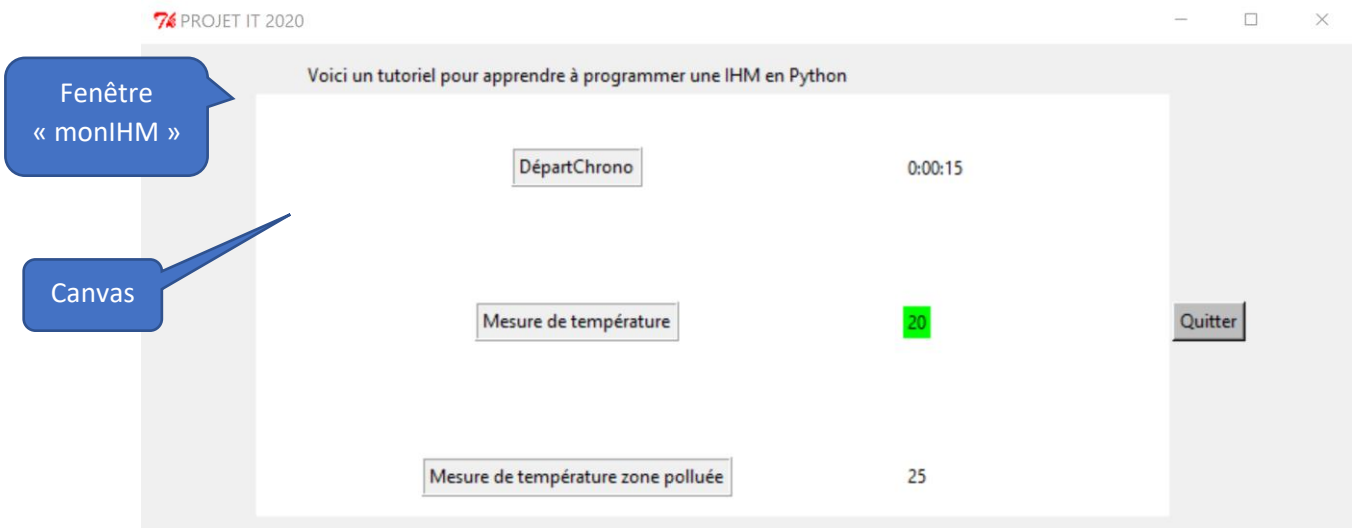
CREATION D'UNE IHM PYTHON POUR RASPBERRY PI

ETAPE 1 : créer le design de l'IHM

- Choisir la disposition des éléments (boutons, étiquettes de texte, Canvas, bouton Quitter)



ETAPE 2 : définir la taille de la grille

Exemple : ici la grille comporte 4 colonnes (column) et 4 lignes (row). Chaque ligne ou colonne est numérotée à partir de 0

Remarque : le bouton Quitter est posé à l'intersection de la ligne n°2 et de la colonne n°3

CREATION D'UNE IHM PYTHON POUR RASPBERRY PI

ETAPE 3 : le code Python

- *Remarque 1 : sous Raspberry PI, utiliser de préférence la version de python 2.7*
- *Remarque 2 : dans le langage python une instruction se termine par un retour à la ligne, il n'y a pas de « ; » contrairement au langage C ou C# utilisé par Arduino.*
- *Python est sensible à l'indentation (décalage par rapport à la marge)*

  Exemple : dans la fonction departChrono ci-dessous, toutes les instructions décalées par rapport à la marge appartiennent à la fonction. Un nouveau def devra donc être positionné directement contre la marge.

