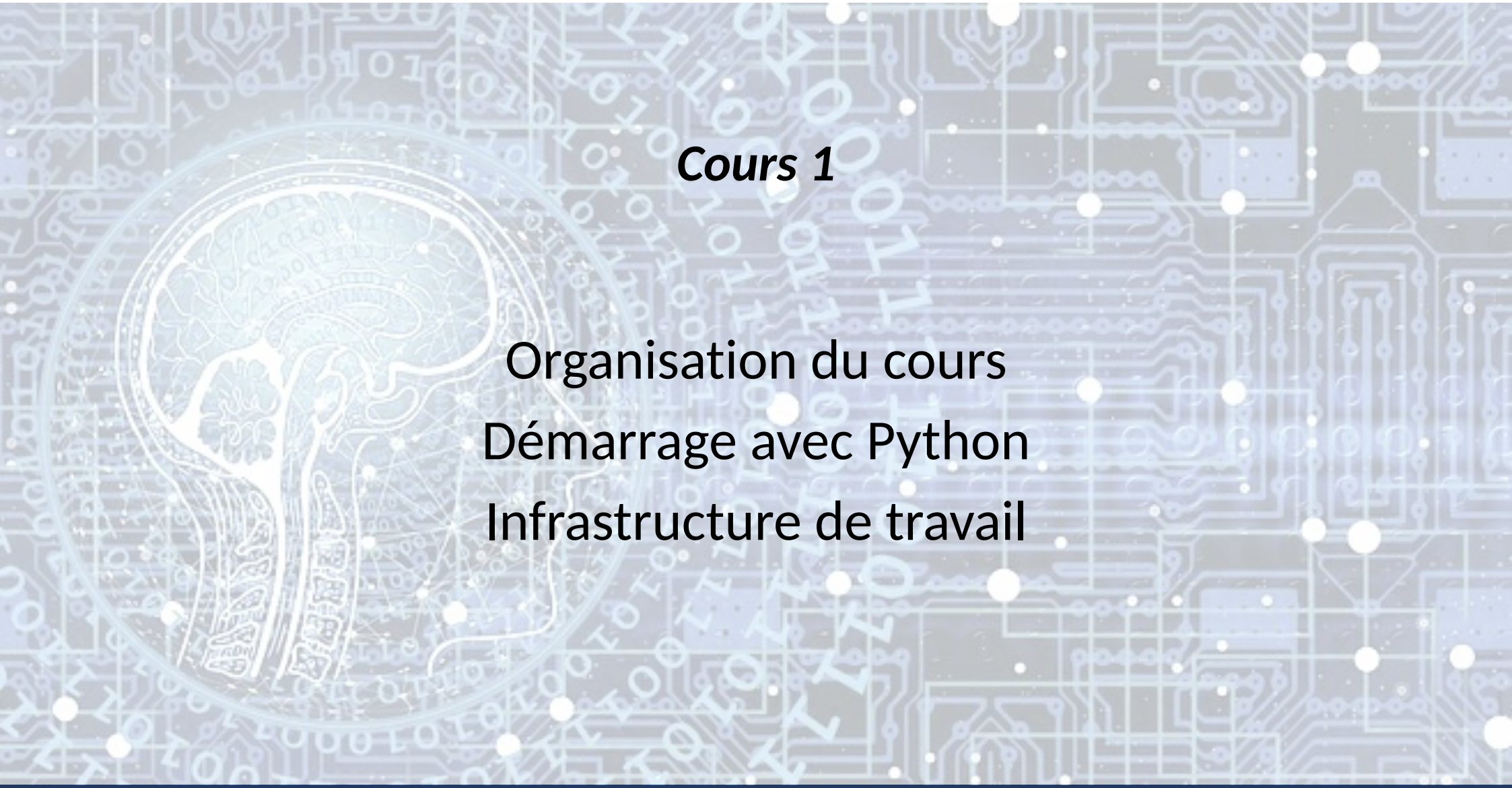
ENG-209

Data science pour ingénieurs avec Python

Cours 1

Jean-Philippe Pellet, Éric Bouillet, Olivier Verscheure

8 septembre 2025



Cours 1

Organisation du cours

Démarrage avec Python

Infrastructure de travail

Cours 1

Organisation du cours

Démarrage avec Python

Infrastructure de travail

Présentations — enseignants

Jean-Philippe Pellet

1re partie — ***sauf aujourd'hui!***

Python pour habitué·e·s de C++;
découverte de Numpy & Pandas

Éric Bouillet

Olivier Verscheure

2e partie

Data Science en Python

Présentations — assistant·e·s

Robin Guillaume-Gentil

Taliesin Perez

Agenda — 1re partie

Date	Contenu
08.09.2025	Démarrer en Python; différences notables avec C++
15.09.2025	Structures de données: *list, set, dict*; dataclasses
22.09.2025	*Férié, Jeûne fédéral*
29.09.2025	Pandas: lecture et écriture de fichiers, manipulations simples, conversions, réarrangements
06.10.2025	Pandas: filtres, groupes, opérations d'algèbre linéaire
