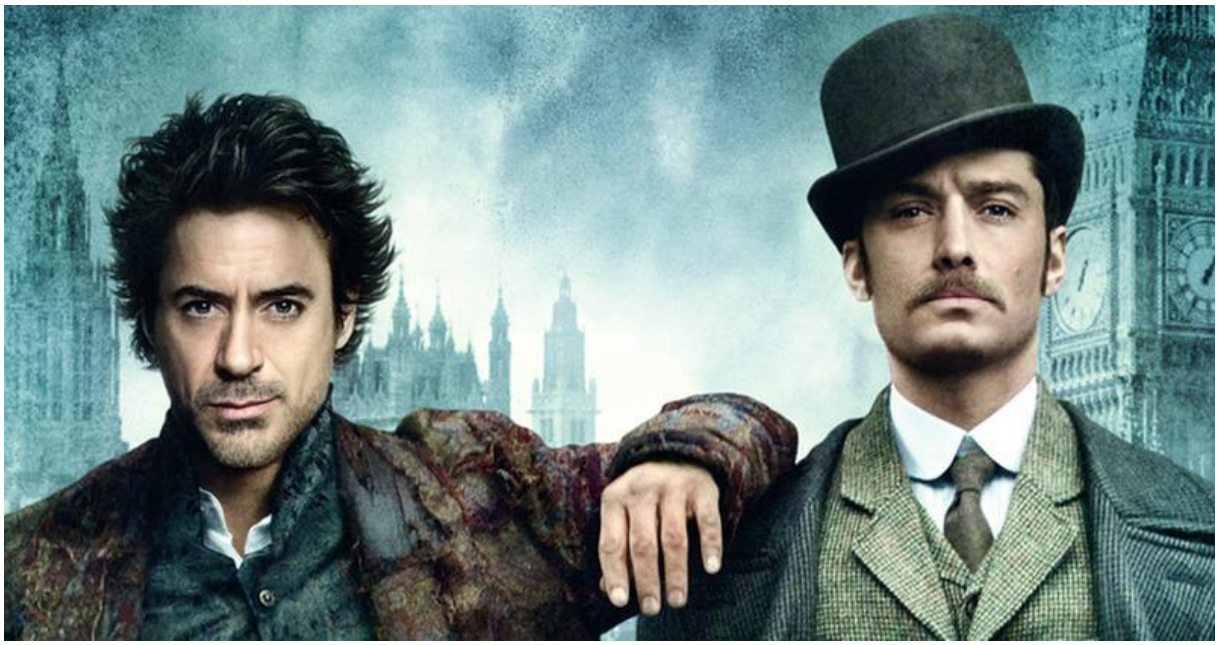
WatSnap

version 2.1

DIVERSITY by EPITECH

I. Introduction

M.Holmes et M.Watson mène une enquête, lors de celle-ci M.Watson a récupéré quelques clichés et souhaiterait les montrer à M.Holmes. Ce dernier étant très occupé il n'a que très peu de temps pour les visionner. C'est pour cela que M.Watson va vous guider dans la réalisation d'un logiciel en Python permettant de transformer un ensemble d'images en GIF.



M.Holmes et M.Watson côte à côte

Le Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Pour ce projet, il vous permettra d'avoir accès facilement à de nombreuses fonctionnalités intégrées au langage.

II. Consignes

* Vous devez installer Python et ses dépendances, pour cela veuillez tout d'abord suivre le tutoriel « Installations Python et ses outils ». Il y a d'autre dépendances à installer, pour cela rendez-vous dans la partie suivante.
* Pour ce projet-là, il vous est demandé de choisir comme nom du repository : Snap_cc. Si vous ne savez pas utiliser git allez lire le tutoriel « Le coffre a jouet du petit git ».
* En cas de soucis ou questions demandez de l'aide à votre voisin de droite. Puis de gauche. Ou inversement. Puis demandez enfin à un cobra si vous êtes bloqué(e).

III. Installations

Avant que Watson ne vous guide dans votre objectif, il vous demande d'installer le nécessaire pour pouvoir commencer.

Vous devez tout d'abord installer "ImageMagik", c'est un package permettant de transformer les images en gif. Il faudra ensuite installer "cv2", une librairie permettant de gérer la vidéo caméra. Puis vous devez installer "numpy" pour les calculs mathématiques complexes tel que les matrices. Pour finir, il faut installer "PIL" qui est une librairie permettant de gérer les images.

a. Pour Linux

Pour pouvoir installer "ImageMagik", suivez les consignes décrites dans ce lien.

