Partie 1: Bases

I Introduction

1 Focus de cette partie

- Introduction
- variables, types, structures de contrôle

2 Présentation générale

2.1 Différences par rapport à la programmation

- langage “abstrait”
- permet de se concentrer sur le problème à résoudre, pluôt que sur les subtilités d’un “vrai” langage

2.2 Utilité d’un algorithme

- prendre une décision
- calculer une valeur
- transformer/aggréger des données

2.3 Problématiques

- correction: on veut la bonne valeur en sortie !
- gestion des données en entrée: s’assurer qu’elles correspondent à ce qui est attendu
- temps pris par l’algorithme quand on l’exécute
- lisibilité: est ce que l’on arrive à comprendre l’algorithme en le lisant

Info

On utilisera le langage Python pour écrire les algorithmes. Ce langage est bien adapté pour cette utilisation, et permettra de pouvoir exécuter les algorithmes “pour de vrai”.

On n’utilisera cependant pas toutes les fonctionnalités du langage: on se restreindra aux fonctionnalités présentées en cours.

3 Resources

- Banque d’exercices: https://perso.liris.cnrs.fr/pierre-antoine.champin/enseignement/algo/exercices/index.html
- Environnement Jupyter lab: https://sapristi.github.io/cours_algo/lab/index.html

II Références

1 Valeurs

1.1 Constantes

Types de base:

- `int`: nombres entiers: `1`, `0`, `-10`, etc
- `float`: nombres à virgule: `2,3`, `0.5`, etc
- `bool`: valeurs booléennes: `True` / `False`
- `str`: chaînes de caractères: `"abc"`
- `None` : absence de valeur : `None`

1.2 Variables

Une variable contient une valeur

1.2.1 Déclaration

Déclare une variable pour un usage futur. Doit toujours être accompagné d'un type.

```
myvar: int
message: str
```

1.2.2 Assignation

```
myvar = 2
myvar = 2 + test
```

1.2.3 Déclaration et assignation

```
myvar: int = 0
message: str = "bonjour"
```

2 Structures de contrôle

2.1 if / else

```
if condition:
  ...
else:
  ...
```

2.2 While

```
while condition:
  ...
```

3 Opérateurs

3.1 Numériques

- Opérateurs basiques `+`, `-`, `*`, `/`
- Opérateur `%`: reste division entière (uniquement entre deux entiers)

3.2 Booléens

- `and`, `or`, `npt`: opérateurs booléens
- `==`: compare deux valeurs quelconques

3.3 Comparaison

- `<`, `≤`, `>`, `≥`: uniquement entre éléments du même type, pour les valeurs numériques et les chaînes
- `==`: teste l'égalité entre deux valeurs
- `!=`: teste la différence entre deux valeurs

4 Autres

4.1 Fonctions

- `def fonction(var1: type1, var2: type2, ... ) -> type_r`: déclare une fonction qui prends en entrée des variables de type donné, et renvoie une valeur de type `type_r`
- `return value`: arrête la fonction, et renvoie la valeur
- `assert expr`: vérifie que `expr` évalue à Vrai. Si ce n'est pas le cas, lève une erreur

4.2 Interaction

- `print(value)`: Affiche la valeur
- `var = input("entrez une valeur")` : assigne la valeur entrée par l'utilsateur (forcément une chaîne)

4.3 Misc

- `# Commentaire`

III Exemples

1 Max

```
def max(val1: int, val2: int) -> int
  """Retourne la plus grande valeur"""
  if val1 >= val2:
    return val1
  return val2
```

2 Division euclidienne

Algorithme qui calcule le quotient de deux nombres entier.

```
def div_eucl(dividende: int, diviseur: int) -> int:
  if diviseur == 0:
    raise Exception("Impossible de diviser par 0")
  if diviseur <= 0 or dividende <= 0:
    raise Exception("Le dividende et le diviseur doivent être positifs")

  quotient: int = 0
  while dividende >= diviseur:
    dividende = dividende - diviseur
    quotient = quotient + 1

  return quotient
```

IV Exercices

1 Niveau 1

1.1 Min 1

```python
def min3(a: int, b: int, c: int) -> int:
  """Sortie: le plus petit des 3 nombres."""
```

1.2 Min 2

```python
def min9(a: int, b: int, c: int, d: int, e: int, f: int, g: int, h: int, i: int) ->
int:
  """Sortie: le plus petit des 9 nombres."""
```

1.3 Conversion

```python
def duree_en_secondes(j: int, h: int, m: int, s: int) -> int:

```

1.4 Algorithme qui...

Trouve un nombre entier positif tel que:

- son carré est plus grand que 6000
- son cube divisé par 100 est plus grand que 10000

2 Niveau 2

2.1 Dates

2.1.1 Années bisextiles

Écrire une fonction qui prends en entrée un nombre entier, et:

- vérifie que le nombre est une année valide (sinon lève une erreur)
- renvoie un booléen qui indique si l'année donnée est bisextile

> Les années sont en général bissextiles si elles sont multiples de quatre, mais pas si elles sont multiples de cent, à l'exception des années multiples de quatre cents qui, elles, sont également bissextiles.

2.1.2 Nombre de jours par an

```python
def jours_par_annee(année: int) -> int:
    """
    Pré-condition: année > 0
    Sortie: nombre de jour de l'année
    """
```

2.1.3 Nombre de jours par mois

```
def jours_par_mois(année: int, mois: int) -> int:
    """
    Pré-condition: année > 0 , 1 ≤ mois ≤ 12
    Sortie: le nombre de jour pour le mois et l'année donnée
    """
```

2.1.4 Comparer dates

```
def compare_dates(année1: int, mois1: int, jour1: int, année2: int, mois2: int,
jour2: int) -> int:
    """
    Pré-condition: année > 0 , 1 ≤ mois ≤ 12, 1 ≤ j1 < jours_par_mois(année, mois)
    Sortie:
     - 0 si les dates sont égales
     - 1 si la 1ère date est plus grande
     - -1 sinon
    """
```

2.2 Entrée utilisateur

Écrire une fonction qui demande son nom à l'utilisateur, puis lui demande de confirmer. Recommence tant que l'utilsateur n'a pas confirmé.

2.3 Algorithme qui...

Trouve un nombre entier positif tel que:

- son carré est plus grand que 6000
- son cube divisé par 100 est plus petit que 4000

2.4 PGCD

> L'algorithme d'Euclide sur deux nombres entiers positifs a et b avec a > b ⩾ 0 procède comme suit :
>
> - si b = 0, l'algorithme termine et rend la valeur a ;
> - sinon, l'algorithme calcule le reste r de la division euclidienne de a par b, puis recommence avec a := b et b := r.

2.5 Affichage

2.5.1 Encadrement

```
def encadre_1(titre: str) -> str:
    """
    Sortie: le titre, encadré par le caractère "*"
    Exemple:
    *******
    *titre*
    *******
    """
```

2.5.2 Diamant

```
def diamant(n: int) -> str:
    """
    Sortie: Un "diamant" en chaîne de caractères
    Exemple pour n = 3:
      *
     ***
    *****
     ***
      *
    """
```

3 Niveau 3

3.1 Impôts par tranche

Écrire une fonction qui calcule le montant des impôts, sachant que

Calcul du montant des impôts (depuis https://www.economie.gouv.fr/particuliers/tranches-imposition-impot-revenu):

Fraction du revenu	Taux d'imposition sur la tranche
Jusqu'à 11 294 €	0 %
De 11 295 € à 28 797 €	11 %
De 28 798 € à 82 341 €	30 %
De 82 342 € à 177 106 €	41 %
Supérieur à 177 106 €	45 %