OUISNAP

Découverte du **Python**

SOMMAIRE

Niveau 0 - BASE PYTHON

1. Qu'est-ce que le Python ?

Python est un langage puissant, à la fois facile à apprendre et riche en possibilités. Dès l'instant où vous l'installez sur votre ordinateur, vous disposez de nombreuses fonctionnalités intégrées au langage.

2. Les librairies

Les librairies sont des banques de fonctions spécifiques à une utilisation.

Elles ne sont pas toutes installées de base ! Pour pouvoir faire fonctionner votre programme, il va d'abord falloir exécuter les commandes qui sont en parenthèses situer un peu plus loin. (Elles sont souvent de la forme "`apt install` *`nom_de_la_librairie`*").

Dans un premier temps, nous allons créer un fichier "`ouiSnap.py`" ("`.py`" étant l'extension du fichier précisant que le langage utilisé est bien du Python) dans lequel nous allons programmer notre logiciel.

Pour les inclure à un programme, on utilise le mot clé "`import`" suivi du nom de la librairie.

Pour ce programme nous allons utiliser les librairies suivantes :

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

- "tkinter" est une librairie graphique qui gère l'affichage d'une fenêtre et la gestion des boutons :

  "sudo apt-get install python3-tk"

- Installer PIP pour Python 3 le gestionnaire de modules pour Python :

"sudo apt install python3-pip"

Où :

"wget https://bootstrap.pypa.io/get-pip.py && sudo python3 ./get-pip.py"

- "cv2", une librairie qui permet de gérer la vidéo caméra.

  "pip install opencv-python"

- "numpy" sert pour les calculs mathématiques complexes (matrices) :

"pip install numpy1"

ou

"sudo python3 –m pip install python-numpy"

- "time" permet de gérer le temps :

  "sudo apt-get install time"

- "PIL" est une librairie qui gère les images :

"pip install pillow"

ou

"sudo –H python3 –m pip install pillow"

Les librairies ci-dessus contiennent des fonctions et des données que nous allons utiliser par la suite. Nous vous expliquerons leur rôle dans les cas où nous en aurons besoin.

Retenez simplement qu'elles doivent être indiquées au début de votre code !

3. Création d'un tableau essentiel au programme

Cette partie étant un peu complexe, nous ne nous attarderons pas sur celle-ci, dites-vous juste qu'il s'agit d'une création de tableau essentiel dans le fonctionnement du programme et à écrire juste en dessous de vos librairies.

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

# create a 5x5 kernel
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]
])

filter_choice = 0
take_gif = 0
```

Nous avons également initialisé deux variables ("filter_choice" et "take_gif") dont nous auront besoin plus tard à zéro sur les deux dernières lignes. Cela permettra de sécuriser la gestion de variables.

Niveau 1 - Afficher votre caméra dans une fenêtre

Nous allons donc voir comment afficher, dans une fenêtre, un flux vidéo continu pris grâce à votre webcam.

Lancer simplement la commande "code "le nom de votre fichier"" (ici ouinsap.py).

Rappel : N'oubliez pas d'ajouter vos librairies au début de votre fichier ainsi que le tableau précédent !

1. Définir les paramètres de la caméra et de la fenêtre :

Nous allons donner ici la hauteur (height) et la largeur (width) de notre caméra. Puis, en utilisant la librairie cv2, nous allons 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, il nous faut créer puis paramétrer la fenêtre (root) :

Ici nous avons choisis la touche <Escape> du clavier pour quitter la fenêtre et terminer le programme. Enfin, nous choisissons d'activer le mode plein écran de notre fenêtre.

```
#Definition de la fenetre
root = tk.Tk()
root.bind('<Escape>', lambda e: root.quit()) # Echap pour quitter
root.attributes('-fullscreen', True) # Plein ecran
```

2. Ajouter l'image dans la fenêtre

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

```
#Ajout de l'image dans la fenetre
lmain = tk.Label(root) # Objet qui va contenir la camera
lmain.pack(fill="none", expand=True) # Centrer l'image
show_frame()
```

Aide : La dernière ligne en surbrillance désigne un appel à une fonction.

Cette fonction permet de structurer des commandes dans un bloc. Il pourra être à nouveau appelé dans notre code.

Pour l'instant, la fonction appelée "show_frame" n'existe pas. Pour définir une fonction en python, la syntaxe est la suivante :

Def nom_de_la_fonction():

```
def show_frame():
    global filter_choice, take_gif
    #Options de la camera
    _, frame = cap.read()
    image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGBA)
    #Recuperation de l'image de cv2 pour Tkinter
    img = Image.fromarray(image)
    imgtk = ImageTk.PhotoImage(image=img)
    #Image de la camera dans l'objet
    lmain.imgtk = imgtk
    lmain.configure(image=imgtk)
    lmain.after(10, show_frame)
```

La première partie de cette fonction permet de définir les options de la caméra. Ensuite, il faut récupérer l'image au format Tkinter. Une fois l'image récupérer, il faut mettre l'image dans l'objet que vous avez créé précédemment ("lmain").

Cette première partie fonctionne déjà. Pour afficher votre fenêtre, il vous fait écrire une simple ligne à la fin de votre fichier avant de pouvoir exécuter votre programme.

