

1 On représente un point par un tuple (x, y) de ses coordonnées.

• Écrire une fonction **milieu** qui prend deux points $p1$ et $p2$ en paramètres et retourne le milieu du segment $(p1, p2)$.

2 On représente une date par un tuple (j, m, a) où j est le jour du mois, m le numéro du mois (de 1 à 12) et a l'année.

a. Écrire une fonction **antérieur** qui prend deux dates $d1$ et $d2$ en paramètres et retourne **True** si $d1$ est antérieure à $d2$.

b. Écrire une fonction **age** qui prend deux dates $d1$ et $d2$ en paramètres et retourne le nombre d'années pleines entre $d1$ et $d2$. Peut-on utiliser la fonction de la question a. ?

3 On représente une personne par un tuple $(nom, date)$ où nom est lui-même un tuple formé du prénom et du nom, et $date$ est également un tuple représentant la date de naissance comme dans l'exercice 2.

Écrire une fonction qui prend une personne en paramètre et retourne un tuple formé du nom de famille et de l'année de naissance.

4 Écrire une fonction **moyenne** qui prend en paramètre un tableau de nombres et retourne leur moyenne arithmétique.

5 Écrire une fonction **est_ordonne** qui prend en paramètre un tableau de nombres et retourne **True** si ses éléments sont ordonnés par ordre croissant.

6 On dispose d'un tableau $t = [(0, 0), (0, 100), (50, 150), (100, 100), (100, 0)]$ dont les éléments sont des tuples (x, y) représentant des points.

a. Écrire une fonction **unzip** qui prend un tel tableau t en paramètre et retourne un tuple (tx, ty) de deux tableaux contenant respectivement les coordonnées x et y des points de t .

b. Écrire une fonction **zip** qui réalise l'opération inverse : à partir de deux tableaux tx et ty de nombres, retourner un tableau de tuples (x, y) .

7 a. Initialiser un tableau **divisible_7** de 1000 booléens de valeur **False**, puis écrire ensuite une boucle qui met à **True** les éléments du tableau dont l'indice est divisible par 7.

Par exemple, **divisible_7[2]** est **False** mais **divisible_7[14]** est **True**.

b. Combien de nombres inférieurs à 1 000 sont divisibles par 7 ? Quel est le plus grand nombre divisible par 7 et inférieur à 1 000 ?

8 Écrire une fonction **palindrome** qui prend en paramètre une chaîne de caractères et retourne **True** si c'est un palindrome, c'est-à-dire s'il peut se lire dans les deux sens, comme le mot « ressassar ».

9 Écrire une fonction qui prend en paramètre une chaîne de caractères représentant un entier positif, par exemple "12345", et retourne le nombre entier correspondant, sans utiliser la fonction **int()**.

10 Écrire une fonction qui prend en paramètre un mot de passe et retourne **True** s'il vérifie les propriétés suivantes : il a au moins 8 caractères et contient au moins une majuscule, une minuscule, un chiffre et un caractère spécial (qui n'est ni une lettre ni un chiffre).

11 Écrire deux versions d'une fonction **nb_mots** qui retourne le nombre de mots dans une phrase. On suppose que tous les mots sont séparés par un espace. Ainsi, **nb_mots("Le petit chat est mort")** retourne 5.

Une version doit utiliser la méthode **split** des chaînes de caractères, l'autre non.

12 Écrire deux versions d'une fonction **longueur_mots** qui prend en paramètre une phrase (une chaîne de caractères) et retourne un tableau contenant la longueur des mots qui la composent. Ainsi, **longueur_mots("Le petit chat est mort")** retourne $[2, 5, 4, 3, 4]$.

Une version doit utiliser la méthode **split** des chaînes de caractères, l'autre non.

13 Initialiser un dictionnaire **capitales** qui associe à chaque nom de pays le nom de sa capitale avec quelques pays européens. Écrire une fonction qui prend un nom de ville en paramètre et retourne le pays dont elle est la capitale, on **None** si elle n'est pas dans le dictionnaire.

