CODING CLUB

EPITECH

Capy capy

DIVERSITY by EPITECH

I. Introduction

Il était une fois, dans une forêt tropicale d'Amérique du Sud, un capybara nommé Capy. Capy était un animal curieux et aimait explorer les mystères de l'informatique. Un jour, il découvrit un ordinateur portable caché dans un tronc d'arbre et décida de l'utiliser pour apprendre le langage Python. Capy se lança dans un projet ambitieux : convertir des images en ASCII !

II. Consignes

Avant de commencer cette introduction, voici les consignes techniques et les outils que vous allez utiliser : 1. Langage de programmation : Python

1. Bibliothèques nécessaires :
    1. argparse : pour gérer les arguments passés lors du lancement du programme
    2. opencv2 (OpenCV) : pour lire et manipuler les images
2. Installation des bibliothèques : Vous pouvez installer les bibliothèques nécessaires en utilisant pip :

```
pip install opencv-python
```

1. Environnement de développement : Utilisez un éditeur de code ou un environnement de développement intégré (IDE) de votre choix pour écrire et exécuter le code Python.
2. Structure du projet : Créez un fichier ["main.py"] où vous écrirez le code pour la conversion d'image en ASCII.

Suivez ces consignes et vous serez prêt à commencer le projet de conversion d'images en ASCII avec l'histoire du capybara Capy.

III. Argparse

Avant de commencer, Capy devait apprendre à utiliser "argparse", une bibliothèque Python pour gérer les arguments passés lors du lancement d'un programme. Pour ce faire, il créa un fichier "main.py" et importa "argparse" :

```
import argparse

def main():
    parser = argparse.ArgumentParser(description="Convertir une image en ASCII
    parser.add_argument("--path", required=True, type=str, help="Chemin vers l
    args = parser.parse_args()

    print("Image à convertir :", args.path)

if __name__ == "__main__":
    main()
```

Avec ce code, Capy pouvait exécuter le programme en passant le chemin de l'image en argument :

```
python main.py --path chemin/vers/image.jpg
```

IV. Classe Image

Maintenant que Capy savait comment récupérer les arguments, il décida de créer une classe "Image" pour gérer les opérations de conversion. Il utilisa la bibliothèque "opencv2" pour lire et manipuler les images :

```
import cv2

class Image:
    def __init__(self, path):
        self.image = cv2.imread(path, cv2.IMREAD_GRAYSCALE)

    def resize(self):
        # Écrire un programme qui resize notre image
        pass

    def to_ascii(self, ascii_chars):
        # Écrire la logique de conversion ici
        pass

    def display(self):
        # Afficher l'image ASCII ici
        pass
```

V. Conversion en ASCII

Dans cette étape, Capy devait écrire la logique de conversion d'une image en ASCII en se basant sur une chaîne de caractères triée du plus sombre au plus clair. Il utilisa une fonction "to_ascii" pour effectuer cette conversion nous disposons d'un autre programme qui nous ai donné nous permetant de créer une "ascii_chars", il s'agit d'une chaine de charactère trier du plus sombre au plus claire.

Générer une ascii_chars

Vous pouvez créer une ascii_chars en utilisant ce programme ainsi :

```
python3 ascii_chars_generator [-s START] -e END
```

avec START le code ascii du premier charactère et END le code ascii du dernier charactère que nous voulons trier. END >= START.

Convertir l'image en ascii

```
def to_ascii(self, ascii_chars):
    # déclarer mon tableau qui stockera les charactères ascii de mon image
    # faire un boucle pour chaque ligne de mon image
        # créer un tableau qui stockera ma ligne
        # créer une boucle pour chaque pixel
            # déterminer l'index du charactère ascii (0-len(ascii_chars)) uti
            # ajouter ce charactère à ma ligne
        # ajouter à mon tableau ma ligne qui est la concaténation de tous les
    # retourner la concaténation de toutes les lignes séparer par un "\n"
```

VI. Affichage de l'image ASCII

Enfin, Capy ajouta une fonction "display" pour afficher l'image ASCII à l'écran :

```
def display(self):
    # récupérer notre image en ascii
    # print notre image
```

Mais un problème se posa, Capy voyait plein de ligne s'afficher mais n'arriver pas à distinguer son image !

VII. Resize de l'image

Dans cette section de l'histoire, Capy le capybara, qui vit dans une forêt numérique, découvre qu'il peut redimensionner les images pour les adapter à l'écran de son ordinateur magique. Capy adore regarder les images de ses amis et de sa famille et veut les afficher correctement sur son écran, sans déformation ni perte de qualité.

Un jour, alors qu'il se promène dans la forêt, Capy rencontre un sage python qui lui enseigne comment redimensionner les images en conservant leur ratio hauteur/largeur. Le python explique à Capy que pour adapter l'image à l'écran, il doit suivre ces étapes :

1. Obtenir les dimensions de l'écran (largeur et hauteur en caractères).
2. Calculer les ratios de redimensionnement pour la largeur et la hauteur.
3. Utiliser le plus petit ratio pour redimensionner l'image tout en conservant son ratio hauteur/largeur.

Capy est très attentif et suit les instructions du sage python pour redimensionner les images de ses amis et de sa famille. Grâce à ces nouvelles connaissances, Capy peut désormais adapter les images à l'écran de son ordinateur magique sans déformation ni perte de qualité, ce qui rend ses souvenirs encore plus précieux.