9 – CALCULABILITÉ – DÉCIDABILITÉ – RÉCURSIVITÉ - MODULARITÉ

Exercices

1

Compléter la table de transition d'une machine de Turing, présentée ci-dessous, qui doit effectuer une multiplication par 2 d'un nombre écrit en binaire.

État	Lecture	Écriture	Déplacement	État suivant
E1	blanc	blanc	gauche	E2
E2	0	...	...	...
	1	...	...	...
	blanc	...	...	...

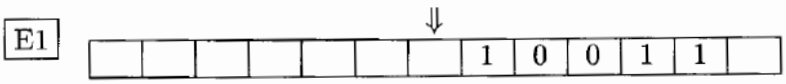
Le déplacement est celui du ruban et le nombre à multiplier est écrit en binaire sur ce ruban. Par exemple, si le nombre à multiplier est 11, soit 1011 en binaire, la situation de départ, avec la machine dans l'état E1, est la suivante :

2

La table de transition d'une machine de Turing est donnée par le tableau suivant :

État	Lecture	Écriture	Déplacement	État suivant
E1	blanc	blanc	gauche	E2
E2	0	0	gauche	E2
	1	1	gauche	E2
	blanc	blanc	droite	E3
E3	0	1	droite	Fin
	1	0	droite	E3
	blanc	1	droite	Fin

1. Quel est résultat s'il est écrit 10011 sur le ruban ?
La machine est dans l'état E1 et la position initiale de la tête de lecture est la suivante :



2. Écrire un programme correspondant en Python sur les modèles des fonctions `machine` du cours. Tester avec un ruban `r` de valeur initiale `[None, 1, 0, 0, 1, 1, None, None]` puis avec un ruban `r` de valeur initiale `[None, 1, 1, 1, 1, None, None]`.

3

On considère le script suivant :

```
def somme(n) :
  if n == 0:
    return 0
  else:
    return (n + somme(n-1))
```

1. Suivre le déroulement du programme pour n = 3, en explicitant chaque appel à l'instruction return.
2. Que se passe-t-il si n < 0 ? Proposer plusieurs solutions pour résoudre ce problème.
3. Même question pour le cas où n n'est pas un entier.

4

Ecrire en Python une fonction récursive *inverse* qui prend en paramètre une chaîne de caractères *ch* et renvoie la chaîne obtenue en inversant l'ordre des caractères. Par exemple, inverse('azerty') renvoie 'ytreza'.

5

Soit u_n la suite d'entiers définie par :

$$\begin{array}{rcll} u_{n+1} & = & u_n/2 & \text{si } u_n \text{ est pair,} \\ & = & 3 \times u_n + 1 & \text{sinon.} \end{array}$$

avec u_0 un entier quelconque plus grand que 1.

Écrire une fonction récursive `syracuse(u_n)` qui affiche les valeurs successives de la suite u_n tant que u_n est plus grand que 1.

La conjecture de Syracuse affirme que, quelle que soit la valeur de u_0, il existe un indice n dans la suite tel que $u_n = 1$. Cette conjecture défie toujours les mathématiciens.

6

On considère la suite u_n définie par la relation de récurrence suivante, où a et b sont des réels quelconques :

$$u_n = \begin{cases} a \in \mathbb{R} & \text{si } n = 0 \\ b \in \mathbb{R} & \text{si } n = 1 \\ 3u_{n-1} + 2u_{n-2} + 5 & \forall n \geq 2 \end{cases}$$

Écrire une fonction récursive `serie(n, a, b)` qui renvoie le `n`-ème terme de cette suite pour des valeurs `a` et `b` données en paramètres.

7

La courbe de Koch est une figure qui s'obtient de manière récursive. Le cas de base à l'ordre 0 de la récurrence est simplement le dessin d'un segment d'une certaine longueur l, comme ci-dessous (figure de gauche).

Le cas récursif d'ordre n s'obtient en divisant ce segment en trois morceaux de même longueur $l/3$, puis en dessinant un triangle équilatéral dont la base est le morceau du milieu, en prenant soin de ne pas dessiner cette base. Cela forme une sorte de chapeau comme dessiné sur la figure de droite ci-dessus. On réitère ce processus à l'ordre $n-1$ pour chaque segment de ce chapeau (qui sont tous de longueur $l/3$). Par exemple, les courbes obtenues à l'ordre 2 et 3 sont données ci-dessous (à gauche et à droite, respectivement).

Écrire une fonction `koch(n, l)` qui dessine avec `Turtle` un flocon de Koch de profondeur n à partir d'un segment de longueur l.

8

Tester le programme ci-dessous puis en écrire une version récursive.

```
from turtle import *

couleurs = ['blue', 'green', 'yellow', 'orange', 'red', 'purple']
bgcolor('black')

def dessin():
    for i in range(180):
        color(couleurs[i%6])
        forward(i)
        right(59)

dessin()
```

9

Créer un module *stats* qui contient deux fonctions *somme* et *moyenne*. Ces deux fonctions prennent une liste de nombres non vide en paramètre. La fonction `somme` renvoie la somme des éléments de la liste et la fonction `moyenne` renvoie la moyenne des éléments de la liste. Tester le module en l'important dans un autre fichier où les deux fonctions sont utilisées.

Il est possible ensuite de compléter le module `stats` avec d'autres fonctions qui peuvent être utiles en statistique.

10

Création d'un module

On considère un fichier **dessin.py** qui contient le code suivant :

```
from turtle import *

import figures as f

up()
goto(50, -60)
down()
f.rectangle(100, 75, 'red')

up()
goto(-80, -20)
down()
f.triangle(85, 'blue')

up()
goto(20, 60)
down()
f.cercle(60, 'green')
```

Il s'agit d'écrire le fichier **figures.py** qui est importé dans le programme. Lorsque le programme est exécuté, on obtient dans la fenêtre graphique du module Turtle un rectangle, un triangle et un cercle avec des couleurs différentes. De plus le périmètre et l'aire de chaque figure sont affichés avec le dessin.

Les principales commandes du module Turtle sont résumées à ces deux adresses :
https://fr.wikibooks.org/wiki/Programmation_Python/Turtle
https://docs.python.org/fr/3.8/library/turtle.html.

On veillera à **documenter** le module.

11

En utilisant un interpréteur Python, rechercher dans un module nommé **statistics**, inclus dans la bibliothèque standard, s'il existe une fonction calculant la moyenne de plusieurs nombres et une fonction déterminant la médiane.

Comment utiliser ces fonctions pour calculer la moyenne et la médiane de plusieurs nombres ?

12

Utilisation d'une API

Consulter le site à l'adresse http://api.open-notify.org/ qui concerne l'ISS, la station spatiale internationale. Utiliser le module *requests* pour écrire un code en Python pour :

1. obtenir les personnes à bord de l'ISS ;
2. obtenir la position de l'ISS ;
3. obtenir les passages de l'ISS au-dessus d'un point.