## Table des matières :

[EXERCICE 1](#_oqahslca0tq)

Méthode

Jeux d’essais

[EXERCICE 2](#_pb6mslur0tms)

Méthode

Jeux d’essais

[EXERCICE 3](#_aa1qwhkh2h9f)

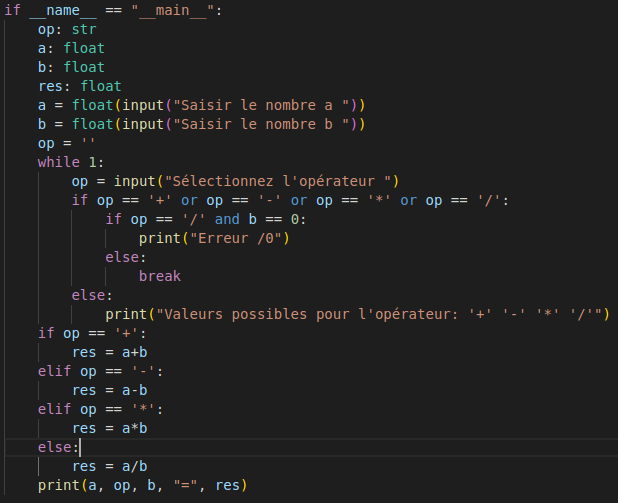
Méthode

Jeux d’essais

* Avec un année positive
* Avec une année négative
* Cas particulier

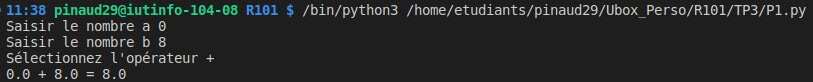
## **EXERCICE 1 :**

### **Méthode :**

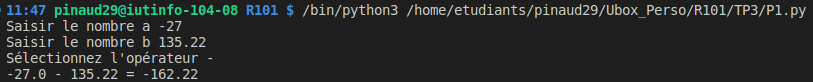


On saisit les deux nombres (type float), puis l’opération sous forme d’un caractère. On fait boucler le programme jusqu’à l'obtention d’un caractère valide ou d’une division possible (avec b différent de 0). On effectue l’opération correspondante et l’on affiche le résultat.

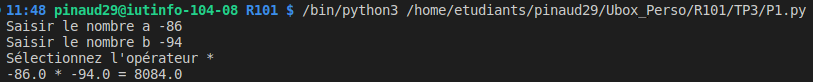
### **Jeux d’essais :**



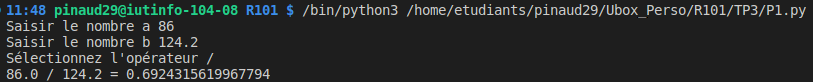
Avec 0 et un nombre positif, l’opération +



Avec un nombre négatif et un nombre à virgule, l’opération -



Avec l’opérateur \*



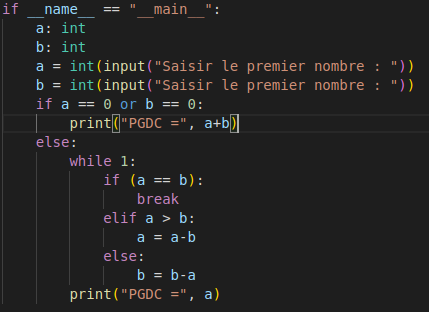
Avec l’opérateur /



Erreur en cas de division par 0 puis en cas de sélection d’un caractère ne correspondant à aucun opérateur valide.

## **EXERCICE 2 :**

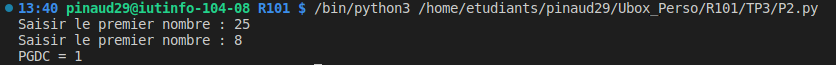
### **Méthode :**



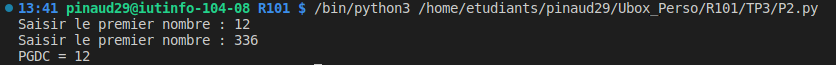
Le PGDC n’est possible qu’entre deux entiers, on utilise donc le type int pour nos variables.

Le problème principal de cet algorithme s’illustre avec le choix d’un grand nombre et d’un petit. En effet, le nombre de soustractions à effectuer est parfois si grand que le programme prendrait des années à donner le résultat. On gère le zéro indépendamment pour éviter de faire boucler le programme indéfiniment.

