SPUS201 - UE SCIENCES : Introduction à la programmation 2

Denis Dubruel - Cours Magistral N°3

Année 2024/2025

courriel : prenom.nom@univ-cotedazur.fr

Table des Matières.

Table des matières

Méthodologie

1. **Démarrer sur un échantillon réduit :**
   - Travailler sur un petit ensemble de données traitable manuellement.
   - Utiliser du papier
2. **Identifier les données :**
   - Définir données d'entrée et de sortie.
   - Préciser leur domaine, contraintes et relations.
3. **Résolution manuelle :**
   - calculer étape par étape
   - noter les opérations répétées
   - découper les étapes complexes en sous-problèmes simples.

Voir Moodle

Méthodologie

4. **Formalisation -Identification des structures :**
    - Boucles (sur quoi ? condition d'arrêt ?)
    - Tests (quelle condition ?)
    - Données en entrée.
    - Calculs à mener.
    - Stockage des résultats.
    - Résultat final.
5. **"Factorisation" en fonctions** :
    - Détecter les "répétitions" pour créer des fonctions ou des classes.

Méthodologie

6. **Passage au code** (traduction des observations) :
    - Variables → issues des noms manipulés.
    - Tests → if condition :
    - Boucles sur séquence → for variable in séquence :
    - Boucles conditionnelles → while condition :
    - Séquences répétées → fonctions (def fonction():).
    - Retour de résultat → return variable.
    - Vérification des erreurs → Tests et exceptions

Méthodologie

ENONCE : Gestion des notes d'étudiants

Un enseignant souhaite automatiser le calcul des moyennes de ses étudiants. Il dispose d'un fichier contenant les noms des étudiants ainsi que leurs notes dans son UE. L'objectif est de développer un algorithme qui :

- Lit les données (noms des étudiants et notes) depuis un fichier ou une entrée utilisateur.
- Vérifie la validité des données (notes comprises entre 0 et 20, nombres valides, etc.).
- Calcule la moyenne de chaque étudiant.
- Détermine la mention associée :
- Affiche les résultats et les sauvegarde dans un fichier.

Méthodologie

1. Travailler sur un petit échantillon : commencer avec 3 étudiants et quelques notes.
2. Identifier les entrées et sorties : noms et notes en entrée, moyenne et mention en sortie.
3. Résolution à la main : effectuer les calculs manuellement sur un échantillon avant d'implémenter.

Méthodologie

4 Formalisation :
   - Boucles pour parcourir les étudiants.
   - Tests pour vérifier la validité des notes et assigner la mention.
   - Stockage des résultats en mémoire et dans un fichier.

5 Factorisation(ou découpage) en fonctions :
   - charger_les_données(fichier)
   - calcul_moyenne(notes)
   - attribuer_mention(moyenne)
   - sauvegarder_resultats(fichier, resultats)

Table des matières

Ensemble - Définition

Un ensemble est une collection d'objets uniques hétérogènes. Cet ensemble est non ordonné et non indexable.

Les éléments de l'ensemble sont de type immutable (entier, flottant, chaine) donc pas de type list par exemple.

SHELL

```
>>> E={ 14, 34 , 'toto'}
>>> vide=set()   # défini un ensemble vide.  {} est un dictionnaire
>>> 14 in E     # pour vérifier l'appartenance
True
>>> E[1]   # non indexable !
'TypeError: 'set' object is not subscriptable   '
```

..

Ensemble - Ajout & Suppresion d'éléments

Ajout, suppression des éléments :

SHELL

```
>>> Ensemble={0,3,6,9,12,15,18}
# Ajout d'un élément
>>> Ensemble.add(21)
# suppression d'un élément
>>>Ensemble.delete(0)
# Que reste t'il après les modifications :
>>>Ensemble
{18, 3, 21, 6, 9, 12, 15}  # les éléments ne sont pas dans le même ordre.
```

Ensemble - Exemple

```
def divisibleInf100(n : int) -> set[int] :
    E : set = set()
    for i in range (100//n+1):
        E.add(n*i)
    return E
# Les entiers divisibles par 15 et inférieurs à 100.
print(divisibleInf100(15))
```

SHELL

```
>>> %Run ensemble.py
{0, 75, 45, 15, 90, 60, 30}   # l'ordre est quelconque
```

..