14 On considère un dictionnaire **personnes** qui associe à des noms de personnes un dictionnaire contenant des informations personnelles :

```
personnes = {
    "Jean Aymar": {"taille": 178,
                  "pays": "USA", "age": 31},
    "Clio Patre": {"pays": "Portugal",
                  "age": 32, "taille": 179}
}
```

a. Écrire une fonction qui prend un nom de personne en paramètre et retourne son age, ou **None** si la personne n'est pas dans le dictionnaire.

b. Écrire une fonction qui calcule la taille moyenne des personnes dans le dictionnaire.

15 Cartes à jouer

On définit une carte comme un tuple $(valeur, couleur)$. *valeur* est un entier de 2 à 14 inclus, où 11 représente le Valet et 14 l'As. *couleur* est une chaîne de caractères parmi « Pique », « Coeur », « Carreau », « Trèfle ».

a. Écrire une fonction **carte_valide** qui prend un tuple en paramètre et retourne un booléen qui indique s'il représente une carte valide.

b. Écrire une fonction **nom_carte** qui prend un tuple représentant une carte en paramètre et retourne une chaîne de caractères avec le nom de la carte, par exemple "As de Trèfle" ou "7 de Pique".

c. Écrire un programme qui crée un tableau contenant toutes les cartes d'un jeu de 52 cartes.

d. À l'aide de la fonction **random()** de la bibliothèque **random**, tirer une carte au hasard dans le tableau et afficher son nom.

16 Calcul de distances

a. Écrire une fonction **distance** qui prend deux tuples contenant chacun deux coordonnées x, y , et qui retourne leur distance euclidienne. Pour rappel, la fonction **sqrt** retourne la racine carrée d'une expression, et doit être importée depuis la bibliothèque **math**.

b. Un dessin est stocké dans un tableau contenant ses points successifs sous forme de tuples (x, y) . Écrire une fonction qui prend en paramètre un tel tableau et retourne les dimensions (*largeur, hauteur*) du dessin.

c. Écrire une fonction qui calcule la longueur du dessin, en utilisant la fonction **distance** de la première question.

17 Inverser un tableau

a. Écrire une fonction **inverse_tableau** qui prend en paramètre un tableau et retourne un tableau avec les mêmes éléments en ordre inverse.

b. Écrire une fonction **inverse_chaine** qui fait la même chose pour une chaîne de caractères. Peut-on utiliser la fonction **inverse_tableau** ?

18 Tester le hasard

La fonction **random** de la bibliothèque Python **random** retourne des nombres aléatoires entre 0 et 1. On veut vérifier que ces nombres sont bien aléatoires, c'est-à-dire qu'ils sont uniformément répartis entre 0 et 1.

a. Écrire un programme qui appelle **random** 1 000 fois et enregistre dans un tableau **t** de dix entiers le nombre de fois où le résultat est dans chacun des dix intervalles $[0, 0.1], [0.1, 0.2], \dots, [0.9, 1.0]$. Afficher le résultat sous forme d'histogramme avec le code suivant :

```
from matplotlib.pyplot import bar, show
bar(range(len(t)), t)
show()
```

b. Répéter le calcul et l'affichage plusieurs fois, puis avec un nombre de répétitions de plus en plus grand (jusqu'à 10 millions). Qu'observe-t-on ?

19 Loi Normale

a. Simuler le lancement de deux dés et compter le nombre de fois où la somme vaut 1, 2, ..., 12. Afficher l'histogramme comme dans l'exercice 18 pour différents (grands) nombres de tirages.

NOTE

La fonction **randint(a, b)** de la bibliothèque **random** retourne un nombre aléatoire entre a et b inclus.

b. Modifier le programme précédent pour simuler le lancement de n dés un grand nombre de fois et afficher l'histogramme correspondant.

