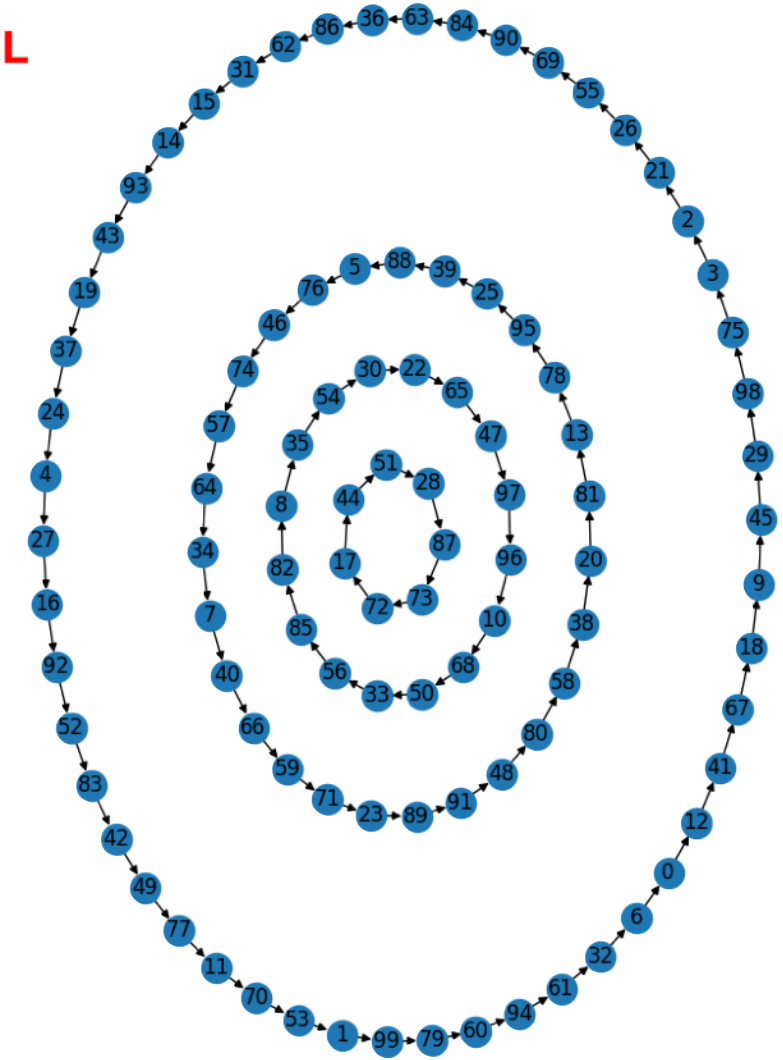

Information, Calcul et Communication

Partie Programmation

Cours 6: Structures de données (fin)

Portée des variables

27.10.2023

Patrick Wang

Au sommaire cette semaine

1. Tuples – Muabilité, immuabilité
2. Ensembles
3. Dictionnaires
4. Portées des variables

Au sommaire cette semaine

1. Tuples – Muabilité, immuabilité
2. Ensembles
3. Dictionnaires
4. Portées des variables

1. Tuples – Muabilité, immuabilité

Retour sur l'exercice 1 de la série 5

```python
def min_max(l: List[int]) -> int:
    low: int = l[0]
    high: int = l[0]
    for element in l:
        if element < low:
            low = element
        if element > high:
            high = element
    return low, high
```

Un message d'alerte a peut-être été affiché quant au type de retour

Cette fonction retourne plusieurs entiers

1. Tuples – Muabilité, immuabilité

Déclaration et manipulation d'un tuple

- Un tuple est assimilable à une liste immuable.
  - Cela signifie qu'un tuple ne peut pas être modifié une fois créé.
  - Pas de modification d'élément, pas d'ajout ou de retrait d'élément, ...


```python
my_tuple: Tuple[int] = (12, 53, 0)
# Récupération des éléments par indice
for i in range(len(my_tuple)):
    print(my_tuple[i])
```

1. Tuples – Muabilité, immuabilité

Utilisation avec le `return` d'une fonction

```python
from typing import List, Tuple

def min_max(l: List[int]) -> Tuple[int]:
    low: int = l[0]
    high: int = l[0]
    for element in l:
        if element < low:
            low = element
        if element > high:
            high = element
    return low, high

l = [3, 54, 19, 0, -2, 13]
t = min_max(l)
low = t[0]
high = t[1]
```

Ajout pour l'annotation

Récupération des valeurs retournées dans un tuple

1. Tuples – Muabilité, immuabilité

Que se passe-t-il si je souhaite modifier un élément ?

```
28  my_tuple: Tuple[int] = (12, 53, 0)
29  my_tuple[0] = 13
```

```
PROBLEMS 30   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

❯ PYTHONIOENCODING='utf-8' venv/bin/python3 "/Users/p54244/dev/ICC/c06_tuple.py"
Traceback (most recent call last):
  File "/Users/p54244/dev/ICC/c06_tuple.py", line 19, in <module>
    my_tuple[0] = 13
    ~~~~~~~~^^^
TypeError: 'tuple' object does not support item assignment
```

Au sommaire cette semaine

1. Tuples – Muabilité, immuabilité
2. Ensembles
3. Dictionnaires
4. Portées des variables

2. Ensembles

Définition des ensembles

- Un ensemble (ou set) est une structure de données qui :
  - garantit l'unicité des éléments
  - ne possède pas d'ordre (impossible d'utiliser les indices)
- Comme les listes, les ensembles sont des objets muables (possibilité d'ajouter et de retirer des éléments d'un ensemble)