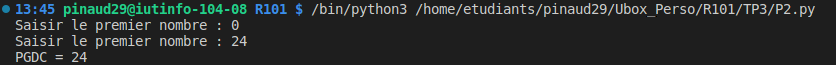
**Jeux d’essais :**



Cas avec deux nombre différents de 0 et a > b



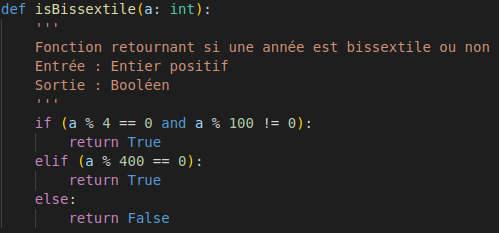
Cas avec deux nombre différents de 0 et b > a



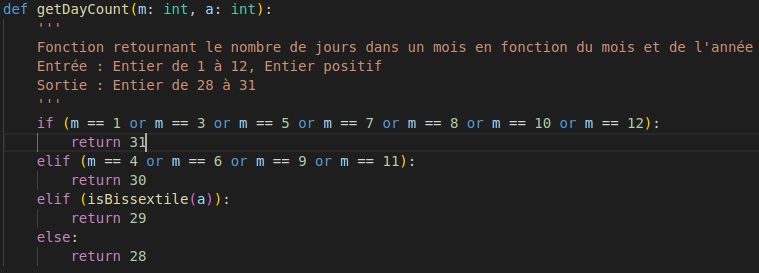
Cas avec 0

## **EXERCICE 3 :**

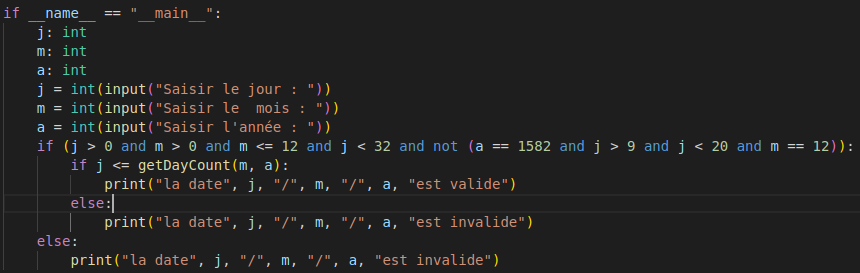
### **Méthode :**



On définit une fonction chargée de vérifier le statut de l’année choisie. L’avantage de l’opération % est qu’elle permet d’utiliser des années négatives



On définit une autre fonction chargée de compter le nombre de jours dans chaque mois. Elle utilise l’année en paramètre d’entrée et la fonction précédemment déclarée pour vérifier l’état du mois de février (28 ou 29 jours).



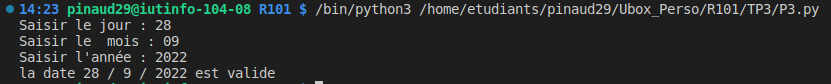
On utilise une première condition pour retirer les valeurs fausses dans tous les cas :

* les jour négatifs
* les mois négatifs
* les mois supérieurs à 12
* les jours supérieurs à 31
* les 10 jours inexistants du calendrier (en France)

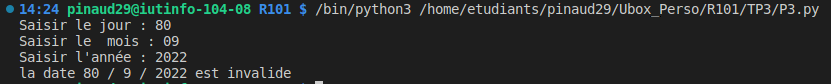
Puis on utilise la fonction définie plus tôt ‘getDayCount()’ pour vérifier que le jour saisi, bien qu’inférieur à 31, ne dépasse pas la limite inhérente à chaque mois.

**Jeux d’essais :**

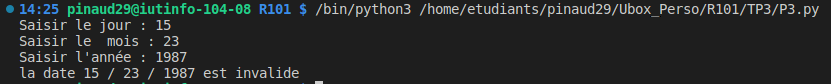
**Avec année positive**



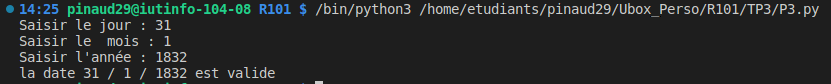
Test avec la date du jour



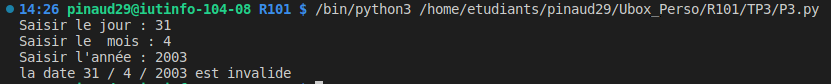
Test avec un jour supérieur à 31



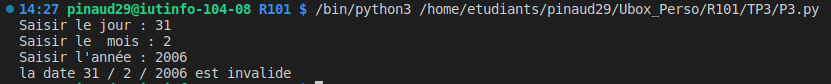
Test avec un mois supérieur à 12



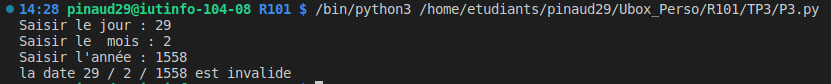
Test avec 31 jours un mois comportant ce nombre de jours