La courbe obtenue lorsque n augmente s'approche de la « Loi Normale » et est très utilisée en probabilités et statistiques.

c. Une autre façon de faire apparaître la Loi Normale est de simuler un jeu de pile ou face : Tirer des séries de 100 lancers de pile ou face et compter le nombre de fois où pile apparaît. Compter dans chaque élément $t[i]$ d'un tableau le nombre de fois où une série donne i fois pile. Afficher l'histogramme correspondant.

20 Taux d'imposition

En France comme dans beaucoup de pays, les impôts et taxes suivent un barème progressif, appelé « tranches ». Ainsi le taux d'impôt sur le revenu est de 0 % en-dessous de 10 000 € de revenus annuels, de 11 % entre 10 000 € et 25 000 €, de 30 % entre 25 000 et 75 000 €, 41 % entre 75 000 et 150 000 €, 45 % au-delà de 150 000 €.

a. Définir un type construit pour représenter les tranches d'imposition et initialiser une variable **tranches** avec les tranches définies ci-dessus.

b. Écrire une fonction qui calcule le montant de l'impôt étant donné un revenu.

c. Pour montrer la progressivité de l'impôt, créer à l'aide de cette fonction un tableau contenant les montants des impôts pour des revenus de 0 à 200 000 € par pas de 5 000 €. Afficher ce tableau à l'aide du code suivant, où **tx** est le tableau des abscisses (les revenus) et **ty** celui des ordonnées (l'impôt) :

```
from matplotlib.pyplot import plot, show
plot(tx, ty)
show()
```

- d. Même question mais cette fois afficher le *taux marginal d'imposition*, c'est-à-dire le rapport, exprimé en pourcentage, du montant de l'impôt par rapport au revenu.
- e. Comparer le taux marginal avec le taux d'imposition de la tranche correspondante pour des revenus annuels de 15 000 €, 50 000 €, 200 000 €.

21 Anagramme

- a. Écrire une fonction `retire_element(t, e)` qui retire l'élément `e` de `t` s'il est présent et retourne `True`, ou sinon retourne `False`. On utilisera la méthode `pop` pour retirer l'élément du tableau.
- b. Écrire une fonction `meme_contenu` qui prend en paramètres deux tableaux et retourne `True` s'ils contiennent les mêmes éléments, même si leur ordre diffère ; `False` sinon. Attention aux alias !
- c. Utiliser cette fonction pour écrire une fonction `anagramme` qui prend en paramètres deux chaînes de caractères, et retourne `True` si elles sont composées des mêmes lettres dans un ordre différent.

22 Mélanger un tableau

- a. Écrire une fonction `mélanger` qui prend un tableau en paramètre et retourne un tableau avec les mêmes éléments dans un ordre aléatoire. On utilisera la méthode `pop` des tableaux et la fonction `randint` de la bibliothèque `random`. Attention aux alias !
- b. Appliquer cette fonction au tableau des 52 cartes de l'exercice 15.

23 Bataille

Dans le jeu de la Bataille, à chaque tour de jeu, chaque joueur pose une carte sur la table puis on établit celle qui l'emporte : un 3 bat un 2, un Valet bat un 10, un As bat un Roi.

- a. Écrire une fonction `duel` qui prend deux tuples représentant deux cartes en paramètre et retourne 0 en cas d'égalité, 1 si la première carte gagne, 2 si c'est la seconde.
- b. Écrire une fonction `tirer` qui tire une carte au hasard. À l'aide de cette fonction, écrire une fonction `jouer` qui tire deux cartes et les met en duel. En cas d'égalité, elle tire deux nouvelles cartes jusqu'à ce qu'un joueur gagne. La fonction retourne un tuple formé du numéro du gagnant et du nombre de duels.
- c. Modifier la fonction `tirer` pour utiliser la fonction `mélanger` de l'exercice précédent et ainsi éviter que l'on tire plusieurs fois la même carte.

