1. Syntaxe en python

a. Créer une classe et instancier un objet

En Python pour créer une classe il suffit d'écrire :

```
class Nom_de_classe :
    #code de la classe
```

Par convention on mettra toujours une majuscule aux noms de classes.

Pour instancier un objet d'une classe on écrira :

```
mon_objet=Nom_de_classe()
```

Exemple : écrire la déclaration d'une classe « Personnage » et créer un personnage.

```
class Personnage :
    #code
mon_personnage = Personnage()
```

b. Créer le constructeur

La méthode constructeur a elle aussi une syntaxe particulière :

```
def __init__(self,param1 = valdefaut,param2 = valdefaut) :
        self.attribut1 = param1
        self.attribut2 = param2
```

Les attributs étant les attributs de l'objet et les paramètres étant les valeurs données au constructeur pour instancier l'objet.

Une fois le constructeur créé, l'instanciation s'effectue donc comme cela :

```
mon_objet= Nom_de_classe(valeur de l'attribut 1, valeur de l'attribut 2)
```

Si jamais j'écris mon_objet = Nom_de_classe(), alors les attributs prendront les valeurs par défaut.

c. Le mot clef : self

Self est un mot clef en Python qui est une auto-référence. Cela permet de dire dans une classe que l'on parle des méthodes et des attributs propres à la classe.

Donc :

- `self.qqch` : désigne un attribut de la classe
- `qqch` : désigne une variable n'étant pas un attribut de la classe

Comme les méthodes d'une classe appartiennent forcément à la classe, le premier paramètre d'une méthode sera toujours et obligatoirement « self ». Une classe ne prenant aucun paramètre prendra quand même comme paramètre « self »

Exemple : écrire le constructeur de la classe « Personnage », sachant qu'un personnage a 2 attributs, un pseudo et un niveau :

```
def __init__(self, pseudo = "defaut", niveau = 1) :
     self.pseudo = pseudo
     self.niveau = niveau
```

Exemple : écrire l'instanciation d'un personnage nommé p1 dont le pseudo est Link et étant de niveau 99 :

```
p1 = Personnage("Link",99)
```

d. La composition

Rien n'empêche un objet d'être composés d'autres objets. C'est même souvent le cas. On parle alors d'objets composites.

Exemple :

- Créer une classe Arme et son constructeur avec 1 attribut, les dégâts de l'arme.
- Modifier la classe Personnage précédente pour qu'un personnage ait un attribut arme
- Créer un personnage avec une arme qui fasse 12 de dégât.

```
Class Arme :
     def __init__(self,valeur) :
          self.degat = valeur
Class Personnage :
     def __init__(self,pseudo="rien",niveau=1,arme) :
          self.pseudo = pseudo
          self.niveau = niveau
          self.arme = arme

p1 = Personnage("Link",99,Arme(12))
```

e. Les méthodes

Pour ajouter une méthode dans une classe il suffit de la déclarer comme une fonction en ajoutant self comme 1er paramètre.

Exemple : Ajoutons la méthode cri_de_guerre(texte) qui permet d'afficher le texte donné en paramètre :

```
def cri_de_guerre(self,texte) :
     print(texte)
```

Une fois la méthode créée pour l'utiliser il faut le faire avec un objet et l'opérateur '.', par exemple si je veux que p1 utilise la méthode cri_de_guerre pour crier GROARG, il faut écrire :

```
p1.cri_de_guerre("GROARG")
```

De façon général lorsqu'on crée une classe il faut toujours penser à créer les méthodes qui permettent de modifier les attributs et de récupérer leurs valeurs.

Exemple :

- Ajouter les méthodes de la classe Personnage permettant de modifier le pseudo et l'âge
- Ajouter les méthodes permettant de récupérer l'âge et le pseudo
- Modifier le nom de p1
- Afficher le niveau de p1

```
def set_niveau(self,valeur) :
     self.niveau=valeur

def set_pseudo(self,valeur) :
     self.pseudo = valeur

def get_pseudo(self) :
     return self.pseudo

def get_niveau(self) :
     return self.niveau

p1.set_pseudo("Hulk")
print(p1.get_niveau())
```

2. Exercice :

a. La classe pixel

Une image numérique est généralement composée de pixels, l'objectif de cette partie est de créé en Python une classe pixel.

Exercice 1. A partir de vos connaissances ou d'une recherche sur internet, définir ce qu'est un pixel et de quoi cela est composé. En déduire les attributs de la classe.

Exercice 2. Créer la classe Pixel et le constructeur de façon à ce que par défaut les pixels soient noirs.

Exercice 3. Créer 3 pixels :

- p1 dont on ne connait pas les couleurs,
- p2 qui est rouge,
- p3 qui est jaune. (On pourra consulter internet si besoin pour savoir comment faire du jaune)

Exercice 4. Créer 3 méthodes qui permettent chacune de changer une des valeurs de couleur du pixel.