Voici ci-dessous les commandes à exécuter pour installer les dépendances Python nécessaires pour Linux :

```
pip3 install numpy
pip3 install scikit-build
sudo apt-get install build-essential cmake unzip pkg-config
sudo apt-get install python3-pil.image
pip3 install opencv-python
sudo python3 –m pip install pillow
```

b. Pour Windows

Dans un premier temps vous pouvez installer "ImageMagik", pour ce faire télécharger l'exécutable ici et lancez-le pour poursuivre l'installation. Par la suite vous devez installer les dépendances de Python. Vous pouvez ouvrir une console et lancer les commandes suivantes :

```
python3 -m pip install opencv-python
python3 -m pip install numpy
python3 -m pip install pillow
```

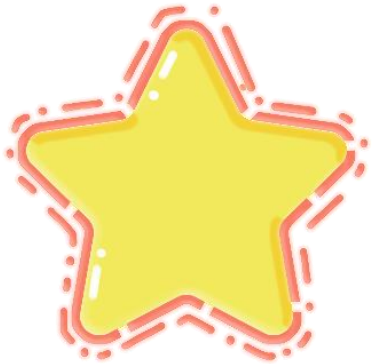
c. Pour MacOS

Dans un premier temps vous installez "ImageMagik", pour ce faire taper ces deux lignes de commande dans une console :

```
brew install imagemagick
brew install ghostscript
```

Ensuite, vous avez besoin d'installer les dépendances Python, voici les commandes à exécuter :

```
python3 -m pip install opencv-python
python3 -m pip install python-numpy
python3 -m pip install pillow
```



Génial, maintenant vous êtes prêt ou prête à enquêter !

IV. Accomplissez la première mission avec M.Watson

Dans un premier temps, M.Watson vous demande de créer un fichier « Snap.py » dans lequel vous allez programmer votre logiciel.

Pour ce faire commencer par ouvrir votre fichier.

a. Découvrez les bibliothèques pour obtenir des indices

Les bibliothèques sont des banques de fonctions spécifiques à une utilisation. Pour les inclure dans un programme, on utilise le mot clé « import » suivi du nom de la bibliothèque. Pour ce programme on utilise les bibliothèques suivantes qui ne sont pas toutes installées de base avec votre Python. C'est pour ça que vous les avez installées préalablement.

Voici ce que vous devez avoir dans les premières lignes de votre fichier :

```
import tkinter as tk
import cv2
import numpy as np
import time
from PIL import Image, ImageTk
import os
```

Maintenant que vous avez importé les bibliothèques dans votre programme essayons de comprendre à quoi elles servent :

- « time », permet de gérer le temps.
- « Operating System», os est une bibliothèque liée à votre système d'exploitation (SE ou OS en anglais).
- Vous avez du installer et voir « tkinter » via le tutoriel « Installations Python et ses outils »
- En ce qui concerne « cv2 », « numpy » et « PIL » vous pouvez retrouver les explications précédemment.

b. Tester son logiciel

Durant l'activité vous devrez vérifier le comportement de votre programme. Donc à chaque étape vous auriez la possibilité de le lancer. Plus précisément, vous devez ouvrir une console là où se trouve votre fichier et rentrer la commande suivante :

```
python3 Snap.py
```

V. Récupération des premiers clichés pris par M.Watson

a. Il est temps d'encadrer ces images

Vous allez donner ici la hauteur "height" et la largeur "width" de votre caméra. Puis, en utilisant la bibliothèque « cv2 », vous allez récupérer la vidéo pour lui assigner ses paramètres.

```
#definition de la camera
width, height = 800, 600
cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, width)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, height)
```

Ensuite, vous allez créer puis paramétrer la fenêtre "root" :

```
#definition de la fenêtre
root = tk.Tk()
root.bind("<Escape>", lambda e: root.quit())
root.attributes("-fullscreen",True)
```

Ici le choix s'est porté sur la touche échap du clavier pour quitter la fenêtre et terminer le programme. Enfin, vous devez choisir d'activer le mode plein écran de votre fenêtre.

b. Visualisez les images afin de récupérer des preuves

Une fois cette étape terminée, vous allez pouvoir rajouter une partie de code qui va mettre l'image au centre de la fenêtre.

```
#ajout de l'image dans la fenêtre
lmain = tk.Label(root)
lmain.pack(fill="none", expand=True)
show_frame()
```

La dernière ligne ci-dessus désigne un appel à une fonction “show_frame” n’appartenant pas à une librairie. En effet, vous allez la créer ! Elle contiendra une partie de votre code correspondant à une action précise (ici à afficher la vidéo dans la fenêtre). Découper son code en différentes fonctions permet de le structurer et de lui donner la lisibilité. De plus cela va vous permettre de réutiliser cette partie de code sans avoir à l’écrire plusieurs fois !

