Cet exercice porte sur la gestion des processus et des ressources par un système d'exploitation.

1. Les états possibles d'un processus sont : *prêt*, *élu*, *terminé* et *bloqué*.

   a. Expliquer à quoi correspond l'état *élu*.

   b. Proposer un schéma illustrant les passages entre les différents états.

2. On suppose que quatre processus C_1, C_2, C_3 et C_4 sont créés sur un ordinateur, et qu'aucun autre processus n'est lancé sur celui-ci, ni préalablement ni pendant l'exécution des quatre processus.

   L'ordonnanceur, pour exécuter les différents processus prêts, les place dans une structure de données de type file. Un processus prêt est enfilé et un processus élu est défilé.

   a. Parmi les propositions suivantes, recopier celle qui décrit le fonctionnement des entrées/sorties dans une file :

      i. Premier entré, dernier sorti

      ii. Premier entré, premier sorti

      iii. Dernier entré, premier sorti

   b. On suppose que les quatre processus arrivent dans la file et y sont placés dans l'ordre C_1, C_2, C_3 et C_4.

      - Les temps d'exécution totaux de C_1, C_2, C_3 et C_4 sont respectivement 100 ms, 150 ms, 80 ms et 60 ms.

      - Après 40 ms d'exécution, le processus C_1 demande une opération d'écriture disque, opération qui dure 200 ms. Pendant cette opération d'écriture, le processus C_1 passe à l'état bloqué.

      - Après 20 ms d'exécution, le processus C_3 demande une opération d'écriture disque, opération qui dure 10 ms. Pendant cette opération d'écriture, le processus C_3 passe à l'état bloqué.

      Sur la frise chronologique donnée en annexe (à rendre avec la copie), les états du processus C_2 sont donnés. Compléter la frise avec les états des processus C_1, C_3 et C_4.

C1
élu
C2
prêt
élu
terminé
C3
C4
0
20
40
60
80
100
120
140
160
180
200
220
240
260
280
300
320
340
360
380
400

II

Cet exercice porte sur la gestion des processus et la programmation orientée objet

On rappelle qu'un processus est l'instance d'un programme en cours d'exécution. Il est identifié par un numéro unique appelé PID. L'ordonnanceur est la composante du système d'exploitation qui gère l'allocation du processeur entre les différents processus. Nous allons nous intéresser à l'algorithme d'ordonnancement du tourniquet dont le fonctionnement est résumé ci-dessous :

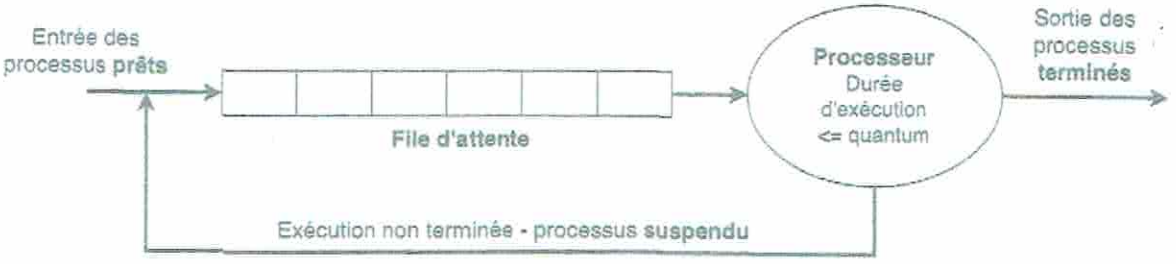

Schéma d'ordonnancement du tourniquet

- Les processus prêts à être exécutés sont placés dans une file d'attente selon leur ordre d'arrivée ;
- L'ordonnanceur alloue le processeur à chaque processus de la file d'attente un même nombre de cycles CPU, appelé **quantum** :

- Si le processus n'est pas terminé au bout de ce temps, son exécution est suspendue et il est mis à la fin de la file d'attente ;
- Si le processus est terminé, il sort définitivement de la file d'attente.

1. On considère trois processus soumis à l'ordonnanceur **au même instant** pour lesquels on donne les informations ci-dessous :

PID	Durée (en cycles CPU)	Ordre d'arrivée
11	4	1
20	2	2
32	3	3

   a) Si le quantum du tourniquet est d'un cycle CPU, recopier et compléter la suite des PID des processus dans l'ordre de leur exécution :
   11, 20, 32, 11, ...

   b) Donner la composition de la suite des PID lorsque le quantum du tourniquet est de deux cycles CPU.

2. L'objectif de la suite de l'exercice est d'implémenter en langage Python l'algorithme du tourniquet.
   Nous allons utiliser une liste pour simuler la file d'attente des processus et la classe `Processus` dont le constructeur est donné ci-dessous :

```
class Processus:
    def __init__(self, pid, duree):
        self.pid = pid
        self.duree = duree
        # Le nombre de cycle qui restent à faire :
        self.reste_a_faire = duree
        self.etat = "Prêt"
```

   Les états possibles d'un processus sont : « *Prêt* », « *En cours d'exécution* », « *Suspendu* »et « *Terminé* ».

   a) Recopier et compléter l'instruction Python suivante permettant de créer la liste d'attente initiale des processus donnés dans le tableau précédent (le processus PID 11 est à l'indice 0 de la liste d'attente) :

```
liste_attente = [Processus(...,...), ..........., ..........]
```

   b) Recopier (sans les commentaires) et compléter les trois méthodes suivantes de la classe `Processus` :

```
    def execute_un_cycle(self):
        """Met à jour le reste à faire après l'exécution d'un
        cycle."""
        ..........................................


    def change_etat(self, nouvel_etat):
        """Change l'état du processus avec la valeur passée en
        paramètre."""
        ..........................................


    def est_termine(self):
```

```
        """Renvoie True si le processus est terminé, False sinon,
        en se basant sur le reste à faire."""
        ..... .....
```

c) La fonction `tourniquet` ci-dessous implémente l'algorithme décrit dans l'exercice.
Elle prend en paramètre une liste d'objets `Processus` donnés par ordre d'arrivée et un nombre entier positif correspondant au quantum. La fonction renvoie la liste des PID dans l'ordre de leur exécution par le processeur.
Recopier et compléter sur la copie le code manquant.

```
1  def tourniquet(liste_attente, quantum):
2      ordre_execution = []
3      while liste_attente != []:
4          # On extrait le premier processus
5          processus = liste_attente.pop(0)
6          processus.change_etat("En cours d'exécution")
7          compteur_tourniquet = 0
8          while ..... and .....:
9              ordre_execution.append(.....)
10             processus.execute_un_cycle()
11             compteur_tourniquet = compteur_tourniquet + 1
12         if .....:
13             processus.change_etat("Suspendu")
14             liste_attente.append(processus)
15         else:
16             processus.change_etat(.....)
17     return ordre_execution
```