root, text='MAX', command = lambda a=lb: self.max(a))
        btn2.grid(row=2,column=0, sticky=W+E)
        self.label = Label(root,text='')
        self.label.grid(row=3,column=0)

    def sum(self,l):
        s = 0
        for item in range(0,l.size()):
            s += l.get(item)
        self.label['text']='Sum= '+str(s)
```

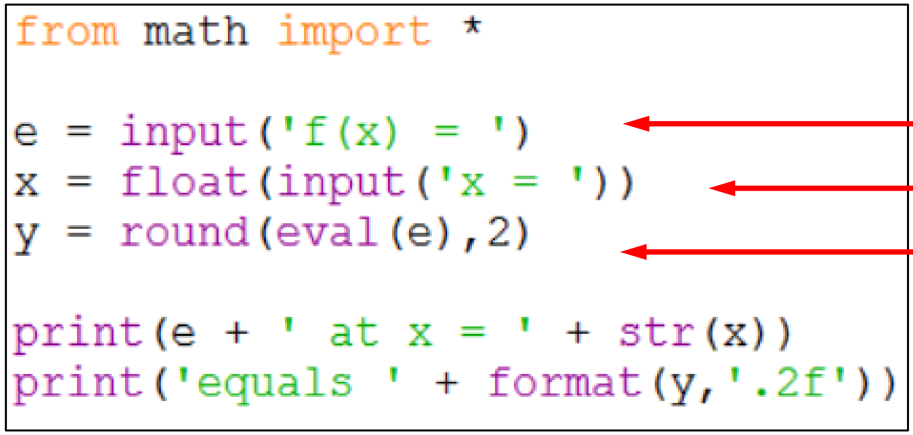
# List of Numbers - solution

- The code for the GUI

```
    def max(self,l):
        m1 = l.get(0)
        for item in range(1,l.size()):
            m2 = l.get(item)
            if m1<m2:
                m1 = m2
        self.label['text']='Max= '+str(m1)

def main():
    t = Tk()
    t.geometry('150x280')
    t.title('List of Numbers')
    sb = Sb(t)
    t.mainloop()

main()
```

# Python’s dynamic typing

- Expressions may be evaluated dynamically via the **eval** command.
- Any valid Python expression the user enters will be evaluated at the given x.



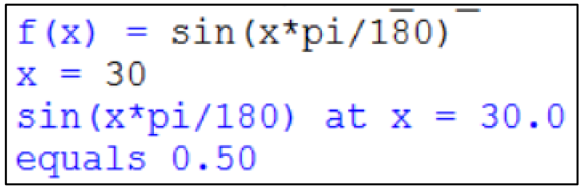
```python
from math import *

e = input('f(x) = ')
x = float(input('x = '))
y = round(eval(e),2)

print(e + ' at x = ' + str(x))
print('equals ' + format(y,'.2f'))
```

read the expression for f(x)

read the value for x

compute the value for the expression f at x

print the result

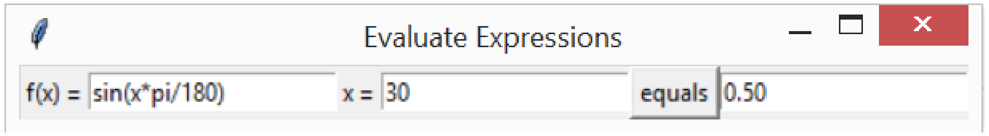