24 Inverser un dictionnaire

- a. Écrire une fonction `inverse_dict` qui prend en paramètre un dictionnaire et inverse ses clés et ses valeurs. Appliquer cette fonction au dictionnaire des capitales de l'exercice 13.
- b. Est-il toujours possible d'inverser un dictionnaire ? Pourquoi ?
- c. Est-on sûr que `inverse_dict(inverse_dict(dico)) == dico` ? Pourquoi ?

25 Scrabble

Le jeu de Scrabble consiste à construire des mots à l'aide de lettres sur une grille. Chaque lettre a une valeur numérique permettant de calculer des scores.



Le dictionnaire suivant indique les valeurs des lettres dans les règles françaises du Scrabble :

```
valeurs_Scrabble = {10: 'kwxyz', 8: 'jq',
                    4: 'fhv', 3: 'bcp', 2: 'dmg'}
```

Toutes les autres lettres ont une valeur de 1.

- a. En utilisant `valeurs_Scrabble`, calculer la valeur des mots « pizza », « whisky », « dédramatiser » (é compte pour e).
- b. Il n'est pas pratique de parcourir chaque valeur et chaque chaîne de `valeurs_Scrabble` pour trouver la valeur de chaque lettre. À partir de `valeurs_Scrabble`, créer le dictionnaire `lettres_Scrabble` qui fait correspondre à chaque lettre de l'alphabet son score : `{"a": 1, "b": 3, "c": 3, ...}`.
- c. Écrire une fonction qui calcule la valeur d'un mot. Quel mot apporte le plus de points parmi : entraîs, ratines, satiner, riantes, transie ?
- d. J'ai accès à une case « lettre compte triple » au 4^e élément d'une rangée de 7 lettres. Cette case triple la valeur des points de la lettre qui est posée dessus. Écrire un programme permettant de savoir lequel des mots précédents me rapportera le plus de points.

26 Géographie



- a. On dispose d'un dictionnaire `villes_coordonnees` qui associe à chaque nom de ville (chaîne) ses coordonnées sous forme d'un tuple (`longitude, latitude`). Afficher les noms de toutes les villes dont la latitude est inférieure à 23,43 degrés.
- b. On dispose également d'un dictionnaire `villes_population` qui associe à chaque nom de ville (chaîne) son nombre d'habitants (entier). Afficher les coordonnées de la ville ayant le plus d'habitants.
- c. On dispose enfin d'un dictionnaire `villes_pays` qui associe à chaque nom de pays (chaîne) un tableau de ses villes. Afficher le nombre total d'habitants des villes d'un pays donné et la latitude et la longitude moyenne de ces villes.

27 Liste de tâches simple

On définit une liste de tâches `todo_list` sous la forme d'un dictionnaire dont les clés sont des chaînes de caractères (les tâches à effectuer) et les valeurs des booléens (selon qu'une tâche est effectuée ou non).



- a. Créer un dictionnaire `todo_list` et y ajouter les tâches « Faire les courses » (fait), « Ranger le garage » (à faire), et « Compléter l'exercice 4 » (à faire).
- b. Écrire une fonction qui retourne le nombre de tâches non complétées dans `todo_list`, et les affiche.
- c. Écrire un programme avec une interface textuelle pour gérer la liste de tâches. L'interface affiche la liste numérotée des tâches et attend une commande de l'utilisateur. Les tâches déjà réalisées sont précédées d'un caractère v. Les commandes sont :
- + pour ajouter une tâche. Le programme demande ensuite le nom de la tâche ;
 - suivi d'un numéro pour retirer la tâche correspondante ;
 - v suivi d'un numéro pour changer le statut fait / à faire de la tâche correspondante ;
 - ? pour afficher seulement les tâches restant à faire.

