Il existe en Python de nombreuses fonctions prédéfinies qui ne sont pas automatiquement chargées mais qui sont stockées dans des fichiers appelés modules. Pour utiliser de telles fonctions, il faut au préalable importer le module qui les comporte.

Pour charger un module, on utilise la syntaxe suivante :

```
from nom_du_module import *
```

Il existe de très nombreux modules. Nous allons en évoquer seulement deux ici.

### 4.1 *Le module* `math`

Le module `math` contient toutes les constantes et les fonctions (au sens mathématiques) dont nous aurons besoin. On le charge à l'aide de la syntaxe suivante :

```
from math import *
```

Il contient en particulier :

- `pi` : renvoie une valeur approchée de $\pi$ à $10^{-16}$ près ;
- `sqrt(x)` : renvoie la racine carrée du nombre $x$ ;
- `abs(x)` : renvoie la valeur absolue du nombre $x$ ;
- `cos(x)` : renvoie le cosinus du nombre $x$ (en radians) ;
- `sin(x)` : renvoie le sinus du nombre $x$ (en radians) ;
- `e` : renvoie une valeur approchée du nombre e à $10^{-16}$ près ;
- `exp(x)` : renvoie l'image de $x$ par la fonction exponentielle (i.e. le nombre $e^x$) ;
- `log(x)` : renvoie l'image de $x$ par la fonction logarithme népérien (i.e. le nombre $\ln(x)$) ;
- `log10(x)` : renvoie l'image de $x$ par la fonction logarithme décimal (i.e. le nombre $\log(x)$).

### 4.2 *Le module* `random`

Le module `random` est un module qui permet d'engendrer des nombres pseudo-aléatoires.

On le charge à l'aide de la syntaxe suivante :

Il contient en particulier les fonctions :

- `random()` qui renvoie un nombre flottant pseudo-aléatoire entre 0 et 1 ;
- `randint(a,b)` qui renvoie un entier pseudo-aléatoire compris (au sens large) entre les deux entiers a et b.

### 4.3 *Exercices*

♦ **PYT.1**

**1)** En utilisant la fonction `sqrt`, écrire une fonction `carre_parfait` qui renvoie `True` si l'entier $n$ passé en argument est le carré d'un entier et `False` sinon.

**2)** En utilisant la fonction précédente, écrire une fonction `somme_carrés` qui renvoie la somme des carrés d'entiers compris entre 1 et $n$ où $n$ est un entier passé en argument. Ainsi, `somme_carré(10)` renvoie $1 + 4 + 9$ c'est-à-dire 14.

♦ **PYT.2** Écrire une fonction `pile_ou_Face` qui renvoie au hasard et de manière équiprobable pile ou face.

♦ **PYT.3** Écrire une fonction `lancer_de_dé` qui renvoie au hasard et de manière équiprobable un nombre entier entre 1 et 6 (compris).

♦ **PYT.4** Lorsqu'on joue au Monopoly, on lance deux dés au hasard et on calcule la somme des valeurs obtenues.

En utilisant la fonction `lancer_de_dé` de l'exercice 31, écrire une fonction `tirage_Monopoly` qui renvoie le résultat d'un tel tirage.

♦ **PYT.5** En utilisant la fonction `lancer_de_dé` de l'exercice 31, écrire une fonction `simulation` prenant en argument un entier naturel $n$ non nul, qui simule $n$ lancers successifs d'un dé cubique équilibré et qui renvoie la fréquence de 6 obtenus (c'est-à-dire le nombre de 6 obtenus divisé par le nombre total de lancers).

♦ **PYT.6** En utilisant la fonction `lancer_de_dé` de l'exercice 31, écrire une fonction qui simule une répétition de lancers de dé et renvoie le nombre de lancers nécessaires pour obtenir un 6 pour la première fois.

♦ **PYT.7** La suite de Syracuse est une suite de nombres entiers définie par une valeur initiale $u_0$ et par la relation de récurrence :

$$\forall n \in \mathbf{N}, \quad u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{sinon} \end{cases}$$

**1)** Écrire une fonction `syracuse` qui prend en arguments deux entiers naturels $n$ et $a > 0$ et qui renvoie la valeur de $u_n$ lorsque $u_0 = a$. On pourra utiliser la fonction `est_pair` de l'exercice 10.

**2)** Une conjecture (non encore démontrée à ce jour) postule que cette suite finit toujours par atteindre le nombre 1. La suite des valeurs de $u_n$ partant de $u_0$ et allant jusqu'à la première apparition de 1 s'appelle un vol.

En utilisant la fonction `syracuse`, écrire une fonction `vol` qui prend en argument un entier $a > 0$ et qui renvoie toutes valeurs prises par $(u_n)$ lors du vol lorsque $u_0 = a$.

**3)** On appelle temps de vol la plus petite valeur de $n$ telle que $u_n = 1$. Écrire une fonction `temps_vol` prenant en argument un entier $a > 0$ et qui renvoie le temps de vol pour la suite $(u_n)$ telle que $u_0 = a$.

**4)** Lors d'un vol, on appelle altitude maximale la plus grande valeur prise par la suite $(u_n)$.

Écrire une fonction `altitude_max` qui prend en argument un entier $a > 0$ et qui renvoie l'altitude maximale pour la suite $(u_n)$ telle que $u_0 = a$.