```
root.mainloop()
```

Vous pouvez déjà lancer votre programme depuis le terminal avec la commande suivante :

```
➜ CC python3 ./ouiSnap.py
```

N'oubliez pas de tester votre programme à la fin de chaque étape !

Niveau 2 - Ajout de divers filtres sélectionnables

Dans cette deuxième étape, nous allons voir comment faire pour ajouter des filtres sur vos gifs.

Premièrement, afin de sélectionner le filtre que vous souhaitez appliquer sur vos photos ou GIFs, il sera nécessaire de rajouter deux boutons : "Filtre suivant" et "Filtre précédent".

Nous allons donc rajouter ceci à la fin de notre code :

```
#Ajout du bouton Suivant
button = tk.Button(root, text = 'Filtre Suivant', command=next_filter)
button.pack(side="right")
```

Ici nous avons créé le bouton permettant d'aller vers le filtre suivant, c'est à vous de créer celui permettant d'aller vers le filtre précédent ! Tout en sachant que la fonction permettant d'aller se placer sur le filtre précédent est nommée "prev_filter".

Au début de notre code nous avions déclaré une variable globale nommée filter_choice et que l'on initialise à 0.

Nous allons maintenant coder les fonctions prev_filter et next_filter, elles vont permettre de modifier la valeur de la variable indiquant quel filtre nous voulons utiliser.

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

Cette fonction nous permet d'aller vers le filtre précédent, à vous de coder celle permettant d'aller vers le suivant ! Sachant qu'actuellement nous avons 4 filtres et qu'elle doit se nommer "next_filter".

Maintenant afin d'appliquer ces filtres sur vos photos il faut rajouter quelques lignes de code à la fonctions show_frame :

```
def show_frame():
        global filter_choice, take_gif
        #Options de la camera
        _, frame = cap.read()
        #On applique le filtre a la frame
        image = apply_filter(frame)

        img = Image.fromarray(image)
        imgtk = ImageTk.PhotoImage(image=img)
    #Image de la camera dans l'objet
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
    #On rappelle la fonction
        lmain.after(10, show_frame)
```

On peut voir qu'on a rajouter une ligne pour appliquer les filtres sur la variable "image". En effet, on affecte la valeur du retour de la fonction apply_filter à "image"

```
def apply_filter(frame):
        global filter_choice
        #On applique le filtre a la frame
        if filter_choice == 0:
                image = frame
        elif filter_choice == 1:
                image = cv2.filter2D(frame, -1, kernel_5x5)
        elif filter_choice == 2:
                graySrc = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
                image = cv2.Laplacian(graySrc, cv2.CV_8U, graySrc, ksize=15)
        elif filter_choice == 3:
                image = max_rgb_filter(frame)
        return (image)
```

Pour le filtre numéro 4 (qui correspond au "filter_choice == 3"), on doit créer la fonction "max_rgb_filter" qui va appliquer des effets sur l'image.

```
def max_rgb_filter(image):
    # split the image into its BGR components
    (B, G, R) = cv2.split(image)

    # find the maximum pixel intensity values for each
    # (x, y)-coordinate,, then set all pixel values less
    # than M to zero
    M = np.maximum(np.maximum(R, G), B)
    R[R > M] = 0
    G[G > M] = 0
    B[B > M] = 0

    # merge the channels back together and return the image
    return cv2.merge([B, G, R])
```

Et voilà vous pouvez désormais appliquer 4 filtres sur vos photos ! Si vous le souhaitez, vous pouvez tenter d'en rajouter plusieurs.

Niveau 3 - Réalisation d'un GIF

Dans cette étape, nous allons ajouter un bouton pour vous permettre de prendre la photo.

Pour cela, nous allons réutiliser la fonction de la librairie graphique tkinter "tk.Button". On y ajoute ici le texte que l'on veut sur le bouton, ici, 'Prendre le gif' puis nous le positionnons dans la partie "top" de la fenêtre.

```
#Ajout du bouton Prendre le gif
button = tk.Button(root, text = 'Prendre le GIF', command=_take_gif)
button.pack(side="top")
```

On peut voir que le dernier paramètre de la fonction tk.Button fait appel à une fonction que nous devons créer "_take_gif" dans le cas où le bouton est activé.

Cette fonction va permettre de lancer la prise de photo.

ATTENTION : le nom de la variable dans cette fonction est "take_gif" et le nom de la fonction est "_take_gif", ce sont deux éléments du code totalement différents !

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

Il faut également modifier la fonction "show_frame" pour qu'elle applique les effets sur l'image enregistré.

Il vous faudra importer une dernière librairie qui est installée par défaut : « import os »

```
def show_frame():
        global filter_choice, take_gif
        #Options de la camera
        _, frame = cap.read()
        #On applique le filtre a la frame
        image = apply_filter(frame)
        #On prend le gif si la variable est > 0 -> appui sur le bouton
        if take_gif > 0:
                if take_gif == 1:
                        print("Starting to take GIF")
                if filter_choice == 3:
                        image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                if take_gif % 2 == 0:
                        cv2.imwrite('myimg' + str(take_gif) + '.png', image)
                take_gif += 1
        if take_gif == 120:
                print("Finished taking GIF")
                print("Converting pngs to GIF")
                os.system('convert -delay 10 myimg*.png animation.gif')
                os.system('rm -rf *.png')
                print("Done !")
                take_gif = 0
        if filter_choice == 0:
                image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        elif filter_choice == 3:
                image = max_rgb_filter(frame)
        img = Image.fromarray(image)
        imgtk = ImageTk.PhotoImage(image=img)
    #Image de la camera dans l'objet
        lmain.imgtk = imgtk
        lmain.configure(image=imgtk)
    #On rappelle la fonction
        lmain.after(10, show_frame)
```

Nous avons donc, comme pour les fonctions pour les filtres, créée une "global" pour être sûr de pouvoir réutiliser notre variable n'importe où dans notre code, notamment en dehors de notre fonction.

Si nous ne l'avions pas définie en "global", nous aurions dû appeler une autre fonction à l'intérieur de "_take_gif", en lui passant en paramètre la variable pour pouvoir la réutiliser.

Félicitation ! Vous avez réussi votre mission !!

Prenez à présent un super GIF du Winter Camp et postez-le sur Twitter en taguant @Epitech Toulouse !