```
f(x) = sin(x*pi/180)
x = 30
sin(x*pi/180) at x = 30.0
equals 0.50
```

# GUI Expression Evaluator

- Widgets used in the GUI app:
- 2 Label widtets: `f(x)=` and `x=`
- 3 Entry widtets:
  - for the expression, after `f(x)=`
  - for the value of `x`, after `x=`
  - and the result, after the `equals`
- 1 Button, with the text `equals`

Clicking on the `equals` will trigger a callback function to evaluate the function `f(x)` at `x`.


Evaluate Expressions

f(x) = sin(x*pi/180) x = 30 equals 0.50

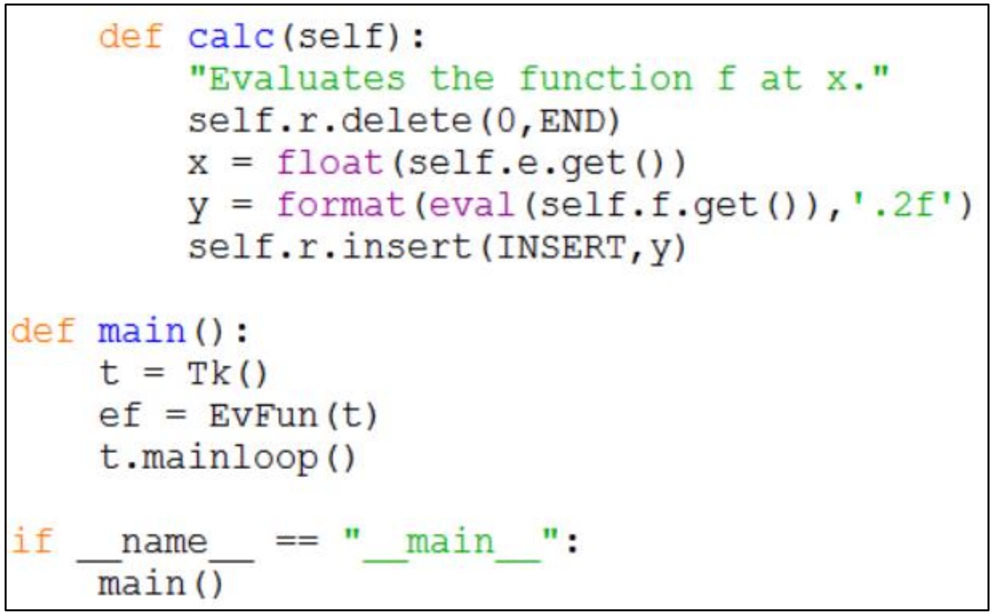
# GUI Expression Evaluator - solution

- The code for the GUI

```
from tkinter import *
from math import *

class EvFun():
    """ GUI to evaluate user given expressions. """
    def __init__(self,wdw):
        "Determines the layout of the GUI."
        wdw.title("Evaluate Expressions")
        self.L1 = Label(wdw,text="f(x) =")
        self.L1.grid(row=0,column=0)
        self.f = Entry(wdw)
        self.f.grid(row=0,column=1)
        self.L2 = Label(wdw,text="x =")
        self.L2.grid(row=0,column=2)
        self.e = Entry(wdw)
        self.e.grid(row=0,column=3)
        self.b1 = Button(wdw,text="equals",command=self.calc)
        self.b1.grid(row=0,column=4)
        self.r = Entry(wdw)
        self.r.grid(row=0,column=5)
```

# GUI Expression Evaluator - solution

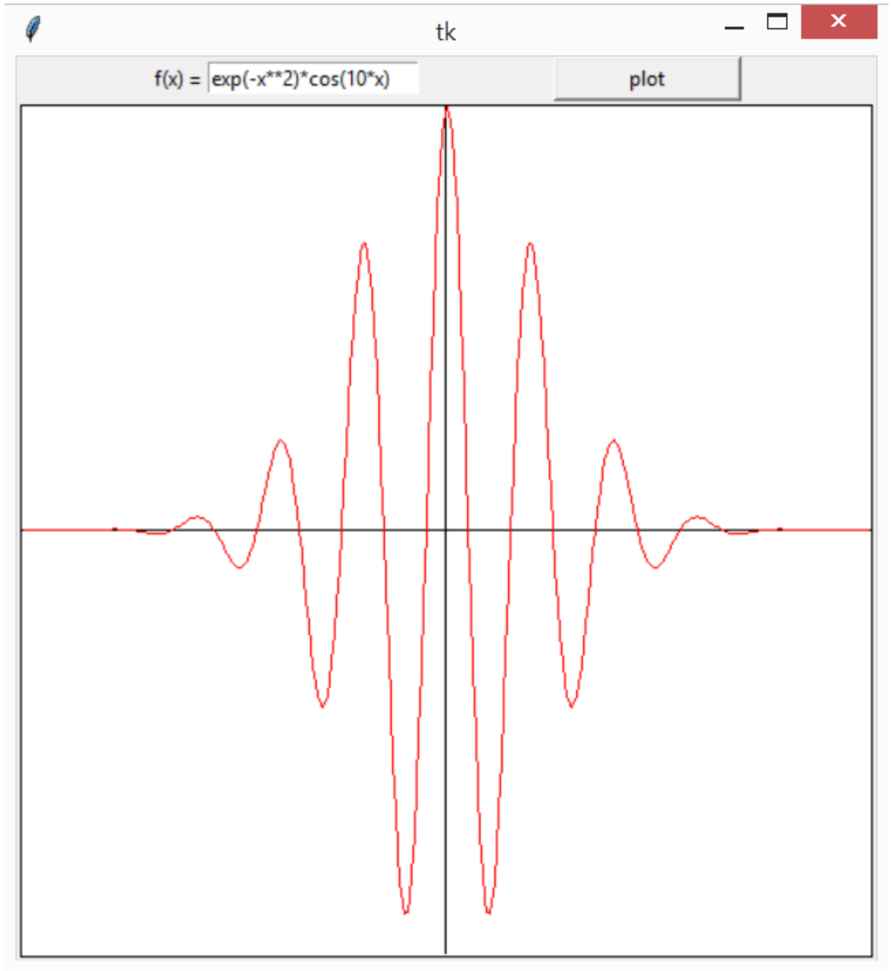
- Code for callback and main functions