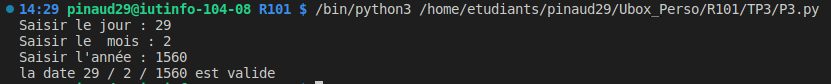
Test avec 31 jours un mois comportant 30 jours



Test avec 31 jours un mois en comportant 28 ou 29

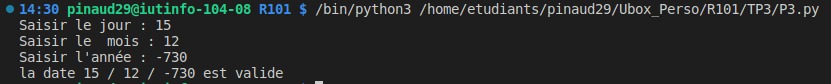


Test avec 29 jours un mois de février d’une année non bissextile

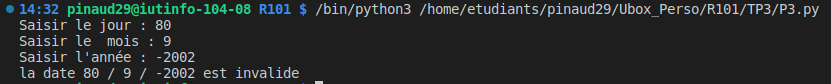


Test avec 29 jours un mois de février d’une année bissextile

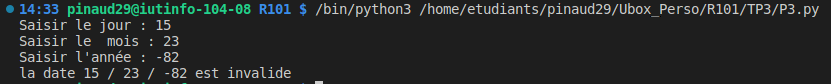
**Avec année négative**

****

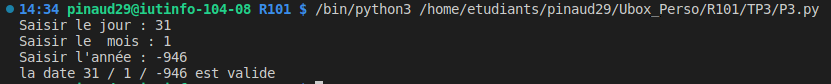
Test une date sans cas particulier



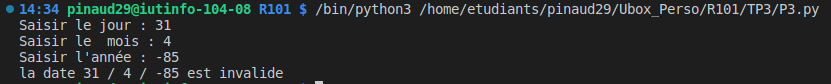
Test d’une date avec un jour supérieur à 31



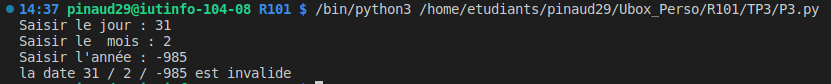
Test avec un mois supérieur à 12



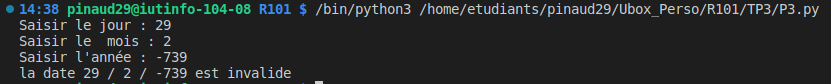
Test avec 31 jours un mois comportant ce nombre de jours



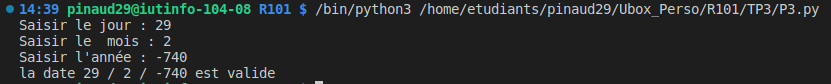
Test avec 31 jours un mois comportant 30 jours



Test avec 31 jours un mois comportant 28 ou 29 jours

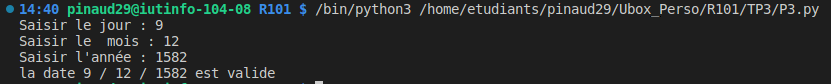


Test avec 29 jours un mois de février d’une année non bissextile

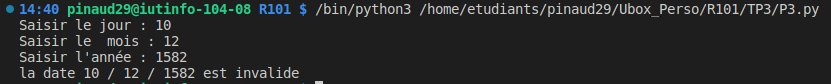


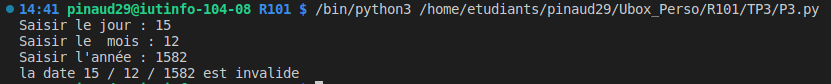
Test avec 29 jours un mois de février d’une année bissextile

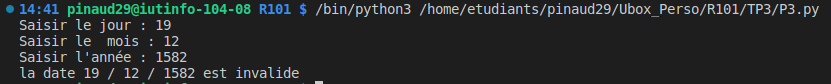
**Cas particulier**



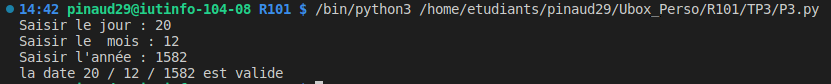
Valide le 9 décembre 1582







Invalide du 10 au 19 décembre 1582



Valide le 20 décembre 1582