Ensemble - Manipulation

Appartenance d'un élément à un ensemble : **in**

SHELL

```
>>> A = {10,30} # Ensemble contenant  10 et 30
>>> 20 in A    # booleéen : se lit 20 appartient à A
False
```

Ensemble - Manipulation

Un ensemble A est inclus dans B si tous les éléments de A sont dans B. ($A \subseteq B$ en math ou A <= B en python)

Un ensemble A est **strictement** inclus dans B si tous les éléments de A sont dans B **ET** $A \neq B$.

($A \subset B$ en math ou A < B en python)

script

```
A = { 3 ,4 }
B = { 3 , 4 , 5 }
print(f"A= {A} et B={B}")
print(f"A <= B est {A <= B}" )
print(f"A<B est {A<B}" )
C = { 3 ,4 ,5 }
print(f"C = {C}")
print(f"B<=C est {B <= C}" )
print(f"B<C  est {B <  C}" )
```

SHELL

```
A= {3, 4} et B={3, 4, 5}
A <= B est True
A <  B est True

C = {3, 4, 5}
B<=C   est True
B<C    est False # car B=C
```

Ensemble - Union , Intersection, Différence.

	notation mathématiques	python
Union	$A \cup B = \{x : x \in A \text{ ou } x \in B\}$	A | B
Intersection	$A \cap B = \{x : x \in A \text{ et } x \in B\}$	A & B
Différence	$A \setminus B = \{x : x \in A \text{ et } x \notin B\}$	A - B
Cardinal	$card(A)$	len(A)

SHELL

```
>>> print({11,22,33} | {44,55}) # Union
{33, 22, 55, 11, 44}    # les éléments sont dans un ordre quelconque
>>>  print({11,22,33} & {44,55}) # Intersection
set()    # pas d'éléments communs donc ensemble vide
>>>  print({11,22,33} & {33,44,55})
{33}
>>>  print({11,22,33} - {33,44,55})    # Différence
 {11, 22}
>>>print(len({11,22,33}))
3
```

Ensemble

Parcours et lecture d'un ensemble :

SCRIPT

```
def parcours(E):
  for element in E:
     print(f"{element} appartient à {E}")
  print()
  print(f"len({E}) renvoie {len(E)}.")

G : set = {'a' , 2 , True}
parcours(G)
```

SHELL

```
>>>
True appartient à {True, 'a', 2}
a appartient à {True, 'a', 2}
2 appartient à {True, 'a', 2}

len({True, 'a', 2}) renvoie 3.
```

SCRIPT

```
vide :set = set()
parcours(vide)
```

Pas d'affichage du contenu de l'ensemble vide. Logique !

SHELL

```
>>>
len(set()) renvoie 0
>>>
>>>print(vide)
set() # et pas {}
   #(pour dictionnaire vide !!!)
```

Table des matières

Dictionnaires

Un **dictionnaire** est une structure de données qui stocke des paires clé-valeur uniques.

Les **clés** sont immuables (comme des chaînes, nombres ou tuples)

Les **valeurs** peuvent être de tout type.

dictionnaire = { clé1 : valeur1,...,cléN : valeurN}

SCRIPT

```
dictionnaireStock = {'patates' : 34 , 'carottes':12}

dictionnaireNotes = {'Arthur' :  15 , 'Elisa' :18}

dictGroupeAge ={ 14 : ['Leo','Lea','Zoe']  ,15 : ['Tom','Zoe']}
```

Dictionnaires

Accès aux valeurs à l'aide des clés, **modification**, **ajout**, **suppression** des éléments :