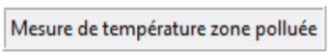
```
def departChrono() :

    # initialisation de la variable start
    global start
    start= default_timer()

    # appel de la fonction chrono()
    chrono()

def ….
```

- les lignes qui commencent par **#** sont des commentaires

Le squelette du programme :

# !/usr/bin/env python2 *# -*- coding: cp1252 -*-* *# Apprenez à programmer une IHM en python* *# Nathalie CALAS-CADEVILLE* *# importation de la bibliotheque tkinter* from Tkinter import * from timeit import default_timer *# bibliotheque du capteur DHT11* #import Adafruit_DHT	**Partie 1 : Importation des bibliothèques** Tkinter permet de bénéficier de méthodes graphiques Enlever le # pour valider l'import de la bibliothèque
####### Definition des fonctions ############### *# fonction chronomètre appelée par le bouton DépartChrono* def departChrono() : *# initialisation de la variable start* global start start= default_timer() *# appel de la fonction chrono()* chrono()	**Partie 8 : Création des fonctions appelées par les boutons et des fonctions diverses nécessaires au bon fonctionnement du programme** DépartChrono Fonction de chronométrage Default_timer() : renvoie le nombre de secondes depuis le démarrage de l'application

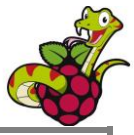
```
def chrono():
   # nombre de secondes
   now=default_timer()-start

   # calcul des minutes et des heures à partir du nombre de secondes
   minutes,seconds =divmod(now,60)
   hours, minutes=divmod(minutes,60)
   str_time="%d:%02d:%02d" % (hours, minutes, seconds)

   # création d'une étiquette chronomètre au format H :MM :SS
   text_clock=Label(monCadre, text=str_time, bg="white")
   text_clock.grid(row=1,column=1,sticky=W)
   monIHM.after(1000,chrono)

# fonction de lecture de la température ambiante appelée par le bouton
# « Mesure de température » :
def tempA():
   # sensor = type de capteur
   sensor=11
   # pin est le numero de GPIO
   pin=23
   # lecture du capteur de temperature et humidite DHT11
   #humidity,tempamb=Adafruit_DHT.read_retry(sensor,pin)
   Tamb=20
   # Placer du texte dans le cadre de l'IHM
   tempAmb=Label(monCadre, text=Tamb, bg="green")
   tempAmb.grid(row=2,column=1)

# fonction de lecture de la température sur la zone polluée appelée par
# le bouton "Mesure de température zone polluée" :
def tempZ():
   # sensor = type de capteur
   sensor=11
   # pin est le numero de GPIO
   pin=23
   # lecture du capteur de temperature et humidite DHT11
   #humidity,tempamb=Adafruit_DHT.read_retry(sensor,pin)
   TZ=25
   # Placer du texte dans le cadre de l'IHM
   tempZ=Label(monCadre, text=TZ, bg="white")
   tempZ.grid(row=3,column=1)
```




```
#Création d'une fenetre graphique nommée monIHM
monIHM=Tk()
monIHM.geometry('800x320')
# Titre de la fenetre
monIHM.title("PROJET IT 2020")
```