13.10.2025	**Évaluation intermédiaire**
20.10.2025	*Pause de mi-semestre*

Sous réserve d'adaptations et modifications mineures

Agenda — 2e partie

Date	Contenu
27.10.2025	Introduction à la modélisation
03.11.2025	La régression linéaire
10.11.2025	La régression polynomiale
17.11.2025	Classification par régression logistique
24.11.2025	Classification par arbres de décision
01.12.2025	Performances d'un classificateur binaire
08.12.2025	La Data Science de bout en bout avec Python
15.12.2025	**Projet d'examen final** (poids : ⅔)

Sous réserve d'adaptations et modifications

Communication

- **Moodle**
  - https://moodle.epfl.ch/course/view.php?id=3701
  - Communication officielles de nous à vous
  - Slides, vidéos, références, exercices, corrigés
  - Semaine par semaine
- **Ed Discussions**
  - Espace **ENG-209** https://edstem.org/eu/courses/2576/discussion
  - Communication en dehors des heures de cours

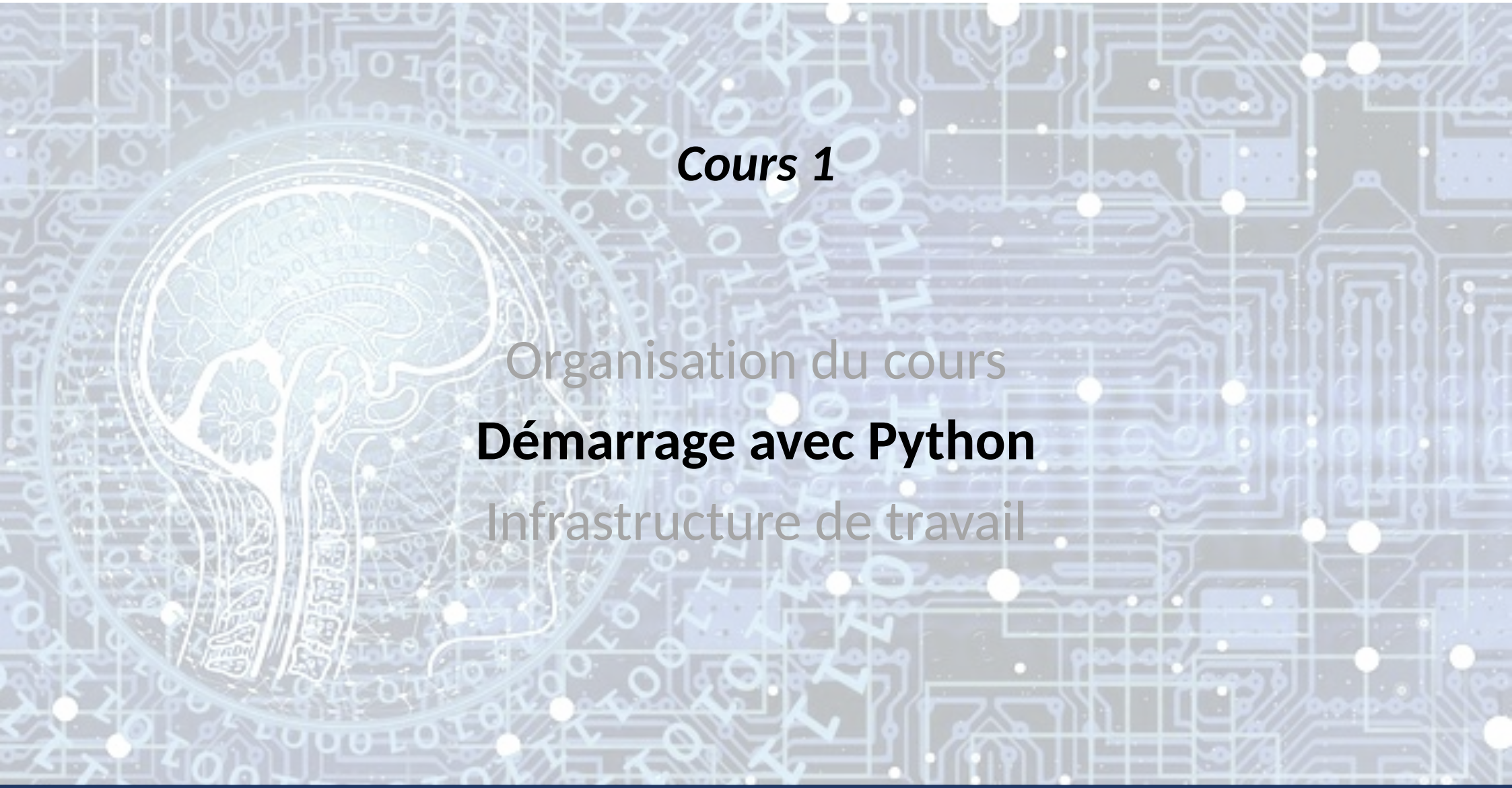
Format du cours (1re partie)