```python
    def calc(self):
        "Evaluates the function f at x."
        self.r.delete(0,END)
        x = float(self.e.get())
        y = format(eval(self.f.get()),'.2f')
        self.r.insert(INSERT,y)

def main():
    t = Tk()
    ef = EvFun(t)
    t.mainloop()

if __name__ == "__main__":
    main()
```

## Graphing Expressions

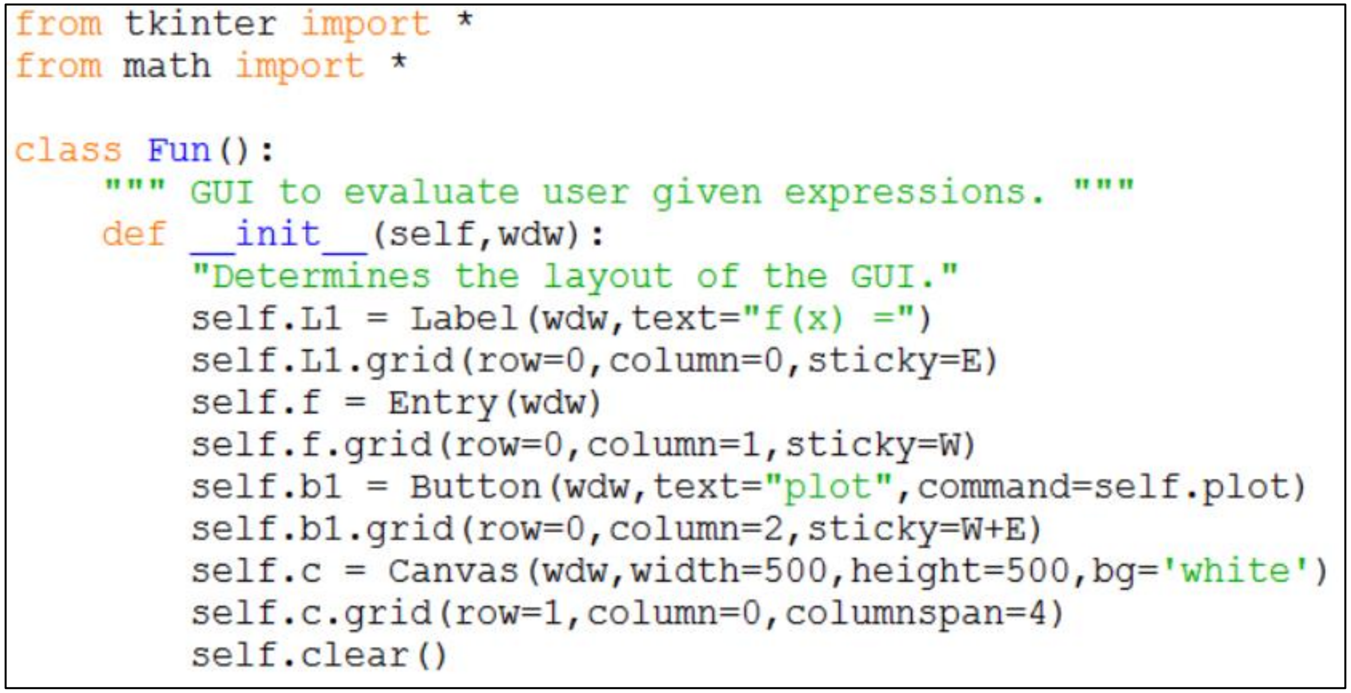
- Using the Canvas widget.