Partie 2 : Création d'une fenêtre graphique

Nom de la fenêtre : monIHM

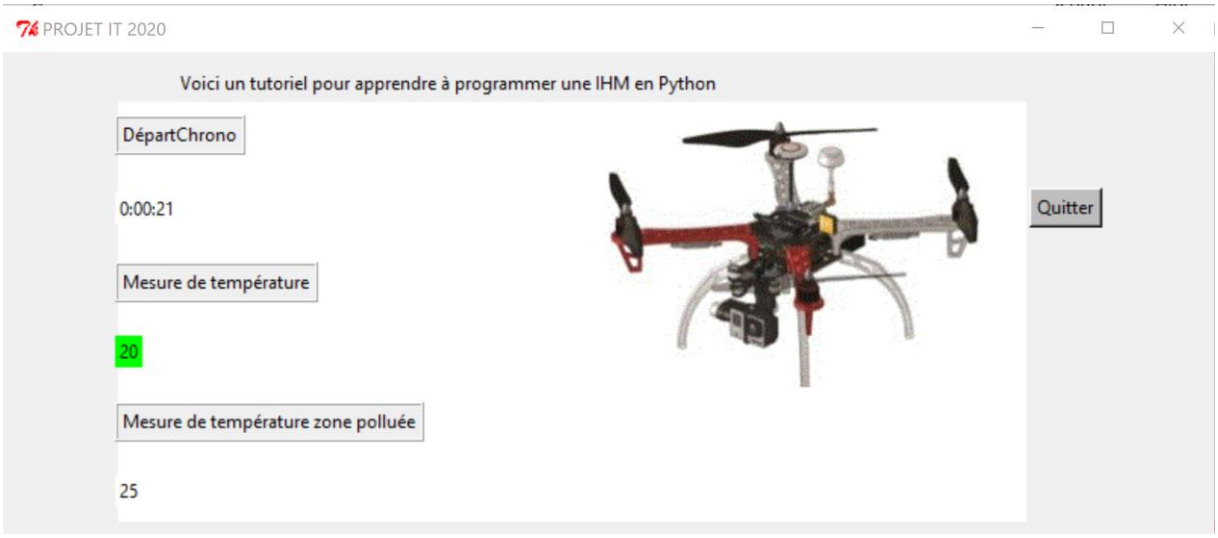


#Définition des dimensions du cadre de la fenêtre *# hauteur=500 largeur=400* monCadre=Frame(monIHM,height=300, width=800) monCadre.pack_propagate(0) *# bordures padx et pady* monCadre.pack(padx=10, pady=10)	**Partie 4 : Création d'un cadre dans la fenêtre graphique** Nom du cadre : monCadre padx et pady permettent de fixer la largeur des bordures du cadre
# Création d'un Canvas. Ce widget permet de faire du dessin canv=Canvas(monCadre, height=300, width =600, bg="white") *# fixer le canvas à la position souhaitée* canv.grid(row=1,column=0, rowspan=3, columnspan=3)	**Partie 5 : Création d'un Canvas** - Canv est un Canvas qui appartient à monCadre et qui a pour couleur de fond « white » - Position du Canvas : intersection de la ligne 1 (row=1) et de la colonne 0 (Column=0) - Rowspan=3 : étendre le Canvas sur 3 lignes - Columnspan =3 : étendre le Canvas sur 3 colonnes
# Placer du texte dans le cadre de l'IHM monTexte=Label(monCadre, text="Voici un tutoriel pour apprendre à programmer une IHM en Python") monTexte.grid(row=0,column=0)	**Partie 6 : Création d'un texte de type Label (étiquette)** Placement du texte à l'intersection de la ligne 0 et de la colonne 0 (en haut du cadre, au-dessus du Canvas)
# Création du bouton de lancement du chronomètre boutonChrono=**Button**(monCadre, text="DépartChrono", relief=RIDGE, cursor="clock",command=**departChrono**) boutonChrono.grid(row=1,column=0)	**Partie 7 : Création des boutons** La méthode **Button** permet de créer un bouton

# Création du bouton de mesure de température boutonTemperature=**Button**(monCadre, text="Mesure de température", relief=RIDGE, cursor="clock",command=**tempA**) boutonTemperature.grid(row=2,column=0) *# Création du bouton de mesure de température* boutonTempZone=**Button**(monCadre, text="Mesure de température zone polluée", relief=RIDGE, cursor="clock",command=**tempZ**) boutonTempZone.grid(row=3,column=0) *# Création du bouton permettant de quitter l'application* boutonQuitter=**Button**(monCadre, bg="grey", text="Quitter", cursor="pirate", command=**monIHM.destroy**) boutonQuitter.grid(row=2,column=3)	Command =nom de la fonction appelée lorsque le bouton est activé Le bouton Quitter appelle la méthode destroy appliquée à monIHM (monIHM.destroy) Ceci permet de fermer proprement l'application.

