EPFL

ICC

Programmation

GC/MX, **Cours 6**

28 octobre 2022

Jean-Philippe Pellet

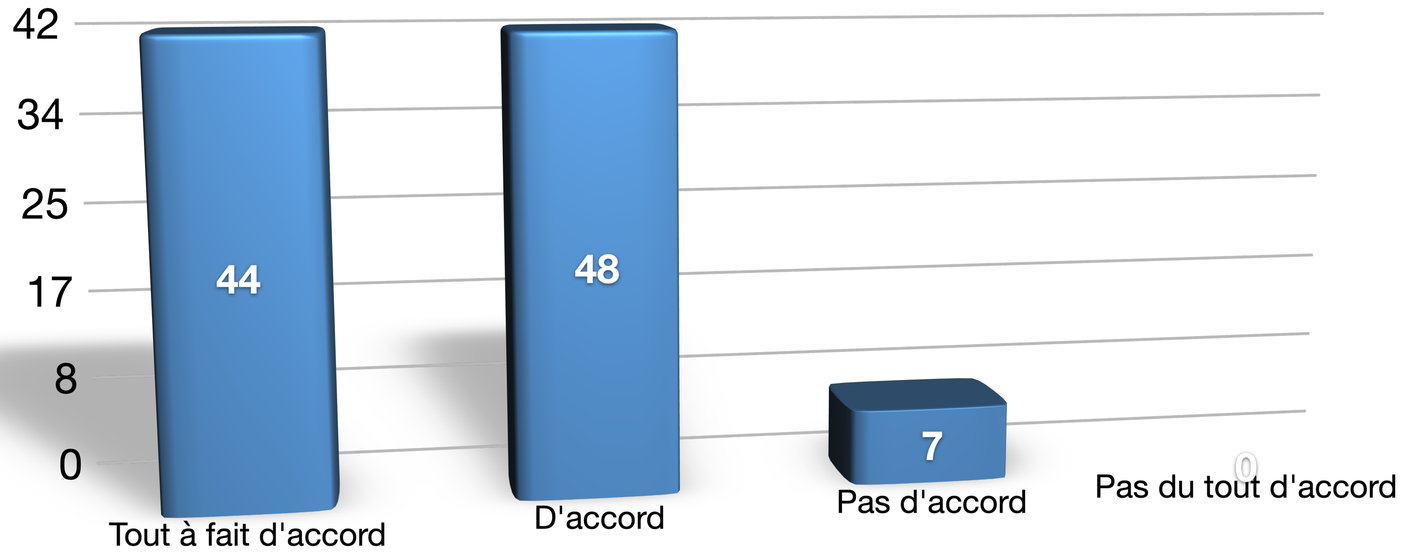
Opening music: The Hillbilly Moon Explosion, album *Raw Deal* (2010)

Previously, on Programmation...

- Types de base en Python: int, float, str, bool
- Méthodes, fonctions et slicing pour calculer des valeurs dérivées
- Conditions pour exécuter du code selon la valeur d'une expression booléenne:
  `if <condition>: ... else: ...` et ses variantes
- Boucles pour exécuter du code plusieurs fois:
  - Boucle `while <condition>: ...`
  - Boucle `for i in range(...): ...`
- Déclaration de fonctions avec type de retour et paramètres:
  - `def calculate_area(r: float) -> float: return ...`
- Utilisation de listes:
  - `values: List[int] = [1, 4, 2, 7, 3]`
  - Les listes sont des objets modifiables

Évaluation du cours

«Le déroulement du cours permet ma formation et un climat de classe approprié.»



Pourcentages cumulés: **44%** **91%** **98%**

Participation: 101/154, dont 42 avec remarque

Merci pour votre feedback!

C'est un plaisir pour Olivier et moi de donner ce cours dans de telles conditions!