28 Annuaire

- a. Écrire une fonction qui prend en paramètre une chaîne de caractères formée d'un prénom, d'un nom et d'un numéro de téléphone, par exemple `"Jean Dupont 0987654321"` et retourne un tuple constitué du nom, du prénom et du numéro de téléphone.

- b. On veut créer un dictionnaire dont les clés sont des noms de personne et les valeurs des chaînes de caractères représentant leur numéro de téléphone. Écrire une fonction qui prend un tuple de la forme ci-dessus et ajoute l'information au dictionnaire.

c. Écrire un programme avec une interface textuelle pour gérer un annuaire. Le programme demande si l'on veut ajouter ou rechercher une entrée dans l'annuaire. Dans le premier cas, le programme demande à entrer sur une seule ligne le prénom, le nom et le numéro de téléphone et ajoute l'entrée au dictionnaire ; dans le second cas, il demande à entrer seulement le prénom et le nom et affiche le numéro de téléphone correspondant s'il existe, un message d'erreur sinon.

d. Modifier le contenu du dictionnaire et le programme pour que l'on puisse rechercher un numéro de téléphone en entrant seulement le nom ou le prénom.

29 Index alphabétique

On considère un tableau de mots, par exemple :

```
["arbre", "androïde", "martien", "concept",
 "bleu", "but", "abri", "bateau", "art"]
```

On souhaite ranger ces mots dans un index, c'est-à-dire un dictionnaire Python dont les clés sont des lettres ("a", "b", etc.) et dont les valeurs sont des *tableaux* de mots commençant par cette lettre.

- a. Écrire une fonction `ajoute` qui prend un mot en paramètre et l'ajoute à l'index. Utiliser cette fonction pour ajouter à l'index les mots de la liste ci-dessus.
- b. Écrire une fonction `retire` qui prend un mot en paramètre et le retire de l'index s'il est présent. Si plus aucun mot ne commence par une certaine lettre, il faut retirer cette lettre des clés de l'index.

30 Analyse de texte



On dispose de la suite des mots d'une texte dans un tableau de chaînes de caractères. On souhaite compter le nombre de fois que chaque mot apparaît dans le texte.

- a. Définir un type construit permettant de compter facilement le nombre d'occurrences de chaque mot.
- b. Écrire une fonction qui prend en paramètre un tableau de mots et retourne une valeur de ce type. On pourra utiliser la méthode `lower()` des chaînes de caractères pour transformer les majuscules en minuscules.
- c. Écrire un programme qui lit un texte et affiche les mots qui apparaissent une seule fois dans le texte. On pourra créer le tableau `mots` des mots d'un texte contenu dans un fichier avec le code suivant :

```
import re
fichier = open("montexte.txt", "r")
mots = re.split(r'\W+', fichier.read())
```

31 Calculatrice

- Écrire un programme qui attend la saisie d'opérations de la forme `10 + 15` ou `5 * 25` et affiche leur résultat.
- Modifier le programme pour que l'on puisse également saisir des affectations de la forme `a = 10` puis utiliser `a` dans des opérations telles que `a + 15`.
- Ajouter enfin la possibilité d'écrire des affectations de la forme `b = a + 10`.

32 Jeux Olympiques

On veut représenter les résultats des épreuves de l'ensemble des jeux olympiques. Chaque épreuve a un nom, par exemple « Natation 400 m Femmes », une année, par exemple 2004, et le nom du champion, dans ce cas « Laure Manaudou ».

Proposer au moins deux représentations utilisant des tuples, des tableaux et/ou des dictionnaires pour pouvoir répondre facilement à des questions du type « Qui a gagné telle épreuve en telle année ». Laquelle vous paraît la plus efficace ?

33 Convertisseur

On veut convertir des mesures entre plusieurs unités, comme par exemple des centimètres en pouces (1 pouce = 2,54 cm). Toutes les conversions consistent en un facteur multiplicatif (on ne pourra donc pas convertir des degrés Centigrades en Fahrenheit, par exemple).