- **Cours**
  - Lundi, 10h15 – 11h, INF3, en présentiel
  - Diffusé en live sur Zoom, enregistré et mis à disposition peu après
- **Exercices**
  - Lundi, 11h15 – 13h, INF3, en présentiel
    - Enseignants et assistants en salle pour groupe en présentiel
    - Présence assurée via Zoom et Slack pour groupes à distance
  - Flexibilité sur la séparation précise cours/exercices
- **Hors heures officielles**
  - Ed toujours disponible — utilisez-le!

Sous réserve d'adaptations et modifications

Checklist

- **Vous êtes inscrit·e au cours sur IS-Academia**
  - Sinon: → http://is-academia.epfl.ch
- **Vous avez accès à la page Moodle et l'avez bookmarkée**
  - Sinon: mail à jean-philippe.pellet@epfl.ch
    - → https://moodle.epfl.ch/course/view.php?id=3701
- **Vous avez accès à l'espace Ed**
  - En lien depuis la page Moodle du cours



Cours 1

Organisation du cours

Démarrage avec Python

Infrastructure de travail

De C++ à Python

- **Le plus difficile est fait! :)**
- **Différences notables**
  - Syntaxe basée sur l'indentation à la place des accolades
  - Langage dynamiquement typé: pas de types à déclarer (mais…)
  - Interprété plutôt que compilé (mais…)
  - Pas de fonction "main", chaque fichier est exécutable
  - Gestion automatique de la mémoire
  - Structure de données faciles à déclarer et à manipuler

Les bases — print, variables

```
print("Welcome to Python")
# Comments preceded by hash sign
# on each line
```

Appel de fonction print *pour afficher du texte sur le terminal*

Chaînes de caractères: avec "..." *ou* '...'

 # Commentaires

```
version = 3.10

print(f"Our code will use version {version}")
```

Affectation d'une variable: sans type, avec déclaration «implicite»

f-strings: avec le préfixe f*, les expressions entre* { } *sont converties en strings et insérées*

```
version = "3.10, the latest release"
print(f"Again: version {version}")
```

Même si ce n'est pas recommandé, les variables peuvent faire référence à des valeurs de types différents dans le même bloc de code

Hey — pas de points-virgules!