Feedback complet

- (Python) Les corrigés vidéos des exercices sont très utiles mais les exercices sont difficiles.
- C'est bien, qu'on commence par le début et les éxpliquations sont aussi bien, mais les exercices de programmation sont assez difficile, si on a jamais vu Python.
- Ce serait bien d'avoir plus d'exercices pour pouvoir s'exercer
- cours bien structuré. une bonne quantité d'exemples avec des professeurs dynamiques et prêt à répondre aux questions.
- Cours très bien fait, rien à redire si ce n'est le contrainte du formulaire à l'examen. J'aurais préféré avoir la possibilité de prendre plus de documents avec moi comme l'année passée.
- Cours très bien structuré, peut-être juste avec l'approche de l'examen de midterm, qui se trouve être notre premier de Bachelor et par conséquent nous rempli d'appréhension, il est dans moins de 2 semaines et nous avons aucunes idées de ce à quoi il va ressembler donc peut-être fournir des examens blancs un peu plus tôt si possible, merci
- Difficile cependant les profs prennent leur temps pour expliquer
- Dommage qu'il n'y ait qu'une seule heure d'exercices d'ICC théorique. Sinon super
- good overall course lack of marked and counted evaluation doesn't help the student address the issues they don't understand and maybe not even know where they lack more work. A one off exam that decides on your future at EPFL is heavily outdated and not inclusive of the capacities of everyone (from material conditions such as those who need a part time job to survive to more personal issues such as mental health and so on). there is lot's of room to improve the examination and support for the students, module leaders from different courses should work together to be relevant to each other when possible throughout the course. students should be provided with not only academic support but also personal and life support from a personal tutor. who would be aware of each student's issues and so on. a good example of that is the higher education system in the UK which is centered around the well-beings of students and not overly obsessed with last millennia's criteria for "success". overall there is a lack of practice and lab work and assignments for first year students which keeps the momentum for the students boring. this is shown throughout the success rate at BA1 with only a 45% success.i would qualify this aspect of the entire course as highly elitist especially for a course that attracts international students there is a huge difference in starting points for different people. hence why a gradual increase of expectations of student is necessary. most of the people here just got out of high school giving anyone high amount of pressure and responsibilities (both academic and personal mind you lots of people are leaving the comfort of living in a family and have to cook clean and so on...) just leads to them overworking themselves or being scared and anxious although under other conditions they could actually out perform themselves shock therapy is then not a good way to deal with this. a good engineer is not an engineer who fills the criteria of examination but someone creative and who likes what they are doing and aware of the ethical and social responsibilities that his jobs upholds, which brings me to my next point lack of social science aspect in the course overall. it's understandable that this is a science course but engineers shape our day to day lives it's important for them to have ethical and social science awareness throughout the course. it should build people not not mindless alienated workers(capitalism cough cough). in my opinion this is a great uni with good public and private investments we should push it forward with more progressive ideas of how we should address 21st century higher education in order to push forward. for society but also for the students their selves to create the engineers of tomorrow. good thing is there is a lot of innovative programs for comparable universities to learn from but I'm sure there are a lot of innovative people and teams that can shape the future of higher education. PS: have you ever considered holding a study about the mental health of the students at EPFL and it relationship with their academic life ? if so please feel free to share it on: *<redacted>*
- J'espère que cette année il y aurait plus d'exos pour s'entraîner pour les examens. J'aimerais aussi avoir un peu plus d'info par rapport au projet d'informatique qui, pour l'instant, ont été très confuses
- j’aime beaucoup
- Je n’ai aucune remarque particulière à faire. Les cours, la partie théorique comme la programmation, sont bien structurées et claires.
- Je trouve ce cours un abstrait et je pense que l'ajout de plus de schemas descriptifs détaillés des idees présentés pourraient le rendre beaucoup plus comprehensible.
- Je trouve qu'il n'y a pas assez d'heure de cours pour la programmation, mais cela n'a pas vraiment a voir avec le cours c'est plutôt une question de planning. J'aurais préféré comme pour la théorie, 2heures de cours et 1 d'exercises.
- La partie programmation bénéficie d'un prof passionné (et passionnant) dont les nombreuses démonstrations illustre parfaitement le cours. L'algorithme ne pourrait pas être mieux enseigné.
- La partie programmation est assez bien enseigné et on comprend bien ce qui se passe. La partie théorie s’occupe avec des problèmes trop longtemps et ne passe pas assez de temps de montrer les concepts et leur definitions en detail et comment les appliquer.
