

# RÉSUMÉ 34 - PARADIGMES DE PROGRAMMATION

Lien vers l'activité : [Paradigmes de programmation](#)  
www.infoforall.fr - Dernière modif. : 08 03 2021



## PROGRAMMES DE LA PARTIE PARADIGME IMPÉRATIF

### Exemple n°1 Impératif

```
1  # Initialisation des variables définissant l'état au début
2
3  vie_a = 400
4  force_a = 60
5  resistance_a = 63
6  nom_a = "Alice"
7
8  vie_b = 300
9  force_b = 70
10 resistance_b = 54
11 nom_b = "Bob"
12
13 # Evolution séquentielle des états des variables
14
15 while vie_a > 0 and vie_b > 0:
16     vie_a = vie_a - force_b + resistance_a
17     vie_b = vie_b - force_a + resistance_b
18
19 # Résolution du problème une fois l'état final atteint
20
21 if vie_a > 0:
22     gagnant = 1
23 elif vie_b > 0:
24     gagnant = 2
25 else:
26     gagnant = 0
27
28 # Affichage du résultat (ou inscription dans un fichier ou autre)
29
30 if gagnant == 0:
31     print("Aucun gagnant : les deux combattants sont ko")
32 elif gagnant == 1:
33     print(f"{nom_a} a gagné et il lui reste {vie_a} PV")
34 else:
35     print(f"{nom_b} a gagné et il lui reste {vie_b} PV")
```

### Exemple n°2

```
15 if vie_a <= 0 or vie_b <= 0:
16     goto 23
17 vie_a = vie_a - force_b + resistance_a
18 vie_b = vie_b - force_a + resistance_b
19 goto 15
20
21 # Résolution du problème une fois l'état final atteint
22
23 if vie_a > 0:
```

**01**° Etudier le code suivant puis répondre aux questions :

1. **ta** est-elle une variable globale ou locale ?
2. **tb** est-elle une variable globale ou locale ?
3. **tc** est-elle une variable globale ou locale ?
4. En Python, quel va être la différence entre les lignes 4 et 5 ?
5. quel va être le contenu du tableau **ta** après l'appel de la fonction ?

```

1  def mystere():
2      '''Agit (ou pas ?) sur le tableau-paramètre p reçu'''
3
4      tb = [v for v in ta]
5      tc = ta
6      tb[0] = 100
7      tc[1] = 200
8
9
10     ta = [10, 20, 30]
11     mystere()

```

**02**° Comment se nomme l'effet qui permet de modifier le tableau **ta** ?

Expliquer le phénomène en utilisant les mots variables, alias et zone mémoire.

**03**° Que devrait afficher ce programme ?

Attention : en Python, l'évaluation du **range** de la boucle ne se fait qu'au départ.

```

1  a = ["A", "B", "C"]
2  b = a
3  c = b
4
5  for i in range(len(a)):
6      c.append(a[i])
7      print(a)

```

**04**° Que devrait afficher ce programme ?

Attention : ici, il n'y a pas de **range** mais une simple fonction native **len** qui sera donc évaluée à chaque tour de boucle...

```

1  a = ["A", "B", "C"]
2  b = a
3  c = a
4
5  i = 0
6  while i < len(a) and i < 60:
7      c.append(a[i])
8      print(a)
9      i = i + 1

```

**05**° Mettre le programme en mémoire puis utiliser les instructions consoles fournies. Que va afficher la dernière commande ?

```

1  entrees = ['tarte au saumon', 'crudités']
2  plats = ['steak frites', 'ratatouille']
3  menus = [entrees, plats]
4
5  entrees2 = entrees
6  plats2 = ['Pates aux épinards', 'Gratin de brocolis']
7  menus2 = [entrees2, plats2]

```

## PROGRAMMES DE LA PARTIE PROGRAMMATION ORIENTÉ OBJET

**06**° Associer le bon vocabulaire aux noms suivants :

1. Personnage
2. a
3. `__init__`
4. `obtenir_vie`
5. `self`
6. `force`

**07**° Expliquer ce que reçoivent les deux paramètres **self** et **adversaire** de la méthode **lancer\_combat\_avec** lorsqu'on lance ceci :

```

>>> a.lancer_combat_avec(b)
'Alice'