```
exp(-x**2)*cos(10*x)
```

## Graphing Expressions - solution

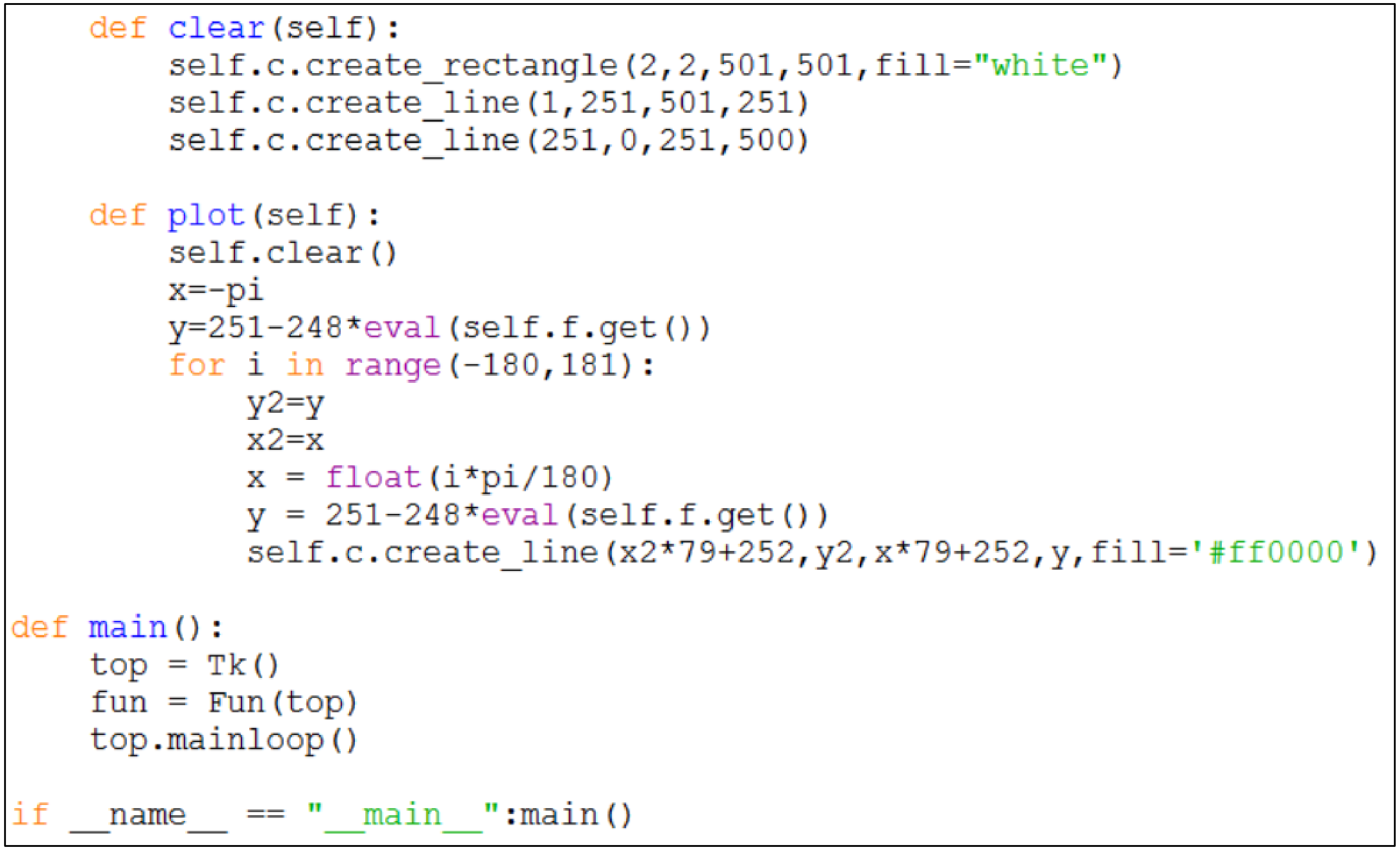
- The code for the GUI



```python
from tkinter import *
from math import *

class Fun():
    """ GUI to evaluate user given expressions. """
    def __init__(self,wdw):
        "Determines the layout of the GUI."
        self.L1 = Label(wdw,text="f(x) =")
        self.L1.grid(row=0,column=0,sticky=E)
        self.f = Entry(wdw)
        self.f.grid(row=0,column=1,sticky=W)
        self.b1 = Button(wdw,text="plot",command=self.plot)
        self.b1.grid(row=0,column=2,sticky=W+E)
        self.c = Canvas(wdw,width=500,height=500,bg='white')
        self.c.grid(row=1,column=0,columnspan=4)
        self.clear()
```

# Graphing Expressions - solution

- Code for callback (clear, plot) and main functions



```python
    def clear(self):
        self.c.create_rectangle(2,2,501,501,fill="white")
        self.c.create_line(1,251,501,251)
        self.c.create_line(251,0,251,500)

    def plot(self):
        self.clear()
        x=-pi
        y=251-248*eval(self.f.get())
        for i in range(-180,181):
            y2=y
            x2=x
            x = float(i*pi/180)
            y = 251-248*eval(self.f.get())
            self.c.create_line(x2*79+252,y2,x*79+252,y,fill='#ff0000')

def main():
    top = Tk()
    fun = Fun(top)
    top.mainloop()

if __name__ == "__main__":main()
```