SHELL

```
# Accès aux valeurs à l'aide des clés :
>>> dictionnaireStock['patates']
 34
>>> dictGroupeAge[15]
['Tom','Zoe']
# Modification d'une valeur
>>> dictionnaireStock['patates']=40
# Ajout d'un élément (clé : valeur)
>>> dictionnaireStock['salades']=25
# Suppresion d'un élément
>>>dictionnaireStock.pop("patates")
# Que reste t'il après les modifications :
>>>dictionnaireStock
{'carottes': 12, 'salades': 25}
```

Dictionnaires

Les clés sont non mutables (non modifiables). Pour les clés, il faut des types entiers, flottants, chaîne, tuple (d'éléments non mutables bien sûr !!)

SHELL

```
# dictionnaires de points du plan. Les clés sont des tuples de flottant.
>>>PointsDuPlan = { (0 , 0) :' O' , (1 ,4 ) : 'A' , (4.3 , 5) : 'B' }
#
>>> dicoPb = { [4, 5] : 'liste de prix'} # [4,5] est mutable donc non hachable
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'unhashable' type: 'list'
```

Dictionnaires

Parcours et lecture d'un dictionnaire :

SCRIPT

```
def parcours(dico):
    for cle in dico:
        valeur = dico[cle] #
        print(f"{cle} ({v})")
```

SHELL

```
>>> parcours(dictionnaireStock)
carottes (12)
salades (25)
```

SCRIPT

```
E : dict ={ }   # dictionnaire vide

print(f"len(E) renvoie {len(E)}")
print(f"Avec  E ={E}.")
```

SHELL

```
>>> %Run -c $EDITOR_CONTENT
len(E) renvoie 0
Avec  E ={}.
```

Exercice : L'EX2 du TP 2 peut se résoudre à l'aide d'un dictionnaire dont la clé est u_0 et sa valeur, le temps de vol. contenu...

Table des matières

Mais quel type ??

SCRIPT

```
pi : float = 3.14
chaine : str ="Valrose"
dico : dict = { "yes":"oui" , "no":"non"}
#
j : int = int (pi)
print(f"j ={j}")
print(f"type(j) : {type(j)}" )

chaine2 : str =str(pi)
print(f"chaine2 ={chaine2}")
print(f"chaine2 : {type(chaine2)}" )
#Ensemble
F : set = set(dico)
print(f"F ={F}")
print(f"type(F) : {type(F)}" )

G : set = set(chaine)
print(f"G={G}")
print(f"type(G) : {type(G)}" )
```

SHELL

```
>>> %Run conversion_Type.py
j =3 # arrondi de 3.14
type(j) : <class 'int'>


chaine2 =3.14
chaine2 : <class 'str'>


F ={'yes', 'no'} # Que les clés !
type(F) : <class 'set'>


G={'V', 'l', 'e', 'o', 'a', 'r', 's'}
type(G) : <class 'set'>
```

Typage & conversion ??

En programmation bien connaître et maîtriser le type de données. ⚠ source d'erreur très mais très fréquente !! ⚠

Conversion de type "simple et facile" avec une instruction pour : passer du type **int** vers **float** vers **str** vers **set**

Attention, en fonction du besoin, il est nécessaire d'écrire une fonction de conversion. Mais surtout , il faut **tester et valider** la conversion !! Rappel : pour les fonctions il est plus clair et explicite de bien écrire les types dans les signatures :

script

```
def multiplie( x : int , y : int ) -> int : # explicite

def calcule (a,b,c) :  # implicite , à voir dans le contexte.
```

Table des matières

Manipulation Fichiers *.txt - Lecture

Lecture de fichier *.txt (vu en CM1)

code.py

```
from io import  TextIOWrapper
lignes : list [str] =[]
fichier : TextIOWrapper = open("fichier_etudiant.txt","r", encoding="utf-8")# ouver
lignes = fichier.readlines( ) #chargement de toutes les lignes
fichier.close( )    #fermeture.
print(lignes)       # affiche le contenu du fichier
```