```

**08**° Expliquer ensuite rapidement le comportement de cette méthode.

## Exemple Objet

```
1 class Personnage:
2     '''Classe définissant le personnage'''
3
4     def __init__(self, nom:str, pv:int, fo:int, re:int):
5         '''Méthode initialisateur ou constructeur'''
6         self.nom = nom
7         self.vie = pv
8         self.force = fo
9         self.resistance = re
10
11     def subir_degats(self, dgt:int):
12         '''(privée) Modifie les pv du personnage'''
13         if type(dgt) == int and dgt > 0:
14             self.vie = self.vie - dgt
15         if self.vie < 0:
16             self.vie = 0
17
18     def combattre(self, adversaire):
19         '''(privée) Self subit l'assaut de l'adversaire'''
20         degats = adversaire.obtenir_force() - self.resistance
21         self.subir_degats(degats)
22
23     def obtenir_nom(self) -> str:
24         '''Renvoie le nom du personnage'''
25         return self.nom
26
27     def obtenir_vie(self) -> int:
28         '''Renvoie le nombre de vie restante'''
29         return self.vie
30
31     def obtenir_force(self) -> int:
32         '''Renvoie la force du personnage'''
33         return self.force
34
35     def obtenir_resistance(self) -> int:
36         '''Renvoie la résistance du personnage'''
37         return self.resistance
38
39     def lancer_combat_avec(self, adversaire) -> str:
40         '''Lance un combat entre 2 personnages et renvoie le gagnant'''
41         while self.obtenir_vie() > 0 and adversaire.obtenir_vie() > 0:
42             self.combattre(adversaire)
43             adversaire.combattre(self)
44         if self.obtenir_vie() > 0:
45             return self.nom
46         elif adversaire.obtenir_vie() > 0:
47             return adversaire.nom
48         else:
49             return "Aucun gagnant"
50
51 if __name__ == '__main__':
52     a = Personnage("Alice", 400, 60, 63)
53     b = Personnage("Bob", 300, 70, 54)
```

## PROGRAMMES DE LA PARTIE PROGRAMMATION FONCTIONNELLE

**09** ° Voici une fonction **plus2**. S'agit-il d'une version impérative ou fonctionnelle ?

```

1  def plus2(t:list) -> None:
2      '''Modifie t par effet de bord en rajoutant 2 à chaque valeur
3
4      :: param t(list) :: le tableau qu'on veut modifier
5      :: return(None) :: fonction-procédure
6
7      :: exemple ::
8      >>> test = [1, 2, 3]
9      >>> plus2_iter(test)
10     >>> test
11     [3, 4, 5]
12
13     '''
14     for i in range(len(t)):
15         t[i] = t[i] + 2

```

10° Voici une fonction **plus2**. S'agit-il d'une version impérative ou fonctionnelle ?

```

1  def plus2_fonc(t:list) -> list:
2      '''Renvoie une copie du tableau où toutes les valeurs sont augmentées de 2
3
4      :: param t(list) :: le tableau qu'on veut modifier
5      :: return(list) :: la nouvelle version du tableau
6
7      :: exemple ::
8      >>> plus2_fonc([1, 2, 3])
9      [3, 4, 5]
10
11     '''
12     return [v+2 for v in t]

```

11° Comparer les 3 versions fournies permettant de rajouter un élément dans un tableau Python. Laquelle pourrait correspondre à une version plus ou moins fonctionnelle ? Quelle est encore le gros défaut d'un point de vue fonctionnel ?

Voir les 3 tests sur le site.

12° Créer la fonction **ajouter\_elt(t, elt)** qui renvoie un nouveau tableau semblable à celui reçu mais en lui rajoutant un nouvel élément.

13° La fonction **somme** ci-dessous peut-elle être considérée comme fonctionnelle ?

```

1  def somme(t:list) -> int:
2      if len(t) == 0:
3          return 0
4      else:
5          return t[0] + somme([t[i] for i in range(len(t)) if i > 0])

```

### Exemple Fonctionnel

```

1  # Déclaration des fonctions
2
3  def nv_pv(df:tuple, at:tuple) -> int:
4      '''Calcule le pv du défenseur df après attaque de l'attaquant at'''
5      return df[0] - max(0, at[1] - df[2])
6
7  def maj(df:tuple, at:tuple) -> tuple:
8      '''Renvoie les nouvelle caractéristiques mise à jour du défenseur df après ataq
9      return nv_pv(df, at), df[1], df[2], df[3]
10
11  def est_hors_jeu(p):
12      '''Fonction booléenne qui renvoie True si le personnage p à 0 pv ou moins'''
13      return p[0] <= 0
14
15  def lancer_combat(a:tuple,b:tuple) -> str:
16      '''Lance le combat entre a et b et renvoie le nom du gagnant ou "Aucun gagnant"
17      if est_hors_jeu(a) or est_hors_jeu(b): # Condition d'arret et cas de base
18          if not est_hors_jeu(a): # a a gagné car il lui reste des pv
19              return a[3]
20          elif not est_hors_jeu(b): # b a gagné car il lui reste des pv
21              return b[3]
22          return "Aucun gagnant"
23      else : # Cas recursif
24          return lancer_combat(maj(a,b), maj(b,a))
25
26  # ETAT DU SYSTEME
27  # Chaque personne est un tuple (vie-0, force-1, resistance-2, nom-3)
28  a = (400, 60, 63, "Alice")
29  b = (300, 70, 54, "Bob")

```