Pour l’instant, elle s’appelle « show_frame » mais vous pouvez lui donner le nom que vous voulez.

⚠ **Attention** : Pour définir une fonction en Python, il faut toujours la placer plus haut que la ligne ou la fonction d’où vous l’appeler, sinon votre programme ne sera pas capable de la trouver au moment où il en a besoin. N’oubliez pas d’indenter les lignes qui lui appartiennent avec des tabulations pour bien les séparer du reste du code !

```
def show_frame():
    global pics
    #Récupération de la frame de la caméra
    _, frame = cap.read()
    #Comversion de la frame en image
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    img = Image.fromarray(frame)
    #Conversion de l’image en image tkinter
    imgtk = ImageTk.PhotoImage(image = img)
    #Configuration de la fenêtre
    lmain.imgtk = imgtk
    lmain.configure(image = imgtk)
    #Actualiser la frame toutes les 20ms (changez le nombre de ms
pour un gif plus long !)
    lmain.after(20, show_frame)
```

Comme vous pouvez le voir sur l’image ci-dessus, la fonction doit récupérer l’image de la caméra et la transformer pour la rendre utilisable avec la bibliothèque graphique. Vous pouvez également constater l’intérêt de l’avoir séparée du reste de votre code. En effet, la méthode « after » de la bibliothèque « Tkinter » sur la dernière ligne permet d’actualiser l’image pour avoir une vidéo. Elle prend donc en paramètre la fonction « show_frame » elle-même pour pouvoir effectuer ces conversions à chaque actualisation !

Si vous avez déjà testé le code, vous avez pu remarquer qu’aucune fenêtre ne s’ouvre et que le programme se termine en moins d’une seconde. En réalité la fenêtre s’affiche, mais qu’une seule fois. Vous devez donc l’afficher en continu, pour cela rajoutez cette ligne à la fin de votre code :

```
root.mainloop()
```

Félicitation ! Vous pouvez maintenant commencer à tester régulièrement votre programme !

VI. Ajouter des filtres à vos clichés

Afin de relever plus facilement les preuves dans vos clichés M.Watson vous invite à modifier les images en y appliquant des filtres, il sera alors plus simple de les montrer à M.Holmes.

Avant tout vous déclarez au début de votre code une variable nommée “filter_choice” que vous initialisez à 0, elle permettra à votre programme de savoir quel filtre il doit appliquer. Ensuite, afin de sélectionner celui que vous souhaitez appliquer sur vos photos, il sera nécessaire de rajouter deux boutons : “Filtre suivant” et “Filtre précédent”.

```
button = tk.Button(root, text="Filtre Suivant", command=next_filter)
button.pack(side="right")
```

Ici, le bouton permettant d’appliquer le filtre suivant a été créé juste au-dessus de la méthode “root.mainloop”. C’est à vous de créer celui pour accéder au filtre précédent. Vous allez maintenant coder les fonctions “prev_filter” et “next_filter”, elles vont permettre de modifier la valeur de la variable indiquant quel filtre vous voulez prendre.

```
filter_choice = 0

def prev_filter():
    global filter_choice
    if filter_choice > 0:
        filter_choice -= 1
```

Cette fonction a pour but de sélectionner le filtre précédent. C’est à vous de coder celle permettant la sélection du filtre suivant, sachant qu’actuellement vous avez 4 filtres et qu’elle doit se nommer “next_filter”. Maintenant, afin d’appliquer ces filtres sur vos photos, il vous faut rajouter deux lignes de code à la fonction “show_frame” :

```
def show_frame():
    global pics, filter_choice
    # Recuperation de la frame de la camera
    _, frame = cap.read()
    #Application du filtre
    frame = apply_filter(frame)
    # Conversion de la frame en image
    if filter_choice == 0:
        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    img = Image.fromarray(frame)
    # Conversion de l’image en image tkinter
    imgtk = ImageTk.PhotoImage(image = img)
    # Configuration de la fenêtre
    lmain.imgtk = imgtk
```

```
      lmain.configure(image = imgtk)
      # Actualise la frame toutes les 20ms (changez le nombre de ms
pour un gif plus long)
      lmain.after(20, show_frame)
```

