Enseignant : Sofiane MAAZI

TP3 IS2A3 : Manipulations avancées (Python)

31 mars 2021

Exercice 1 : Collection ordonnée d'éléments

1. En Python une variable possède un type identique à la valeur qui lui est affectée. En modifiant la valeur d'une variable son type peut donc changer.
   (a) Créez une variable *mystere* égale à 2021 et donnez son type.
   (b) Affectez lui sucessivement les valeurs suivantes et analysez comment varie son type ($\Rightarrow$ *type*) : « 52.1 » ,« "Polytech" », « ["table","chaise"] et « 5+2j » .
2. Il est possible de définir d'autres types à partir de ceux identifiés précédement. En particulier, les collections ordonnées de plusieurs éléments : les listes, tuples et dictionnaires.
   (a) Créez une liste et un tuple qui contiennent chacun les jours de la semaine.
   (b) Les éléments d'une liste et d'un tuple sont identifiées par leur position en utilisant un nombre appelé indice. Sélectionnez 2 jours de la liste et du tuple. Les éléments d'un tuple sont-ils modifiables ?
   (c) Les dictionnaires permettent de rassembler des éléments et de les identifier grâce à une clé. Il s'agit du même principe qu'un dictionnaire de français où on accède à une définition (i.e valeur) grâce à un mot (i.e clé). Créez un dictionnaire avec le nom et la couleur de 3 créatures de la saga Harry Potter.
   (d) Ajoutez le nom et la couleur d'une 4$^{\text{ème}}$ créature au dictionnaire.
   (e) Proposez 2 méthodes pour parcourir le dictionnaire.

Exercice 2 : Nombres pseudo-aléatoires

Pour réaliser des simulations, le statisticien a besoin de suites de valeurs aléatoires. Il est très difficile à un ordinateur de nous en fournir ! On utilise subséquemment à la place des suites « pseudo-aléatoires ». Ce sont des suites dont le terme suivant est obtenu en appliquant une formule au terme précédent. Cette fonction est suffisamment simple pour que le calcul soit rapide. Elle est aussi choisie de sorte à ne pas boucler sur elle-même et donner lieu à une distribution uniforme (i.e un histogramme où toutes les barres ont la même hauteur). Les deux principales fonctions pour obtenir des nombres aléatoires sont *randint(a, b)* et *random()*.

1. Pour tester la qualité de *random()*, écrivez une fonction :
   (a) *monRandom(n, a, b)* qui tire « au hasard » n nombres compris entre a et b et retourne la moyenne et l'écart- type de la suite ainsi obtenue.
   (b) *testRandom(n, a, b)* qui appelle la première et affiche la moyenne et l'écart-type, et pour chacune de ces deux valeurs, affiche la valeur absolue de la différence entre valeur observée et attendue.

Remarque : si la distribution des valeurs renvoyées par *randomt()* est uniforme, quand n devient de plus en plus grand, m doit tendre vers 0.5 et σ vers 0.288675134595, c'est-à-dire 1/ 12 .

2. Pour tester la qualité de *randint()*, écrivez une fonction *monRandint(n, a, b)* qui tire « au hasard » n nombres entiers compris entre a et b, retourne un dictionnaire qui associe à chaque nombre ses occurences et affiche la probabilité $p_k = \frac{n_k}{n}$ pour $k \in [a; b]$.

Remarque : Si la distribution des valeurs renvoyées par *randint()* est uniforme, quand n grandit les probabilités tendent à être égales p_k (i.e : toutes les valeurs ont la même probabilité d'être tirées).

3. Écrivez une fonction *chaineADN(n)* qui renvoie une chaîne ADN aléatoire.
4. Écrivez une fonction *permut(liste)* qui renvoie une liste obtenue en permutant aléatoirement les éléments de la liste donnée.

Exercice 3 : Modèles démographiques

Le modèle démographique de Malthus est un modèle exponentiel d'évolution de l'effectif de la population. Il prévoit que l'effectif de la population décroît vers 0 si le taux de mortalité est supérieur au taux de natalité et croît vers l'infini si le taux de natalité est supérieur au taux de mortalité. Thomas Malthus (1766 - 1834) est l'auteur d'un *Essai sur le principe de la population*.

