Formation I.S.N. 2017-2018

Principales méthodes pour `tkinter`

On rappelle que, pour manipuler un **objet** en **Python**, il faut lui appliquer une **méthode** selon la syntaxe :

```
objet.methode(parametres eventuels)
```

Les méthodes et instructions décrites ci-dessous (*non exhaustives*) sont valables après importation du module **tkinter**.

I – Fenêtre principale

On affecte une nouvelle fenêtre à la variable **parent** avec : `parent = Tk()`.

Méthode(paramètre)	Description
`mainloop()`	Boucle d'attente des événements (*indispensable !*).
`title('titre')`	Titre de la fenêtre (*au format* ***str***).
`geometry('WxH+X+Y')`	Dimensions de la fenêtre (**WxH**) et position (**X+Y**) du coin supérieur gauche par rapport à celui de l'écran.

II – Widgets usuels

1°/ Boutons

La variable intitulée **bouton_nom** contient un bouton appartenant au conteneur **parent**. Un clic sur ce bouton lancera **fonction** :

```
bouton_nom = Button(parent, text='…', command=fonction)
```

Un bouton pourra fermer la fenêtre **fen** si la commande associée est `fen.destroy()`.

2°/ Zone de texte

La variable intitulée **zone_nom** contient une zone de texte (*ou une image*) appartenant au conteneur **parent**.

```
zone_nom = Label(parent, text='Blabla')
```

3°/ Formulaire

La variable intitulée **saisie_nom** contient un formulaire appartenant au conteneur **parent**.

```
saisie_nom = Entry(parent, textvariable = StringVar())
```

On récupère le texte entré en appliquant la méthode `.get()` au formulaire.

StringVar() est un type de variable du module **tkinter** qui permet d'affecter du « texte modifiable » à une variable. Lorsque **StringVar()** est utilisé en dehors d'une création de formulaire, il est **fortement conseillé** d'indiquer la fenêtre principale à l'aide du paramètre **master** (*comme lors d'importation d'images dans un* ***Canvas*** *– voir plus loin*). Par exemple :

```
1 chaine = StringVar(master = parent)
2 saisie_nom = Entry(parent, textvariable = chaine)
```

4°/ Canevas

La variable intitulée **nom** contient un canevas appartenant au conteneur **parent**.

```
nom = Canvas(parent, width = a, height = b)
```

III – Méthodes applicables à tout widget

Méthode(paramètre)	Description
`grid()`	Positionne le widget concerné dans son widget parent selon les numéros de ligne (***row = a***) et de colonne (***column = b***) indiqués en paramètres.
`destroy()`	Supprime un widget (*et tous ses widgets enfants*).
`configure(para)`	Modifie le (*ou les*) paramètre(*s*) désignés du widget.
`bind(evt, fct)`	Au déclenchement de l'événement **evt** dans le widget, la fonction **fct**, d'unique paramètre **event**, se lance.
`after(tps, fct)`	Relance l'action de la fonction **fct** dans le widget toutes les **tps** millisecondes.
`winfo_reqheight()`	Retourne la hauteur actuelle (*en* ***px***) du widget.
`winfo_reqwidth()`	Retourne la largeur actuelle (*en* ***px***) du widget.

IV – Quelques items à tracer dans un canevas

1°/ Segment

Méthode	`.create_line(x0, y0, x1, y1, width=…, fill='…')`
Description	(x_0 ; y_0) et (x_1 ; y_1) sont les coordonnées des points extrémités. **`width`** est un attribut optionnel d'épaisseur du trait, en pixels. **`fill`** est un attribut optionnel donnant la couleur du tracé.

2°/ Rectangle

Méthode	`.create_rectangle(x0, y0, x1, y1)`
Description	(x_0 ; y_0) et (x_1-1 ; y_1-1) sont les coordonnées des sommets haut-gauche et bas-droite opposés du rectangle. **`width`** est un attribut optionnel d'épaisseur du contour, en pixels. **`outline`** est un attribut optionnel donnant la couleur du contour. **`fill`** est un attribut optionnel donnant la couleur du contenu.

3°/ Ovale

Méthode	`.create_oval(x0, y0, x1, y1)`
Description	(x_0 ; y_0) et (x_1-1 ; y_1-1) sont les coordonnées des sommets opposés du rectangle dans lequel est tracé l'ellipse. Les attributs optionnels **`width`**, **`outline`** et **`fill`** sont utilisables de la même manière que pour le rectangle.

4°/ Texte

Méthode	`.create_text(x0, y0, text = 'contenu')`
Description	(x_0 ; y_0) sont les coordonnées de l'**ancre** du texte. **`fill`** est un attribut optionnel donnant la couleur du texte. **`font`** est un attribut optionnel donnant la police suivie de la taille.

5°/ Image au format `.gif`

Chargement	`im = PhotoImage(file = 'mon_image.gif', master=parent)`
Méthode	`.create_image(x0, y0, image = im)`
Description	Le fichier image est atteint avec un lien relatif. (x_0 ; y_0) sont les coordonnées de l'**ancre** de l'image.

V – Méthodes du canevas modifiant les items

Méthode(paramètre)	Description
`delete(nom)`	Supprime l'item donné en paramètre (*par son* ***`nom`****, son identifiant* **`id`** *ou son* **`tag`**).
`delete(ALL)`	Supprime tous les items du canevas.
`itemconfigure(nom, para)`	Modifie le (*ou les*) paramètre(*s*) désigné(*s*) de l'item **`nom`**.
`itemcget(nom, para)`	Retourne la valeur actuelle du paramètre **`para`** de l'item **`nom`**.

VI – Événements déclenchés par la souris

Événement	Description
`'<Motion>'`	Mouvement de la souris à l'intérieur du widget.
`'<Button-1>'`	Clic (*enfoncement*) du bouton gauche (`1`) ou droit (`3`).
`'<ButtonRelease-3>'`	Relâchement du bouton gauche (`1`) ou droit (`3`).
`'<Enter>'`	La souris passe au-dessus du widget.
`'<Leave>'`	La souris «sort» du widget.

VII – Événements déclenchés par le clavier

Événement	Description
`'<k>'`	Appui sur la touche **`h`** (*par exemple*).
`'<KeyRelease-h>'`	Relâchement de la touche **`h`** (*par exemple*).

VIII – Animations automatiques

Méthode	`fenetre.after(temps, fonction)`
Description	(*re*)lance l'action de **`fonction`** dans la **`fenetre`** toutes les **`temps`** millisecondes.