SHELL

```
>>>
['Arthur   18\n', 'Adrien   20\n', 'Amélie   19\n', 'Léa      20\n']
```

Manipulation Fichiers *.txt - Ecriture

Ecriture de fichier *.txt (vu en CM1)

code.py

```
Etudiants = ["Nicolas", "Eudes", "Clotaire"] # data
fichier : TextIOWrapper =open("LesPrenoms2.txt", "w", encoding="utf-8") # ouverture
i : int=0
while i< len(Etudiants):
    fichier.write(f'{Etudiants[i]}\n') # écriture avec ajout du \n
    i=i+1
fichier.close()    # fermeture
```

SHELL

```
>>> affiche_contenu("LesPrenoms.txt")
Nicolas
Eudes
Clotaire
```

Manipulation Fichiers *.csv

Fichier CSV (Comma-Separated Values) contient des données structurées (tableau, feuille de calcul etc).

```
Nom , Age , Profession
Alice , 30 , Ingénieure
Bob , 25 , Influenceur
Charlie , 35 , Informaticien

"Nom" , "Age" , "Profession"
"Alice" , 30 , "Ingénieure"
"Bob" , 25 , "Influenceur"
"Charlie" , 35 , "Informaticien"
```

Nom	Age	Métier
Alice	30	Ingénieure
Bob	25	Influenceur
Charlie	32	Informaticien

Manipulation Fichiers *.csv

Lecture d'un fichier *.csv

Script

```
from io import TextIOWrapper
import csv
# Liste pour stocker les lignes
lignes: list[str] = []
# ouverture
fichier : TextIOWrapper = open("ex.csv", "r", encoding="utf-8")

reader = csv.reader(fichier) # lecture
for ligne in reader:
    lignes.append(ligne)  # Ajouter chaque ligne à la liste
fichier.close()    #fermeture.
print(lignes)
```

SHELL

```
>>> %Run charge_csv.py
[['Nom', 'Prenom', 'Age'], ['Durand', 'Alex', '25'],
 ['Dupond', 'Théo', '32'], ['Dupont ', 'Louis', '45']]
```

Manipulation Fichiers *.csv

Ecriture d'un fichier *.csv

Script

```
from io import TextIOWrapper
import csv
# Données à écrire
donnees : list = [
    ["Nom", "Âge", "Classe"],  # En-têtes
    ["Alice", 20, "Maths"],
    ["Bob", 22, "Physique"],
    ["Charlie", 21, "Informatique"] ]
# Ouvrir le fichier en mode écriture
fichier : TextIOWrapper = open("data.csv", "w", encoding="utf-8", newline="")
# Créer un writer CSV
writer = csv.writer(fichier)
# Écrire les lignes
for ligne in donnees:
    writer.writerow(ligne)
# Fermer le fichier
fichier.close()
```

Manipulation Fichiers json

JSON (**J**ava**S**cript **O**bject **N**otation) : format d'échange de données. A l'origine pour le langage Java mais le contenu structuré textuel est utilisable dans d'autres langages. Types de données contenues :

- une chaîne de caractères
- Un nombre (entier ou décimal)
- Un objet null (avec Python, l'objet None)
- Un booléen
- Une liste (dans d'autres langages, appelé tablea
- Un dictionnaire

Manipulation Fichiers json

```
["Pascal", "Patrick"]
```

```
"Patrick"
```

```
2938.231
```

```
{
  "Patrick": {
      "salaire": 24000,
      "date_embauche": "2020-10-25",
      "chef": true
  },
  "Pascal": {
      "salaire": 35000,
      "date_embauche": "2020-08-17",
      "chef": false
  }
}
```

```
{
    "Cahiers": 3,
    "Stylos": 2,
    "Agrafes": null
}
```

```
["Bonjour", null, "test"]
```

Exemples de contenu json.