On veut définir trois fonctions :

- `ajouter_conversion(unite_depart, unite_arrivee, facteur)` pour ajouter une conversion qui permettra de convertir entre les deux unités ;
- `convertir(valeur, unite_depart, unite_arrivee)` pour afficher le résultat d'une conversion sous la forme « 10 pouce = 25,4 cm », ou « conversion non disponible » si la conversion n'a pas été définie ;
- `convertir_tout(valeur, unite_depart)` pour afficher toutes les conversions possibles de l'unité de départ.

- Définir un type construit pour représenter les conversions de façon à ce que le calcul d'une conversion soit le plus simple possible.
- Écrire un programme qui fournit une interface textuelle au convertisseur. L'utilisateur peut taper les commandes suivantes (`xxx` et `yyy` représentent des valeurs numériques et `u1`, `u2` des unités) :
 - ? pour avoir la liste des unités qui peuvent être converties ;
 - `xxx u1 = yyy u2` pour définir la conversion de `u1` vers `u2`, par exemple `1 pouce = 2,54 cm` ;

- `xxx u ?` pour afficher toutes les conversions de `u`, par exemple `12 pouce ?` ;
 - `xxx u1 en u2` pour convertir `xxx` de `u1` en `u2`, par exemple `12 pouce en cm`.
- c. [Niveau Titan, facultatif] On veut compléter automatiquement la table des conversions : s'il existe une conversion de l'unité A vers l'unité B et de l'unité B vers l'unité C (par exemple `cm` vers `pouce` et `pouce` vers `piéd`), on veut ajouter la conversion de A vers C (`cm` vers `piéd`). Modifier la fonction `ajouter_conversion` pour ajouter toutes les conversions possibles.

34 Recette de cuisine

Une recette de cuisine est constituée d'ingrédients et d'étapes de préparation. Les ingrédients sont décrits par des tuples (`quantité, unité, ingrédient`), par exemple `(150, "g", "farine")` ou `(0.5, "l", "lait")`. Les étapes sont décrites comme un tableau de chaînes de caractères, par exemple `["mélanger le lait et la farine", "ajouter les oeufs"]`.

On veut pouvoir afficher une étape en y associant la quantité des ingrédients qui y sont mentionnés, par exemple :

```
Etape 1 : mélanger le lait et la farine.
0.5l de lait
150g de farine
```

- Quel type construit utiliser pour représenter au mieux les ingrédients en vue de cet affichage ?
- Écrire une fonction `afficher_etape` qui affiche une étape selon le format ci-dessus.
- Écrire un programme qui attend que l'utilisateur tape sur la touche entrée pour afficher les étapes une par une. À tout moment l'utilisateur peut taper ? pour avoir la liste des ingrédients, ?? pour la liste des étapes, ou - pour revenir une étape en arrière.
- En utilisant le convertisseur de l'exercice précédent, traduire chaque quantité dans la première unité correspondante dans le convertisseur, si elle existe.

35 Emploi du temps

On souhaite gérer un emploi du temps hebdomadaire qui contient les horaires (début et fin) et matières de chaque créneau.

- Proposer deux représentations différentes de l'emploi du temps avec des types construits.
- Pour chaque représentation, écrire une fonction qui ajoute un créneau à l'emploi du temps et retourne `True` s'il est libre, ou sinon retourne `False` sans le modifier.

36 Dates

On définit une date par un tuple de trois nombres entiers : le jour, le mois, et l'année, par exemple, `(14, 7, 1789)`.

- Écrire une fonction `date_valide` qui prend un tuple en paramètre et retourne `True` si elle est valide, `False` sinon. Une date est valide si :
 - le tuple contient trois valeurs ;
 - le second nombre est compris entre 1 et 12 ;
 - le premier nombre est compris entre 1 et le nombre de jours du mois correspondant (Cf. exercice 22 du Chapitre 2). Si la date n'est pas valide, la fonction doit afficher la nature du problème (par exemple « Le mois 13 est trop grand. »).
- Écrire une fonction `nombre_jours` qui prend deux dates (tuples) en paramètres et retourne le nombre de jours entre la première et la seconde (ce nombre peut être négatif).