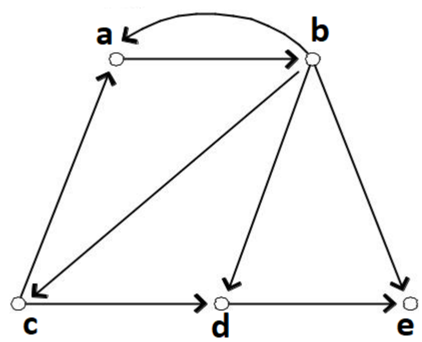
1. Soit un pays de 1 habitants où la population p double à chaque temps t. Écrire une fonction *geom(t)* qui renvoie la population du pays au temps t.
2. On suppose que les ressources du pays valent 1 et augmentent de 1 à chaque temps t. Écrire une fonction *arithm(t)* qui renvoie les ressources du pays au temps t.
3. Représentez sur un même graphique les courbes des fonctions précédentes.
4. On veut déterminer à quel temps t la population p atteint un seuil fixé k.
   (a) Calculez la plus grande valeur de t telle que $2^{\mathrm{t}} \leq k$.
   (b) En déduire le temps t pour lequel le pays comptera 1000 d'habitants.
   (c) En déduire les ressources du pays au temps t.
   (d) Commentez les résultats obtenus.
5. En 1838 le mathématicien belge Verhulst propose un modéle décrivant l'évolution de populations tendant vers un seuil maximal qui ne peut pas être dépassé. En raison des ressources limitées qui empêchent la population de croître davantage.
   (a) Importez le fichier *japon.csv*. La $1^{\text{ère}}$ variable donne le nombre d'années écoulées depuis 1900. La $2^{\text{ème}}$ variable donne l'effectif de la population du Japon (en millions) par paliers de 5 ans.
   (b) Expliquez la baisse de population de 1940 à 1945.
   (c) On décide de modéliser la population entre 1950 et 1980 par les 31 premiers termes d'une suite géométrique v_n vérifiant $v_0 = 83.6$ et $v_{30} = 116.8$. Donnez la raison q de cette suite. En déduire une estimation de la population du Japon en 1973.
   (d) Le modèle vous semble-t-il pertinent pour estimer la population du Japon de 1980 à 2010 ?
   (e) Expliquer en quoi la population actuelle du Japon donne raison au modèle Verhulst.

Exercice 4 : Graphes

Un graphe orienté est caractérisé par un ensemble de sommets et d'arcs. Chaque arc relie 2 sommets appelés respectivement origine et extrémité de l'arc. Une information unique est associée à chaque sommet. Elle est appelée « étiquette du sommet ». 2 sommets distincts n'ont donc pas la même étiquette. Deux représentations sont possibles pour un graphe :

— La liste d'arcs est une liste dont les élèments sont des couples (o,e). Chaque couple (o,e) représente un arc du sommet o vers le sommet e.
⇒ [('a', 'b'), ('b', 'a'), ('c', 'a'), ('c', 'd'), ('b', 'c'), ('b', 'd'), ('b', 'e'), ('d', 'e')]

— La liste d'adjacence est un dictionnaire dont les clés sont les étiquettes des sommets et la valeur associée à l'étiquette d'un sommet o est la liste des étiquettes des sommets adjacents (i.e : les sommets e tels qu'il existe un arc dont o est l'origine et e l'extrémité).
⇒ {'a' : ['b'], 'b' : ['a', 'c', 'd', 'e'], 'c' : ['a', 'd'], 'd' : ['e'], 'e' : [ ] }



Exemple de graphe

1. Soient une liste d'arcs (l_arc) et d'adjacence (l_adjacente). Écrivez la fonction :
   — *ens_sommets(l_arcs)* qui retourne l'ensemble (type set) des sommets du graphe.
   — *ens_successeurs(l_arcs,n)* qui retourne l'ensemble des sommets s tel qu'il existe un arc de n vers s.
   — *ens_predecesseurs(l_arcs,n)* qui retourne l'ensemble des sommets s tel qu'il existe un arc de s vers n.
   — *puits(l_arcs)* qui retourne l'ensemble des sommets sans successeurs.
   — *sources(l_arcs)* qui retourne l'ensemble des sommets sans prédécesseurs.
2. Écrivez une fonction *conver1(l_arcs)* qui retourne une représentation sous forme de liste d'adjacence.
3. Implémentez les 5 fonctions de la première question en prenant comme argument une liste d'adjacence.
4. Quel argument vous semble la plus approprié pour chacune des 5 fonctions ?
5. Écrivez une fonction *conver2(l_ajacente)* qui retourne une représentation sous forme de liste d'arcs.
6. Testez soigneusement vos 12 fonctions avec le graphe de l'exemple et le graphe suivant [('a', 'b'), ('a', 'c'), ('d', 'e'), ('f', 'e')] .