- La partie théorique est très bien et compréhensible et avec un bon rythme. La partie programmation est moins clair et les concepts ne sont pas très bien expliqués; personnellement j'utilise une application à côté (Mimo) pour apprendre Python en complément, et j'ai mieux compris certains concepts fondamentaux grâce à cette application que pendant le cours. Le problème en programmation est que j’ai l’impression qu’on m’a donné une montagne d’informations sans grande structure et cela était très dur à digérer, j’ai eu beaucoup de mal à comprendre les concepts de base et cela m’a fait partir sur de mauvaises bases. Maintenant j’arrive mieux à programmer mais j’ai du mal à mémoriser ce qu’on apprend en cours pour la programmation, tout se réapprend au final sur le tas aux exercices.
- La partie théorique, j’ai vraiment du mal mais la partie programmation, je m’en sors plutôt bien. C’est pourquoi je voulais demander au professeur qui s’occupe de la théorie s’il pourrait donner un peu plus d’exemples chiffrés lorsque l’on fait face à des algorithmes complexes.
- La volonté des deux profeseurs est très bonne. Personellement j'aimerais bien que les présentations du cours soient moins dense je trouve que tout ce qui est dit est déjà sur les slides et nous avons beaucoup trop d'information à gérer
- Le cours est bien. Néanmoins, les exercices de programmation sont assez complexes et longs et j’ai le sentiment que parfois je n’ai pas les armes nécessaires (avec le cours) pour les affronter.
- Les cours ("theorie" et "programmation") sont bien engagé, il est facile de suivre leurs development. De plus, les transparent contient des point clés et sont trés claire. J'aime bien avoir des enregistrement à notre disposition. J'aimerait avoir les corrigés de "theory" avant, si possible le vendredi même, pour pouvoir les étudie pedant le weekend.
- Les cours d’information, calcul et communication sont très bien expliqués par des enseignants clairs et concis. Ils répondent à toutes les questions faite en cours et en séance d’exercices, merci.
- Les deux cours sont intéressants et le contenu est bien expliqué.
- Les deux cours sont très bien faits.
- Les deux cours sont très intéressants et bien expliqués.
- Les deux profs sont très gentils et font tout pour que le cours soit le meilleur possible.
- Les enseignants sont très pédagogiques, le cours est souvent très clair. Cependant, je trouve dommage que pour la partie programmation on fasse à chaque fois des rappels qui empiètent beaucoup sur notre seule heure de cours et ce qui fait que nous n'arrivons que rarement à la fin des slides prévues.
- Nos deux professeurs sont très sympathiques et passionnés également. De plus ils mettent tout a disposition pour notre réussite et de nombreuses ressources pour progresser sont disponibles.
- Parfois, lors du cours de programmation, il arrive que nous n’ayons pas le temps de finir le cours et le professeur passe assez rapidement sur certaines slides…
- Partie théorique assez compliqué mais très intéressante
- Partie théorique: série avec plus d'exercices, on a besoin de plus d'entraînement pour vraiment comprendre les concepts Partie pratique: je trouve que les exercices sont rédigés de manière compliquée pour des personnes dont le français n'est pas la langue maternelle.
- Pour la partie programmation: J'apprécie beaucoup les rappels fait en début du cours de la semaine précédante, mais par moment ils sont un peu trop longs...
- Pour la partie théorique je trouve que c'est dommage de ne pas avoir le temps de tout faire pendant les cours car après ça rend les exercices encore plus compliqué.
- Pour la théorie, c'est bien expliquer. Le prof est très gentil. Malgré que la matière est un peu difficile a comprendre car c'est un nouveau concept. par rapport a la pratique (python), j'aimerais que le prof nous donne plus de technique en programmation afin qu'on maitrise mieux le programme informatique. Python c'est facile mais il fait juste savoir le langage et s'entrainer encore plus. Pour résumé, je trouve la matière intéressante, mais un peu difficile .
- Pour la théorie, je trouve vraiment bien qu'on ait beaucoup de ressources Pour la programmation, je trouve bien le temps passé sur les démos même si ça fait que c'est un peu juste au niveau du temps.
- Prog très bien mais le cours d’algo est parfois un peu décalé par rapport aux séries
- Python: vous parlez très vite, ce qui rend le cours difficile à suivre pour une personne dont le français n’est pas la langue maternelle.
- théorie : ce serait bien si il y avait un peu plus de pauses entre les explications pour avoir le temps de digérer toutes les informations programmation : expliciter un peu mieux ce qui se trouve sur les slides à l'oral
- Théorie : cours clair, détaillé et dynamique. Rien à signaler de particulier. Pratique : idem, cours clair et bien détaillé avec les exemples d'algorithmes sur Python que vous faites en classe en expliquant votre raisonnement est très agréable et permet de voir comment nous pouvons raisonner pour résoudre de tel problème.
- Théorie: Un très bon cours et un professeur passionné. De bonnes séries d'exercices qui permettent de comprendre les concepts. De très bon assistants. Python: Un bon cours (malgré le fait que l'on termine rarement ce qui était prévu et devions reprendre certains concepts la semaine suivante) et un professeur passionné. De bonnes séries et bons assistants.
- tres bien
- Très clair, par contre j'estime que le cours de programmation avant les séances d'exos n'est pas très pertinent (à titre personnel bien sûr) car pour moi, on assimile la programmation par la pratique surtout (donc un cours en pdf par ex ça marche très bien + quest pour les assistants)

