Calculette – manip1

Objectifs de la manip :

- Comprendre, analyser et adapter un programme de demo
- Placement de boutons dans une fenêtre Tkinter avec le manager grid()
- Placement de labels dans une fenêtre et association à une variable
- Gestion de problèmes algorithmiques avec une programmation événementielle
- Mise en œuvre de fonctions et utilisation de quelques variables globales

Description des notions utilisées :

- Création d'une fenêtre (éléments utilisés)

```
my_frame = Tk()
my_frame.title("calculette 2 termes")
my_frame.minsize(200, 200)
```

On crée une fenêtre (ici nommée my_frame), on lui donne un titre et une taille minimum. Selon le contenu, la taille peut dépasser la taille minimum.

- Création d'un label ()

```
calcul = StringVar()
Label(textvariable = calcul, height = 2).grid(row=0, column=0, columnspan=4, sticky=W)
calcul.set('')
```

Un label est simplement une zone de texte qui montre une chaine de caractère.
Si on veut que la chaîne de caractère puisse être une variable, on déclare un StringVar() avant et on précise le nom du StringVar() via le paramètre textvariable du label.
Pour placer le label dans la fenêtre on peut utiliser le manager grid() qui va gérer la disposition des éléments dans la fenêtre suivant un tableau (lignes et colonnes).
On peut décider d'étaler l'élément sur plusieurs cases ; columnspan permet d'étaler sur plusieurs colonnes. Sticky = W permet que le texte du label se colle à gauche (ouest = W).

- Création d'un bouton

```
Button(text = ' ', width = 3).grid(row=3, column=0)
```

On peut mettre du texte sur le bouton et préciser sa largeur (width) en caractères.
Il est possible, comme pour le label, d'étaler le bouton sur plusieurs cases.

- Associer une fonction à l'événement click d'un bouton

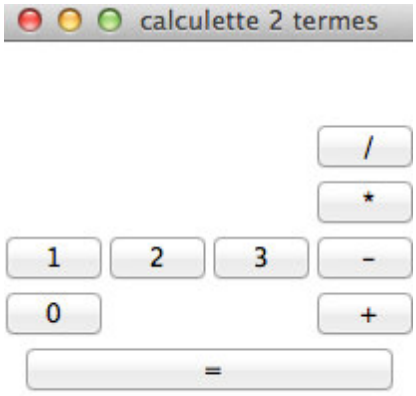
```
Button(text = " = ", width = 20, command = click_egal).grid(row=5, column=0, columnspan=4)
Button(text = STR_ADD, width = 3, command = lambda:click_operateur(STR_ADD)).grid(row=4, column=3)
```

Les deux exemples ci-dessus permettent d'associer une fonction à l'événement click d'un bouton. Le 1er exemple utilise la fonction sans paramètre click_egal(), le 2ème exemple utilise une fonction avec un paramètre. On reconnaît que c'est une fonction avec paramètre grâce au « lambda : » et la valeur du paramètre qui sera passé lors du click est ici la valeur de STR_ADD. Cette façon de faire permet d'avoir une même fonction associée à des clicks de boutons différents.
Cependant, rien n'interdit de faire appel à ces fonctions en dehors d'une routine qui gère un événement.

Programme demo1.py:

```
# calculette 2 termes
# prend en compte des clicks sur boutons
# demo1 pour la 1ere seance

from Tkinter import *

# Les chaines constantes possibles pour la variable operateur
STR_ADD = ' + '
STR_SOUSTR = ' - '
STR_MUL = ' * '
STR_DIV = ' / '

#variables globales
nb1 = ''        #une chaine vide au depart, un nombre ensuite
nb2 = ''        #une chaine vide au depart, un nombre ensuite
operateur = ''  #une chaine vide au depart, une chaine operateur ensuite
egal = ''       #une chaine vide au depart, ' = ' ensuite
result = ''     #une chaine vide au depart, un nombre ensuite

calcul = ''     #sera utilisee comme StringVar() comme variabletext d'un label

#les fonctions
def click_2():
    #lorsqu'on click sur un chiffre
    global nb1, nb2, result
    if operateur == '':         #l'operateur n'est pas introduit
        if nb1 == '':           # -- ces 2 lignes seront utiles
            nb1 = 0             # -- plus tard
        nb1 = 2
        calcul.set(str(nb1))
    else:
        if nb2 == '':           # -- ces 2 lignes seront utiles
            nb2 = 0             # -- plus tard
        nb2 = 2
        calcul.set(str(nb1) + operateur + str(nb2))

def click_operateur(str_operateur):
    #lorsqu'on click sur un operateur
    global operateur
    if egal == '':                          #le calcul n'est pas fini
        if operateur == '' and nb1 != '':   #l'operateur n'est pas introduit et nb1 existe
            operateur = str_operateur
            calcul.set(str(nb1) + operateur)

def click_egal():
    #lorsqu'on click sur egal
    global result, egal
    if nb2 != '':
        egal = ' = '
        if operateur == STR_ADD:
            result = nb1 + nb2
        else:
            result = 999 #a modifier bien sur
        calcul.set(str(nb1) + operateur + str(nb2) + egal + str(result))

#creation de la fenetre
my_frame = Tk()
my_frame.title("calculette 2 termes")
my_frame.minsize(200, 200)

#la 1ere ligne contient un label dont le texte est associe a la variable 'calcul'
calcul = StringVar()
Label(textvariable = calcul, height = 2).grid(row=0, column=0, columnspan=4, sticky=W)
calcul.set('')

#la 2eme ligne
Button(text = STR_DIV, width = 3, command = lambda:click_operateur(STR_DIV)).grid(row=1, column=3)

#la 3eme ligne
Button(text = STR_MUL, width = 3, command = lambda:click_operateur(STR_MUL)).grid(row=2, column=3)

#la 4eme ligne
Button(text = ' ', width = 3).grid(row=3, column=0)
Button(text = "2", width = 3, command = click_2).grid(row=3, column=1)
Button(text = ' ', width = 3).grid(row=3, column=2)
Button(text = STR_SOUSTR, width = 3, command = lambda:click_operateur(STR_SOUSTR)).grid(row=3, column=3)

#la derniere ligne
Button(text = ' ', width = 3).grid(row=4, column=0)
Button(text = STR_ADD, width = 3, command = lambda:click_operateur(STR_ADD)).grid(row=4, column=3)
Button(text = " = ", width = 20, command = click_egal).grid(row=5, column=0, columnspan=4)

#la phase d'attente d'un evenement
my_frame.mainloop()
```

Calculette, énoncé de la manip 1 :



Au départ la fenêtre n'est pas complète, il faudra dans un premier temps la compléter pour avoir une fenêtre comme celle ici à droite.

Il faudra également compléter les créations de boutons et les fonctions associées pour faire fonctionner la calculette avec seulement ces boutons.

On ajoutera les autres boutons par la suite.

1) /20

Modifiez la fonction click_2() pour permettre de constituer un nombre de plusieurs chiffres.

2) /20

Ajouter un test pour qu'on ne puisse pas compléter nb2 une fois que le égal est fait.
Compléter également les boutons 0, 1 et 3 et leurs fonctions respectives.

3) /20

Compléter click_egal() pour que tous les operateurs fonctionnent.

4) /20

Implémenter une fonction click(chiffre) où « chiffre » est le paramètre qui représente le chiffre cliqué.
Dans click_1() faites un simple appel de click(), dans click_2() idem, ...

5) /20

Modifier, lors de la création des boutons 0, 1, 2 et 3, les fonctions associées pour utiliser une même fonction click avec comme paramètre la valeur du chiffre du bouton enfoncé.