Exercice 5. Changer la composante bleue de p1 pour la valeur 128

Exercice 6. Créer une méthode qui permet de changer la couleur du pixel et qui prend comme paramètre les nouvelles valeurs des composantes vertes, rouges et bleues.

Exercice 7. Changer la couleur de p2 pour qu'il soit blanc.

Exercice 8. Créer 3 méthodes qui permettent chacune d'obtenir la valeur d'une composante de couleur du pixel

Exercice 9. Afficher la valeur de la composante verte de p3

Exercice 10. Créer la méthode qui permette de récupérer un tuple contenant la valeur de chacune des couleurs du pixel.

Exercice 11. Afficher la valeur des couleurs de p1

b. La classe image

L'objectif de cette partie est de créer une classe image que nous appellerons img qui servira à modéliser un fichier image.

Nous savons qu'une image est composée de pixels, mais ce ne sont pas les seuls attributs d'une image.

Exercice 1. Identifier les attributs d'une image :

Exercice 2. Créer la classe Img qui modélise une image, créer son constructeur qui initialise les attributs avec les valeurs en paramètre. (N.B : par défaut on mettra les images au format .jpg)

Pour simplifier les choses nous initialiserons systématiquement une image avec tous ses pixels noirs. *(Il serait cependant tout à fait possible d'initialiser l'image avec une matrice de pixels en paramètre)*

Exercice 3. Modifier le constructeur pour qu'il initialise tous les pixels de l'image à la couleur noir.

Exercice 4. Créer un objet img à partir de cette classe, qui est au format .jpg et qui fait 100 pixels de haut par 200 pixels de large.

Exercice 5. Créer les méthodes qui permettent de récupérer la valeur de chaque attributs (une méthode par attributs)

Exercice 6. Créer les méthodes qui permettent de modifier le nom et l'extension

Exercice 7. Créer une méthode qui permet de récupérer les informations de couleurs d'un pixel à des coordonnées précises

Exercice 8. Créer une méthode qui permet de modifier les informations de couleurs d'un pixel à des coordonnées précises

De façon à pouvoir créer et visualiser les fichiers images que nous créons, nous allons utiliser une autre classe Image pour générer les fichiers à partir de notre classe Img.

Pour cela ajouter l'import de la bibliothèque Image du module PIL à votre classe.

Ensuite ajoutez les 2 méthodes suivantes :

La méthode create permet de créer le fichier image dans le dossier courant du projet :

```
def create(self):
    im = Image.new('RGB', (self.largeur, self.hauteur), color='black')
    for i in range(self.largeur):
        for j in range(self.hauteur):
            im.putpixel((i, j), self.matrice[i][j].get_couleurs())
    im.save(self.nom+self.extension)
```

La méthode show permet d'afficher l'image :

```
def show(self):
    i = Image.open(self.nom+self.extension)
    i.show()
    i.close()
```

Exercice 9. Modifier l'objet img que vous avez créé de façon à ce qu'il y ait un carré blanc de 30 pixels de côté dans la partie inférieur gauche de l'image.

Exercice 10. Créer et afficher votre objet image.

Comme nous l'avons déjà vu, il n'est pas possible de copier une liste avec l'opérateur « = » de même il n'est pas possible de copier un objet de cette façon. Car cela copie l'adresse en mémoire de l'objet, mais pas l'objet en lui-même.
Il n'est donc pas possible de créer une deuxième image en faisant image1 = image2

Pour copier un objet, il faut copier les attributs à copier un à un. Dans notre cas nous ne copierons pas le nom car il n'est pas possible d'avoir 2 fichiers avec le même nom.

Exercice 11. Créer une méthode qui prend comme paramètre une image et qui copie l'extension et les pixels un à un dans l'objet courant.

Exercice 12. Créer un deuxième objet de type img sans préciser les paramètres et copier dedans le 1er objet img.

Exercice 13. Changer le nom de l'objet que vous venez de créer pour l'appeler « image2 »

Exercice 14. Créer et ouvrer l'image, pour vérifier que tout s'est bien passé.

Exercice 15. Créer une méthode « vertical_miroir » qui permet de modifier l'image de façon à obtenir son image miroir. (C'est-à-dire faire une permutation des pixels par rapport à un axe verticale)

Exercice 16. Utiliser la méthode vertical_miroir pour modifier image2. Afficher l'image pour vérifier si cela a fonctionné.

Exercice 17. Créer une méthode « horizontal_miroir » qui permet de modifier l'image de façon à obtenir son image miroir par rapport à un axe horizontal cette fois-ci.

Exercice 18. Créer un 3ème objet img, copier dedans le 2ème objet img et effectuez dessus la rotation horizontale. Affichez l'image pour vérifier si cela a fonctionné.