Feedback, résumé

Bien	Pas bien	Commentaire/action
Profs enthousiastes	*Exercices difficiles*	*peut-être, mais…*
Démos de code	*Infos midterm, infos miniprojet*	*→ aujourd'hui; après midterm*
Structure	*Montagne d'information sans structure*	*→ slides plus structurés*
Enregistrements	*Beaucoup à apprendre aux exercices*	*certes!*
Corrigés vidéo	*Trop abstrait*	*→ plus d'exemples et schémas*
	Cours pas pertinent, branche pratique	*modulable selon votre style*
	Trop d'heures d'exercices, pas assez de cours	*pas changeable*
	Trop de rappels, on n'arrive pas au bout des slides	*→ meilleure planification*
	Vous parlez très vite	*→ je me calme :)*

Cours de cette semaine

Imports

Sets

Dictionnaires

Import

```
import math
print(math.cos(math.pi))
```

Tout ce qui est défini dans math.py est accessible avec «math.xxx»

```
import math as m
print(m.cos(m.pi))
```

Tout ce qui est défini dans math.py est accessible avec «**m**.xxx»

```
from math import *
print(cos(pi))
```

Tout ce qui est défini dans math.py est accessible directement

```
from math import pi, cos as cosine
print(cosine(pi))
```

Seulement *pi* et *cos*, définis dans math.py, sont importés; *cos* est renommé *cosinus*

```
from typing import List
```

Aussi valable pour les types

Partager du code entre plusieurs fichiers

- Chaque fichier *.py* est un module
- Vous pouvez donc créer et importer vos propres modules

Dans *mytools.py*

```
from typing import List

def double(values: List[int]) -> List[int]:
    return [2 * x for x in values]

def make_string(values: List[int], separator: str = ", ") -> str:
    return separator.join([str(x) for x in values])
```

Dans un autre fichier du même dossier

```
from mytools import *
print(double([1, 2, 3]))
print(make_string([1, 2, 3], separator=" -> "))
```

Variante:

```
from mytools import double as dbl
print(dbl([1, 2, 3]))
```

on renomme une fonction

Cours de cette semaine

Imports

Sets

Dictionnaires

Set

- Un set (ensemble) est similaire à une liste, mais
- … n'a pas d'ordre intrinsèque
  - Pas possible d'utiliser l'indexation [i] ou le slicing [x:y]
- … contient un élément au plus une fois
  - et permet de tester rapidement s'il contient un élément ou non
- Objets modifiables (non immuables) comme les listes

Comparaison List/Set

```python
from typing import List, Set


my_list: List[str] = ["bonjour", "hello", "bonjour"]
print(len(my_list))    # 3
print(my_list[2])      # 'bonjour'
my_list = []           # liste vide


my_set: Set[str] = {"bonjour", "hello", "bonjour"}
print(len(my_set))     # 2
#print(my_set[2])      # pas possible, le set n'a pas d'ordre
my_set = set()         # set vide - pas {}


my_set = set(my_list)  # conversion de liste en set
my_list = list(my_set) # conversion de set en liste
```

Listes: avec la notation [ ]

Sets: avec la notation { }

Éléments dupliqués ne sont pas ajoutés une seconde fois

Autres méthodes utiles sur les sets

- `my_set.add(x)` — ajouter un élément *(listes: `append(x)`)*
- `my_set.clear()` — tout effacer
- `my_set.remove(x)` — supprime `x`
- `x in my_set` — teste si `my_set` contient une valeur `x`
  - `if x in my_set: ...`
  - `if x not in my_set: ...`
  - *Complexité constante!*
- Méthodes pour l'union, l'intersection ou encore la différence de plusieurs sets
  - Serait aussi possible avec les listes, mais plus lent qu'avec les sets

Cours de cette semaine

Imports
Sets
Dictionnaires

Dictionnaire

- Un dict (dictionnaire) est une structure qui relie des clés à des valeurs
- Conceptuellement, peut être vu comme un tableur à deux colonnes
  - En plus flexible surtout sur la deuxième colonne
- Objets modifiables (non immuables) comme les listes et les sets