37 Contenu d'un dictionnaire

Écrire les fonctions `cles(dict)`, `valeurs(dict)` et `elements(dict)` qui font la même chose que les méthodes `dict.keys()`, `dict.values()` et `dict.items()` sans faire appel à celles-ci.

On pourra utiliser la méthode `append` pour ajouter des entrées à une liste.

38 Piloter la tortue

Comme nous l'avons vu au Chapitre 3 avec les fonctions de rappel, une fonction peut être utilisée comme valeur :

```
def double(n):
    return n*2
def appel(f, x):
    return f(x)
appel(double, 5) # 10
```

- Utiliser cette possibilité pour définir un type construit qui représente la trajectoire d'un tortue non pas sous forme de coordonnées, mais sous forme de fonction à appeler et de leurs paramètres.
- Utiliser ce type construit pour représenter la trajectoire suivante :

```
forward(100)
left(45)
pencolor('red')
forward(50)
```

- Écrire une fonction qui prend une trajectoire en paramètre et exécute son contenu afin de piloter la tortue.

39 Liste de tâches avec priorité

On définit une liste de tâches `todo_list` sous la forme d'un dictionnaire dont les clés sont les intitulés des tâches à effectuer et les valeurs des *tuples* indiquant (1) si la tâche est effectuée, (2) la date à laquelle elle doit être complétée au plus tard, et (3) une valeur de priorité allant de 1 (peu important) à 5 (très important).

- Écrire la fonction `ajouter_tache` qui prend un intitulé, une date limite et une priorité en paramètres et ajoute cette tâche à `todo_list` comme étant non effectuée. Si un paramètre est invalide, un message doit expliquer ce qui ne va pas.
- Ajouter la tâche « Coiffeur », de priorité 2, à effectuer au plus tard le 12/5/2022. Tester la fonction `ajouter_tache` en essayant diverses valeurs incorrectes pour ses paramètres.
- Écrire la fonction `decire_tache` qui prend en paramètre un intitulé de tâche et qui retourne une chaîne de la forme : « La tâche 'Coiffeur' de priorité 2 et à compléter au plus tard le 12/5/2023 [a/n'a pas] été effectuée. »
- Écrire la fonction `trop_tard` qui prend une date en paramètre, vérifie qu'elle est valide, et affiche l'ensemble des tâches non-effectuées dont la date limite est postérieure à cette date.
- Écrire une fonction `important` qui prend un niveau de priorité en paramètre et affiche les tâches non encore effectuées d'un niveau au moins aussi important.

40 Belote

Le jeu de la Belote ressemble à la Bataille, mais la couleur (Pique, Cœur, etc.) compte. À chaque partie, une certaine couleur est appelée « atout » et bat toutes les autres couleurs.



- Écrire une fonction `belote_simple` qui prend en paramètres l'atout courant et un dictionnaire de la forme `{joueur: carte}` qui indique la carte jouée par chaque joueur. Cette fonction doit retourner le nom du joueur qui remporte cette manche, sachant qu'une carte de la couleur de l'atout bat systématiquement une carte d'une autre couleur. Par exemple, si l'atout est Trèfle, un 2 de Trèfle bat un Roi de Carreau, et un 7 de Cœur bat un 6 de Pique. En cas d'égalité, l'un des joueurs ex-aequo l'emporte.
- Écrire une fonction `belote` qui fait la même chose que `belote_simple`, sauf qu'au sein de la couleur atout, l'ordre de victoire change : le Valet (11) devient la carte la plus forte, suivie du 9 ; les autres cartes, de l'As (14) au 2, gardent le même ordre.