La condition “if filter_choice == 0” permet d’éviter que le filtre de base se superpose avec vos nouveaux filtres ! De plus, on donne à la variable “frame” la valeur de retour de la fonction “apply_filter” que vous allez créer ci-dessous :

```
def apply_filter(frame):
      if filter_choice ==  1:
            # Application du filtre 1 sur la frame
            frame = cv2.filter2D(frame, -1, kernel_5x5)
      elif filter_choice == 2:
            # Application du filtre 2 sur la frame
            gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
            frame = cv2.Laplacian(gray, cv2.CV_8U, gray, ksize=15)
      elif filter_choice == 3:
            # Application du filtre 3 sur la frame
            frame = max_rgb_filter(frame)
      return frame
```

Chaque condition de cette fonction applique un filtre différent. Vous pouvez en ajouter en bonus en cherchant un peu sur internet ! Pour le premier filtre “filter_choice == 1” vous devez initialiser une variable kernel_5x5 :

```
def apply_filter(frame):
      kernel_5x5 = np.array ([
      [-1, -1, -1, -1, -1],
      [-1,  1,  2,  1, -1],
      [-1, 2, 4, 2, -1],
      [-1, 1, 2, 1, -1],
      [-1, -1, -1, -1, -1]
      ])
```

Sa valeur étant un peu compliquée à comprendre, vous devrez seulement retenir qu’elle est nécessaire à cette méthode de filtre. Pour le filtre numéro 4 “filter_choice == 3”, on doit créer la fonction “max_rgb_filter” qui va appliquer des effets sur l’image :

```
def max_rgb_filter(frame):

      (B, G, R) = cv2.split(frame)

      M = np.maximum(np.maximu(R, G), B)
```

```
    R[R < M] = 0
    G[G < M] = 0
    B[B < M] = 0

    return cv2.merge([B, G, R])
```

Et voilà vous pouvez désormais appliquer 3 filtres sur vos photos !

BRAVO, vous pouvez passer

à votre dernière mission !

VII. Transformer vos plus beaux clichés en GIF

Pour que Holmes puisse visionner rapidement vos clichés vous devez les transformer en GIF. Vous allez donc commencer par ajouter un bouton pour vous permettre de prendre la photo. Pour cela, vous allez réutiliser la fonction “tk.Button” de la librairie graphique “tkinter”. On y ajoute ici le texte que l’on veut sur le bouton, ici, ‘Prendre le gif’ puis vous le positionnez dans la partie “top” de la fenêtre.

```
# Ajout du boutton pour prendre une photo
button3 = tk.Button(root, text="Prendre le GIF", command=take_gif)
button3.pack(side="top")
```

On peut voir que le dernier paramètre de la fonction “tk.Button” fait appel à une fonction que vous devez créer “take_gif” dans le cas où le bouton est sélectionné. Comme son nom l’indique, cette fonction va permettre de lancer la prise de photo.

```
pics = 0

def take_gif():
     global pics
     pics = 1
```

Par la suite, modifiez la fonction “show_frame” pour qu’elle transforme les images en gif.

```
def show_frame():
     global pics, filter_choice
     # Recuperation de la frame de la camera
     _, frame = cap.read()
     #Application du filtre
     frame = apply_filter(frame)
     make_gif(frame)
     # Conversion de la frame en image
     if filter_choice == 0:
     frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
     img = Image.fromarray(frame)
     # Conversion de l’image en image tkinter
     imgtk = ImageTk.PhotoImage(image = img)
     # Configuration de la fenêtre
     lmain.imgtk = imgtk
     lmain.configure(image = imgtk)
     # Actualise la frame toutes les 20ms (changez le nombre de ms
pour un gif plus long)
     lmain.after(20, show_frame)
```

La fonction “make_gif(frame)” ressemble à ceci :

```
def make_gif(frame):
     global pics
     if pics > 0:
          if pics % 2 == 0:
               cv2.imwrite("myimg" + str(pics)+ ".png", frame)
          pics += 1
     if pics == 24:
          os.system("convert -delay 10 myimg*.png mygif.gif")
          os.system("rm *.png")
          pics = 0
```

Vous pouvez maintenant essayer de créer votre premier gif ! Si vous y êtes arrivé, félicitations ! Vous avez réussi votre Snap !

Félicitation, grâce à vous

M.Holmes peut résoudre son enquête !

Vous pouvez essayer d’aller plus loin en cherchant de nouveaux filtres !