Clé	Valeur
`"Alex"`	`9`
`"Émelyne"`	`8`
`"Victor"`	`5`

Démo

Dictionnaire: exemple

```python
from typing import Dict

ages: Dict[str, int] = {"Alex": 8, "Émelyne": 7, "Victor": 4}
print(ages)           # {'Alex': 8, 'Émelyne': 7, 'Victor': 4}

print(ages["Alex"])       # 8
# print(ages["Sandra"])   # erreur d'exécution

if "Sandra" in ages:      # seulement si la clé est présente
    print(ages["Sandra"])
else:
    print("n/a")

ages["Nathan"] = 4

print(ages)
# {'Alex': 8, 'Émelyne': 7, 'Victor': 4, 'Nathan': 4}

del ages["Nathan"]
```

Clé	Valeur
"Alex"	9
"Émelyne"	8
"Victor"	5
"Nathan"	4

Autre exemple: fréquence de mots

```python
poem: List[str] = ... # une liste de mots


counts: Dict[str, int] = {}

for word in poem:
    if word not in counts:
        counts[word] = 1
    else:
        counts[word] = counts[word] + 1


print(counts["les"])


for key in counts:
    print(f"{key} -> {counts[key]}")
```

Au début, le dictionnaire est vide

Pour chaque mot du poème, on regarde si le dictionnaire le connaît ou pas et on compte en fonction de ça

Les nouveaux mots ont un compteur de 1

Les nouveaux mots connus ont un compteur incrémenté

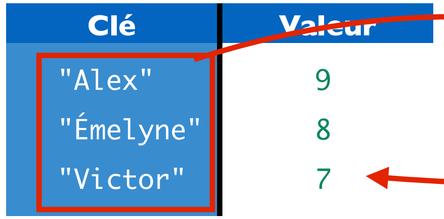
À la fin, le dictionnaire contient une valeur pour chaque mot différent du poème. On peut récupérer la valeur liée à un mot donné ainsi

On peut aussi itérer en demandant au dictionnaire de nous donner tous les mots qu'il connaît

Itération sur les dictionnaires

```
counts: Dict[str, int] = {}
```

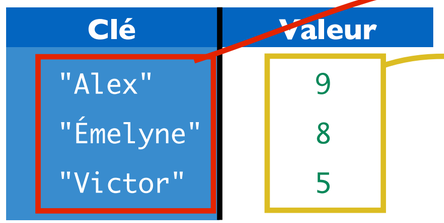
À des clés de type *str*, ce dictionnaire faire correspondre des valeurs de type *int*




```
for key in counts:
    value = counts[key]
    ...
```

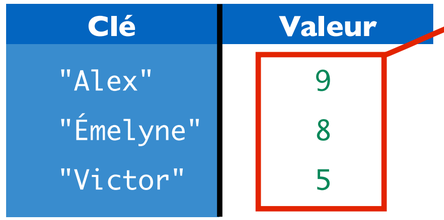
Un *for-in* normal pour un dictionnaire donne la clé à chaque itération de la boucle

On peut bien sûr toujours récupérer la valeur




```
for key, value in counts.items():
    ...
```

Cette itération sur le résultat de la méthode *items()* donne à la fois la clé et la valeur à chaque itération!




```
for value in counts.values():
    ...
```

On peut aussi itérer seulement sur les valeurs, mais c'est moins fréquent, et on ne peut pas facilement récupérer la clé

Exemple plus intéressant: *MyLittleFacebook*

```
from typing import *

# MyLittleFacebook
friendships: Dict[str, Set[str]] = {}

def add_friends(name1: str, name2: str) -> None:
    if name1 in friendships:
        friendships[name1].add(name2)
    else:
        friendships[name1] = {name2}
    if name2 in friendships:
        friendships[name2].add(name1)
    else:
        friendships[name2] = {name1}


→   add_friends("Alex", "Victor")
    add_friends("Alex", "Emelyne")
    add_friends("Alex", "Emelyne")
    add_friends("Emelyne", "Rose")
```

Clé		Valeur associée
`"Alex"`	→	`{"Victor"}` `"Emelyne"}`
`"Victor"`	→	`{"Alex"}`
`"Emelyne"`	→	`{"Rose"}` `"Alex"}`
`"Rose"`	→	`{"Emelyne"}`

Résumé Cours 6

- On utilise `import` ou `from ... import` pour réutiliser du code défini dans un autre fichier
- Un set (de type `Set[T]`) est un peu comme une liste, mais assure l'unicité des éléments
  - Pas d'ordre intrinsèque; on ne peut pas récupérer l'élément i
  - On peut convertir entre des listes et des sets facilement
- Un dictionnaire (de type `Dict[K, V]`) fait correspondre des clés à des valeurs
  - Avec un for-in normal, on itère sur les clés d'un dictionnaire
  - La recherche de la valeur correspondant à une clé donnée est optimisée