- ⚠un seul objet par fichier !
- Faire une liste ou dictionnaire si besoin.
- null pour None en Python.
- true&false en minuscules.

Manipulation Fichiers json

Lecture d'un fichier json :

Le module json se charge automatiquement de convertir les données JSON dans un format de données valides pour Python.

```
from io import TextIOWrapper
import json
nom :str = "f4.json"
fichier : TextIOWrapper = open(nom, "r", encoding="utf-8")
data = json.load(fichier)
fichier.close()
```

Manipulation Fichiers json

Suite du code pour visualiser le chargement.

Script

```
print(f"Le fichier "{nom}" contient ")
print(f"une variable de type {type(data)}.")
print(f"Son contenu est : ")
print(data)
```

SHELL

```
>>> %Run chargeUnSeul.py
Le fichier f4.json contient
une variable de type <class 'dict'>.
Son contenu est :
{'Patrick': {'salaire': 2400, 'date_embauche': '2020-10-25', 'chef': True},
 'Pascal': {'salaire': 3500, 'date_embauche': '2020-08-17', 'chef': False}}
```

Manipulation Fichiers json

Ecriture du fichier json :

ecritFichierJson.py

```
from io import TextIOWrapper
import json

data : list = [1, 2, 3, 4, 5]

fichier : TextIOWrapper =open("le_fichier.json", "w" , encoding="utf-8" )
json.dump(data, fichier)
fichier.close( )
```

Manipulation de fichiers, Synthèse.

Les types ne sont pas précisés, bien noter les 3 étapes communes.

```
fichier = open("fichier_etudiant.txt","r", encoding="utf-8")  # ouverture
lignes = fichier.readlines()                                  # lecture
fichier.close()                                               #fermeture
```

```
import csv
fichier = open("ex.csv", "r", encoding="utf-8") # ouverture
contenu = csv.reader(fichier)  # lecture
for ligne in contenu:
    lignes.append(ligne)      # traitement
fichier.close()    #fermeture
```

```
import json
fichier = open("f4.json", "r", encoding="utf-8")   # ouverture
data = json.load(fichier)                          # lecture
fichier.close()                                    # fermeture
```

Manipulation de fichiers, Synthèse.

comparaison pour l'ECRITURE.

```
Etudiants = ["Nicolas", "Eudes", "Clotaire"] # data
fichier  =open("LesPrenoms2.txt", "w", encoding="utf-8") # ouverture
for etudiant in Etudiants :
     fichier.write(f"{etudiant}\n") # écriture avec ajout du \n
fichier.close( )                    # fermeture
```

```
import csv
donnees = ["Nom", "Âge", "Classe"]   # data réduite
fichier = open("data.csv", "w", encoding="utf-8", newline="") # Ouverture
writer = csv.writer(fichier)
for ligne in donnees:           # Écriture
    writer.writerow(ligne)
fichier.close( )                # Fermeture
```

```
import json
data : list = [1, 2, 3, 4, 5]
fichier = open("le_fichier.json", "w" , encoding="utf-8" ) # ouverture
json.dump(data, fichier)                                   # ecriture
fichier.close( )                                           # fermeture
```

Manipulation de fichiers, Synthèse.

Le choix du format de fichier dépend de vos applications. Par exemple en première approche :

- Que du texte ?
  -> *.txt
- Des données issues d'un tableau "bureautique" ?
  -> *.csv
- Des données structurées issues d'un autre programme ?
  -> *.json

A pratiquer en TP.

Il existe d'autres formats : XML, YAML ... hors cadre de cette UE.

Licence

Ce document est publié sous licence Creative Commons :

- ©2024 — Denis Dubruel - Université Côte d'Azur
- Attribution
- Utilisation non commerciale
- Partage dans les mêmes conditions 4.0 International

https://creativecommons.org/licenses/by-nc-sa/4.0/legalcode.fr