université angers
FACULTÉ DES SCIENCES
Unité de formation et de recherche

Département Informatique
L1 Informatique
Jean-Michel Richer
Développement Python
2025/2026

jean-michel.richer@univ-angers.fr

Contrôle Continu L1 Python Avril 2026

Vous avez le droit de consulter mon site web ou la documentation Python ainsi que vos exercices de TP au **format papier**. Il est interdit de faire appel à des outils comme ChatGPT. Tout programme qui ne s'exécute pas correctement (erreur, exception) est compté pour 0.

Vous devez rendre quatre fichiers qui correspondent chacun à une question. Si vous ne répondez pas à une question ne rendez pas le fichier correspondant.

Arrivé en période P4 vous devez maitriser la compression et la décompression d'archive.

Rendre au maximum 4 fichiers :

- geo.py
- fibprimes.py
- mathzero.py
- auto.py

Exercice 1 - 5 pts -

Vous disposez dans l'archive d'un fichier **pays.txt**, qui contient des commentaires introduits par le symbole '#' en début de ligne et des informations pour différents pays européens :

- nom du pays
- superficie en km^2
- nombre d'habitants en millions
- nombre de frontières avec d'autres pays européens ou non européens

Les champs sont séparés par le caractère ';'.

On vous demande d'écrire un programme **geo.py** afin d'afficher la liste des pays par ordre alphabétique et de calculer le nombre total d'habitants de ces pays. Le résultat à afficher est le suivant :

```
==============================
Allemagne
Belgique
Espagne
France
Italie
Pays-Bas
Pologne
Portugal
Royaume-Uni
Suisse
==============================
411 millions d'habitants
```

Conseils : Pensez à utiliser les fonctions sur les chaînes de caractères comme **split**.

Exercice 2 - 5 pts -

On cherche à savoir si parmi les nombres de la suite de **Fibonacci** il existe des nombres premiers. On rappelle que la suite de Fibonacci est définie par :

- $F(0) = 0$
- $F(1) = 1$
- $F(n) = F(n-1) + F(n-2)$

Les premiers nombres de la suite sont 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89. Parmi ces nombres 2, 3, 5, 13, 89 sont premiers.

Créer le programme `fibprimes.py`.

1. Génération des nombres de la suite de Fibonacci : dans un premier temps on vous demande d'écrire une fonction `liste_fib(n : int) -> list` qui crée et retourne la liste des nombres de la suite de Fibonacci de 0 à n (inclus). Pour cela on crée une $liste$ initialement composée de `[0, 1]` qui sont les premiers termes de la suite. Puis on itère sur une variable i variant de 2 à n (inclus) et on ajoute à la liste le terme $liste[i-1] + liste[i-2]$ à chaque itération.

2. Détection d'un nombre premier : dans un second temps, on crée une fonction `est_premier(n : int) -> bool` implantée de la façon suivante :

1. si $n \leq 1$, on retourne False (0 et 1 ne sont pas premiers)
2. si $n \leq 3$, on retourne True (2 et 3 sont premiers)
3. si n est multiple de 2 ou n est multiple de 3, on retourne False
4. on définit une variable $i = 5$ et tant que $i * i <= n$:
   - si n est divisible par i ou n est divisible par $(i+2)$, on retourne False
   - on incrémente i de 6
5. finalement, en sortie de boucle, on retourne True, car aucun diviseur n'a été trouvé

3. Fonction main dans un troisième temps, on crée une fonction `main()` qui :

1. crée la liste des nombres de la suite de fibonacci de 0 à 83 (inclus) par appel de la fonction `liste_fib`
2. affiche uniquement les nombres premiers de cette suite ainsi :

```
fib(3) = 2 est premier
fib(4) = 3 est premier
fib(5) = 5 est premier
fib(7) = 13 est premier
fib(11) = 89 est premier
fib(13) = 233 est premier
fib(17) = 1,597 est premier
fib(23) = 28,657 est premier
fib(29) = 514,229 est premier
fib(43) = 433,494,437 est premier
fib(47) = 2,971,215,073 est premier
fib(83) = 99,194,853,094,755,497 est premier
```

Exercice 3 - 5 pts -

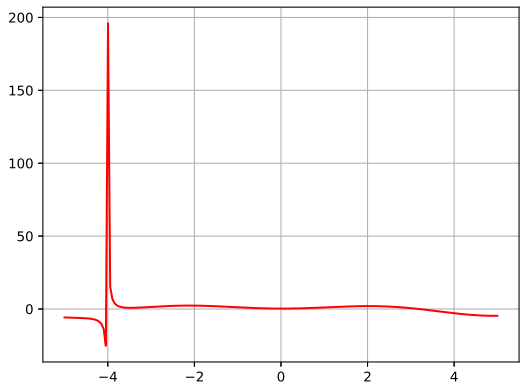
Soit la fonction suivante :

$$f(x) = sin(x) \times x + 1/(x+4)$$

Ecrire un programme **mathzero.py** permettant

1. d'afficher la fonction dans l'intervalle [-5,5[.
2. de trouver la valeur x_0 pour laquelle la fonction s'annule dans l'intervalle $[2, 4]$.

Résultat attendu : le graphique + la valeur de x_0




```
La fonction f s'annule en f(3.185298874974251)=-3.4582278152228696e-09
```

Attention : la valeur de x_0 obtenue dépend de la méthode utilisée et de la précision.

Exercice 4 - 5 pts -

Un nombre auto-descriptif est un nombre qui se décrit lui-même. Par exemple 21200 contient :

- 2 chiffres '0'
- 1 chiffres '1'
- 2 chiffres '2'
- 0 chiffres '3'
- 0 chiffres '4'

Ce qui donne bien 21200 ☺

Créer un programme `auto.py` qui détermine si des nombres sont auto-descriptifs. On vous demande d'écrire également une fonction `autod(x:int) -> bool`, qui détermine si un nombre x est auto-descriptif ou non. Cette fonction crée deux tableaux :

- **digits**, initialement vide qui contiendra les chiffres qui composent le nombre x
- **counts**, qui contiendra le nombre d'occurrences de chaque chiffre, initialement il contient 10 fois la valeur 0

Pour remplir **counts** et **digits**, on procède ainsi :

Tant que x n'est pas égal à 0,

1. on calcule **r** le reste de la division de x par 10, puis on remplace x par le résultat de sa division par 10 (division entière)
2. on ajoute 1 au compteur **counts[r]**
3. on ajoute **r** au **début** de la liste **digits**

Pour établir si le nombre est auto-descriptif, il faut que les listes **counts** et **digits** soient identiques sur leurs **len(digits)** premiers éléments. Par exemple pour 21200, on doit avoir :

```
counts= [2, 1, 2, 0, 0, 0, 0, 0, 0, 0]
digits= [2, 1, 2, 0, 0]
```

Dans la fonction `main()`, tester si les nombres de la liste suivante sont auto-descriptifs ou non :

```
liste = [1210, 21000, 21200, 123405]
```

Résultat attendu, **attention** les nombres sont affichés sur 6 chiffres :

```
  1210 est auto-descriptif ? True
 21000 est auto-descriptif ? False
 21200 est auto-descriptif ? True
123405 est auto-descriptif ? False
```