Partie 1

Rire aléatoire

1.1 Enoncé

La fonction `random` de la bibliothèque `random` produit un nombre aleatoire entre 0 et 1.

```
> from random import random
> random()

0.8324575544825609
```

On peut egalement demander un entier aleatoire entre 0 et 100 avec la fonction `int` :

```
> int(100*random())

62
```

1. Ecrire un programme qui affiche de manière aléatoire la chaine de phonèmes “Ha”, “HaHa”, “HaHaHaHa” ou “HaHaHaHaHa”, avec “Ha” répété un nombre aléatoire de fois, entre 1 et 10.
2. On veut maintenant que le programme affiche un rire aléatoire pouvant aussi comporter des séquences de “Ho”, comme par exemple : “HaHaHaHaHoHoHoHoHaHaHa...”. La séquence devra avoir une longueur inférieure à 50 phonèmes.

1.2 Aide

On utilise de préférence une boucle non bornée avec comme condition :

```
while nombre_sons < nombre_max:
```

Il faudra donc définir les variables : `nombre_sons`, `nombre_max`, `text` (pour stocker la suite de phonèmes).

On y ajoute les variables qui stockent la valeur aléatoire du nombre de ‘Ha’ et de ‘Ho’ : `nombre_son1` et `nombre_son2` que l’on calcule à chaque itération avec

`nombre_son1 = int(10*random())`

La variable `text` est construite en **ajoutant** à chaque itération des phonèmes, avec

```
text += son1 * nombre_son1
```

1.3 Correction partielle

```
from random import random

def rire(son1,son2):
    nombre_sons = 0
    nombre_max = 50
    text = ""
    while nombre_sons < nombre_max:
        nombre_son1 = int(10*random())
        text += son1 * nombre_son1
        nombre_sons += nombre_son1

        nombre_son2 = int(10*random())
        text += son2 * nombre_son2
```

```
        nombre_sons += nombre_son2

    return text

rire('Ha','Ho')
```

Partie 2

Calendrier

2.1 Enoncé

1. Ecrire un programme qui indique si une année **a** est bissextile. Une année est bissextile si elle est divisible par 4.

2. Compléter le programme pour calculer le nombre de jours du mois **m**. **m** étant un entier compris entre 1 et 12. De janvier à juillet, les mois impairs ont 31 jours, les autres 30. Sauf le mois de fevrier a 29 ou 28 selon que l'année est bissextile ou non. Pour les autres mois, du mois d'aout à décembre c'est l'inverse (31 jours pour les **m** pairs).

2.2 Aide

la variable an est multiple de 4 si

```
an % 4 == 0
```

et de 2 (parité) si :

```
an % 2 == 0
```

On aura de plus besoin de la structure conditionnelle avec :

```
if mois == 2:
    ...
elif mois < 8 and mois % 2 == 1:
```

2.3 Correction partielle

```
def bissextile(an):
    return an % 4 == 0

def jours_mois(an,mois):
    if mois == 2:
        if bissextile(an):
            jours = 29
        else:
            jours = 28
    elif mois < 8 and mois % 2 == 1:
        jours = 31
    elif mois < 8 and mois % 2 == 0:
        jours = 30
    elif mois >= 8 and mois % 2 == 1:
        jours = 30
    else :
        jours = 31
    return jours

jours_mois(2021,10)
```

Partie 3

Fonction sin

3.1 Enoncé

1. Ecrire un programme qui utilise une boucle pour afficher les valeurs de la fonction `sin` pour `x` entre 0 et π

- On découpera l'intervale $[0\,;\pi]$ de façon à afficher 15 valeurs de `sin(x)`.
- Pour utiliser la fonction `sin`, il faudra l'importer avec la librairie math :

Exemple d'import de la fonction sin et de la constante PI en console :

```
> from math import sin,pi
> sin(pi/2)

1.0
```

Exemple de valeurs obtenue pour sin(x), où x est dans $[0\,;\pi]$

```
0.0
0.20791169081775931
0.40673664307580015
...
0.2079116908177593
5.66553889764798e-16
```

3.2 Aide

Pour obtenir 15 valeurs réparties de 0 à PI, il faut utiliser une boucle bornée avec `range(16)`. Le variant de boucle vaut alors {0, 1, 2, ... 15}.

On veut établir une correspondance sur {0, PI/15, 2*PI/15, ... PI}

On peut alors construire une relation :

$$x \rightarrow x * pi/15$$

```
for x in range(16):
  print(sin(x*pi/15))
```

3.3 Correction partielle

```
from math import sin,pi
def sin_eval():
    for x in range(16):
        print(sin(x*pi/15))
sin_eval()
```

3.4 Enoncé

2. Pour afficher le graphe de la fonction `sin`, on peut remplacer chaque valeur calculée précédement par une barre horizontale de longueur proportionnelle à sa valeur. On pourra par exemple multiplier le résultat de `sin(x)` par 30, et utiliser la fonction `int` pour transformer le nombre en entier, afin d'afficher un nombre de barres entre 0 et 30 : `int(30 * sin(x))`

Utiliser le programme pour afficher un graphique qui aura l'allure suivante :

```
=====
===========
================
...
================
===========
=====
```

3.5 Correction partielle

```python
def sin_trace():
    for x in range(16):
        y = sin(x*pi/16)
        n = int(y*30)
        segment = 30 * " " + n * "="
        print(segment)

sin_trace()
```

Partie 4

Jeu de hasard

4.1 Enoncé

1. Ecrire un programme qui, étant donné un nombre entre 2 et 12, affiche toutes les combinaisons possibles permettant d'obtenir ce nombre avec 2 dés.

Par exemple, pour obtenir 7, il pourrait afficher :

```
'1 ET 6 , 2 ET 5 , 3 ET 4 , 4 ET 3 , 5 ET 2 , 6 ET 1 , '
```

Aide : Pour faire une boucle bornée, avec un variant qui va de 1 à 6, on peut faire :

```python
for i in range(1,7):
  ...
```

2. Etendre ce programme pour afficher, pour chaque nombre entre 2 et 12, toutes les combinaisons possibles permettant d'obtenir ce nombre avec les 2 dés.

Par exemple :

```
Pour obtenir 2, on peut faire 1 ET 1 ,
Pour obtenir 3, on peut faire 1 ET 2 , 2 ET 1 ,
Pour obtenir 4, on peut faire 1 ET 3 , 2 ET 2 , 3 ET 1 ,
...
Pour obtenir 11, on peut faire 5 ET 6 , 6 ET 5 ,
Pour obtenir 12, on peut faire 6 ET 6 ,
```

4.2 Aide

On va utiliser 2 boucles imbriquées :

- une premiere boucle bornée pour le premier dé, où le variant i prend successivement les valeurs {0, 1, ... 6}
- une deuxième boucle bornée pour le deuxième dé, où le variant j prend successivement les valeurs {0, 1, ... 6}

On affiche alors les valeurs de i et de j lorsque i = j vaut N. Il faut une exprression conditionnelle `if [condition]` :

4.3 Correction partielle

```
def combinaisons2D(N):
    combinaison = ""
    for i in range(1,7):
        for j in range(1,7):
            if i + j == N:
                combinaison += str(i) + ' ET ' + str(j) + ' , '
    return combinaison

def toutes_combi2D():
    for r in range(2,13):
        print('Pour obtenir {}, on peut faire {}'.format(r, combinaisons2D(r))
    )
```