2. Ensembles

Déclaration d'ensembles

- Déclaration d'un ensemble vide: `set()`
- Déclaration d'un ensemble avec des éléments : `{e1, e2, e3}`

```
from typing import Set

empty_set: Set[int] = set()
nonempty_set: Set[int] = {1, 2, 3, 6, 6, 8}
```

2. Ensembles

Manipulations d'ensembles (1/2)

- Exercice 2 de la série 4 : suppression de doublons dans une liste

```python
numbers: List[int] = [1, 1, 1, 3, 5, 5]
numbers = list(set(numbers))
print(numbers)
```

2. Ensembles

Manipulations d'ensembles (1/2)

- Exercice 2 de la série 4 : suppression de doublons dans une liste

```
numbers: List[int] = [1, 1, 1, 3, 5, 5]
numbers = list(set(numbers))
print(numbers)
```

2. Ensembles

Manipulations d'ensembles (1/2)

- Exercice 2 de la série 4 : suppression de doublons dans une liste

```
numbers: List[int] = [1, 1, 1, 3, 5, 5]
numbers = list({1, 3, 5})
print(numbers)
```

2. Ensembles

Manipulations d'ensembles (1/2)

- Exercice 2 de la série 4 : suppression de doublons dans une liste

```
numbers: List[int] = [1, 1, 1, 3, 5, 5]
numbers = list({1, 3, 5})
print(numbers)
```

2. Ensembles

Manipulations d'ensembles (1/2)

- Exercice 2 de la série 4 : suppression de doublons dans une liste

```
numbers: List[int] = [1, 1, 1, 3, 5, 5]
numbers = [1, 3, 5]
print(numbers)
```

2. Ensembles

Manipulations d'ensembles (1/2)

- Exercice 2 de la série 4 : suppression de doublons dans une liste

```
numbers: List[int] = [1, 1, 1, 3, 5, 5]
numbers = list(set(numbers))
print(numbers) # Affiche [1, 3, 5]
```

- Parcours des éléments d'un ensemble (impossible de parcourir avec des indices !)

```
nonempty_set: Set[int] = {1, 2, 3, 6, 6, 8}
for element in nonempty_set:
    print(element)
```

2. Ensembles

Manipulations d'ensembles (2/2)

- Quelques méthodes utiles :

```
nonempty_set: Set[int] = {1, 2, 3, 6, 6, 8}
empty_set: Set[int] = set()

empty_set.add(1)    # Ajoute un élément, sauf s'il est déjà présent
empty_set.remove(1) # Retire l'élément, ou produit une erreur s'il n'est pas présent
empty_set.clear()

u: Set[int] = empty_set.union(nonempty_set)
i: Set[int] = empty_set.intersection(nonempty_set)
```

Au sommaire cette semaine

1. Tuples – Muabilité, immuabilité
2. Ensembles
3. Dictionnaires
4. Portées des variables

3. Dictionnaires

Définition des dictionnaires

- Un dictionnaire est une structure de données qui fonctionne avec le principle de (clé, valeur)
  - La clé est nécessairement d'un type immuable (int, str, Tuple, ...)
  - La valeur peut être de n'importe quel type
- Dans un dictionnaire, une clé ne peut apparaître qu'une fois

3. Dictionnaires

Déclaration d'un dictionnaire

- Déclaration d'un dictionnaire vide: `{ }`
- Déclaration d'un dictionnaire avec éléments : `{key1: value1, key2: value2}`

```
from typing import Dict

empty_dict: Dict[int, str] = {}
romandie: Dict[str, str] = {"BE": "Berne", "FR": "Fribourg", "GE": "Genève",
                            "JU": "Jura", "NE": "Neuchâtel", "VD": "Vaud",
                            "VS": "Valais"}
```

3. Dictionnaires

Manipulations d'un dictionnaire (1/2)

- Récupérer la valeur de la clé key : `vaud: str = romandie["VD"]`
- Modifier la valeur de la clé key : `romandie["GE"] = "Genf"`
- Récupérer toutes les clés : `keys = romandie.keys()`
- Récupérer toutes les valeurs : `values = romandie.values()`
- Récupérer les paires `(key, value)` : `key_value = romandie.items()`

3. Dictionnaires

Manipulations d'un dictionnaire (2/2)

- Parcourir les éléments d'un dictionnaire

```
for key in romandie.keys():
    print(f"{key}: {romandie[key]}")

for key, value in romandie.items():
    print(f"{key}: {value}")

for value in romandie.values():
    print(f"{value}") # Mais on perd l'information des clés
```

Démo !

Au sommaire cette semaine

1. Tuples – Muabilité, immuabilité
2. Ensembles
3. Dictionnaires
4. Portées des variables

4. Portées des variables

Exemple sur ED discussion

Variable déclarée en dehors de toute fonction

```
my_var = 0

def f():
    my_var = 2
    print(my_var)

print(my_var)
f()
print(my_var)

# Affiche
# 0
# 2
# 0
```

Variable déclarée à l'intérieur d'une fonction. Sa portée est limitée à cette déclaration de fonction

Conclusion – Informations sur le midterm

- Les concepts :
  - Variables
  - Structures de contrôle (`if`, `for`, `while`)
  - Les fonctions
  - Les structures de données (`List`, `Tuple`, `Set`, `Dict`)
- Leurs utilisations :
  - Créer des branchements conditionnels
  - Répéter des instructions
  - Parcourir (indice ou élément ?) des chaînes de caractères, listes, ensembles, dictionnaires
  - Décomposer un problème en sous-problèmes plus facile à résoudre