# Démarrage de l'IHM 	
# mainloop est la boucle principale qui permet à monIHM de continuer # à fonctionner et d'être mise à jour sans sortir du programme 	
monIHM.mainloop()
	Partie 3: commande de démarrage de l'application

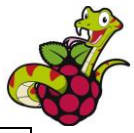
VERSION AVEC ILLUSTRATION : le code Python



En rouge : les modifications de la grille précédente (lignes et colonnes) et l'insertion d'une image

```
# !/usr/bin/env python2
# -*- coding: cp1252 -*-
# Apprenez à programmer une IHM en python
# Nathalie CALAS-CADEVILLE


# importation de la bibliotheque tkinter
from Tkinter import *
from timeit import default_timer
# bibliotheque du capteur DHT11
#import Adafruit_DHT


# Definition des fonctions


def departChrono() :
    global start
    start= default_timer()
    chrono()

def chrono():
    now=default_timer()-start
    minutes,seconds =divmod(now,60)
    hours, minutes=divmod(minutes,60)
    str_time="%d:%02d:%02d" % (hours, minutes, seconds)
    text_clock=Label(monCadre, text=str_time, bg="white")
    text_clock.grid(row=2,column=0,columnspan=3, sticky='w')
    monIHM.after(1000,chrono)
```

```
def tempA():
    # sensor = type de capteur
    sensor=11
    # pin est le numero de GPIO
    pin=23
    # lecture du capteur de temperature et humidite DHT11
    #humidity,tempamb=Adafruit_DHT.read_retry(sensor,pin)
    Tamb=20
    # Placer du texte dans le cadre de l'IHM
    tempAmb=Label(monCadre, text=Tamb, bg="green")
    tempAmb.grid(row=4,column=0,columnspan=3, sticky='w')

def tempZ():
    # sensor = type de capteur
    sensor=11
    # pin est le numero de GPIO
    pin=23
    # lecture du capteur de temperature et humidite DHT11
    #humidity,tempamb=Adafruit_DHT.read_retry(sensor,pin)
    TZ=25
    # Placer du texte dans le cadre de l'IHM
    tempZ=Label(monCadre, text=TZ, bg="white")

    tempZ.grid(row=6,column=0,columnspan=3, sticky='w')



#Création d'une fenetre graphique nommée monIHM
monIHM=Tk()
monIHM.geometry('800x320')
# Titre de la fenetre
monIHM.title("PROJET IT 2020")



#Définition des dimensions du cadre de la fenetre hauteur=500 largeur=400
monCadre=Frame(monIHM,height=300, width=800)
monCadre.pack_propagate(0)
# bordures padx et pady
monCadre.pack(padx=10, pady=10)

# Création d'un Canvas. Ce widget permet de faire du dessin
canv=Canvas(monCadre, height=300, width =600, bg="white")
# fixer le canvas à la position souhaitée
canv.grid(row=1,column=0, rowspan=7, columnspan=3)
```



```
# Placer du texte dans le cadre de l'IHM
monTexte=Label(monCadre, text="Voici un tutoriel pour apprendre à programmer une IHM en Python")
monTexte.grid(row=0,column=0)

# creation de la variable de stockage de l'image
fichier=PhotoImage(file="drone.gif")
imageDrone=canv.create_image(450,100,image= fichier)
# placer l'image au premier plan
canv.tag_raise(imageDrone)

# Création du bouton de lancement du chronomètre
boutonChrono=Button(monCadre, text="DépartChrono", relief=RIDGE,
cursor="clock",command=departChrono)
boutonChrono.grid(row=1,column=0,sticky=W)

# Création du bouton de mesure de température
boutonTemperature=Button(monCadre, text="Mesure de température", relief=RIDGE,
cursor="clock",command=tempA)
boutonTemperature.grid(row=3,column=0,sticky=W)

# Création du bouton de mesure de température
boutonTempZone=Button(monCadre, text="Mesure de température zone polluée", relief=RIDGE,
cursor="clock",command=tempZ)
boutonTempZone.grid(row=5,column=0, sticky=W)

# Création du bouton permettant de quitter l'application
boutonQuitter=Button(monCadre, bg="grey", text="Quitter", cursor="pirate",
command=monIHM.destroy)
boutonQuitter.grid(row=2,column=3)

# Démarrage de l'IHM
# mainloop est la boucle principale qui permet à monIHM de continuer à fonctionner et d'être mise à jour sans sortir du programme
monIHM.mainloop()
```