Les bases — input, conversion, if

```
line = input("Enter your age: ")
age = int(line)


if age >= 18:
    print("Here's your beer.")
elif age >= 8:
    print("Here's an ice tea.")
else:
    print("Here's some water.")


print("Enjoy!")
```

Fonction input *pour lire une ligne lors de l'exécution*

Fonction int *pour convertir une chaîne en nombre entier*

Pas de parenthèses ou d'accolades. Un deux-point signale le début du corps du if*. Tout le corps est indenté*

if – elif – else*, chaque fois avec bloc indenté*

Toujours pas d'accolade pour fermer un bloc — il suffit de désindenter

Attention aux «copier-coller» avec des niveaux d'indentations différents

Les bases — conversion, types

```
line = input("Enter your height in meters: ")
height = float(line)

height_as_string = str(height)

print(f"'{line}'")
print(f"'{height_as_string}'")
```

Fonction float *pour convertir une chaîne en nombre à virgule flottante*

Fonction str *pour convertir beaucoup de choses en une chaîne de caractères*

Output pas forcément identique

```
msg: str = 'Welcome to Python!'
print(msg)

msg = 5
```

Les types vous manquent? Annotez vos variables

Ceci est noté ensuite comme erreur par le linter (mypy). Mais le fichier reste exécutable

Quelques types de base: int, float, str, bool

```
print(f"{msg=}, {line=}")
print(f"{msg:10}")
```

Les f-strings, plus puissants qu'il n'y paraît! Possible de changer/formater l'output [Lien]

Les bases — boucles, range

```
i = 0
while i < 10:
    print(i ** 2)
    i += 1
```

Boucle while*, test répété d'une condition*

Même principe qu'avec le if*, avec le deux-points et l'indentation*

** *pour exponentiation,* i += 1 *pour incrémentation*
*(*i++ *n'existe pas)*

Résultat: affiche les carrés de 0 à 81

```
for i in range(10):
    print(i ** 2)
```

for-in *avec un* range: *pour itérer facilement de manière concise*

```
range(0, 10)
```

Même effet, avec début explicité, toujours inclus. Fin toujours exclue

```
range(1, 10)
```

Début à 1

```
range(1, 10, 2)
```

Début à 1, incrément de 2

```
range(9, -1, -1)
```

Début à 9, fin à 0 (–1 est exclu), incrément de –1

Une particularité: le for… else

```
n = ...
for i in range(2, int(math.sqrt(n)) + 1):
    if n % i == 0:
        print("Not prime")
        break
else:
    print("Prime")
```

Le else *peut être indenté pour correspondre à un* for

Le bloc else *est alors exécuté uniquement si la boucle ne se termine pas de manière prématurée. Il y a donc en général un* break *dans la boucle, qui termine la boucle et saute ainsi le* else

```
n = ...
is_prime = True
for i in range(2, int(math.sqrt(n)) + 1):
    if n % i == 0:
        is_prime = False
        print("Not prime")
        break
if is_prime:
    print("Prime")
```

On peut bien sûr se passer du for… else avec un booléen supplémentaire

Les bases — fonctions

```
def is_even(x):
    return x % 2 == 0

if is_even(int(input())):
    print("Looks even!")
```

Fonction définie avec def*, nom, liste de paramètres, deux-points, et corps indenté*

Affiche Looks even! *si un nombre pair est entré lors de l'exécution*

```
def is_even(x: int) -> bool:
    return x % 2 == 0
```

Vous pouvez spécifier les types des paramètres et le type de retour

```
def say_hello(to: str = "my friend") -> None:
    print(f"Hello, {to}!")
```

None *est semblable à* void*, pas de valeur de retour*

Les paramètres peuvent avoir des valeurs par défaut

```
say_hello()
say_hello("Jane")
say_hello(to = "Mary")
```

Hello, my friend!
Hello, Jane!
Hello, Mary!

Les paramètres peuvent être nommés (et réordrés) lors d'un appel de fonction

```
def foo(a, b, /, c, d, *, e, f):
    ...
```

Possible de déterminer quels paramètres sont «nommables» (c–f) ou pas, voire à nommer obligatoirement (e–f) [Lien]

Une particularité: deux affectations

```
total = current = 0
```

En C++, l'affectation avec **=** *retourne la valeur affectée. Pas en Python.*

L'affectation multiple reste possible, mais on ne peut pas utiliser la valeur de retour d'une affectation avec **=** *dans une expression plus complexe (comme un test dans une boucle)*

```
def get_next() -> int | None:
    ...
```

*Imaginons une fonction qui livre soit un int, soit rien (*None*, qui est l'équivalent Python de* null/NULL/null_ptr*)*

```
item = get_next()
while item is not None:
    ...
    item = get_next()
```

Avec uniquement l'affectation simple avec **=***, l'appel* get_next() *doit être répété ici deux fois*

```
while (item = get_next()) is not None:
    ...
```

Ceci n'est pas possible…

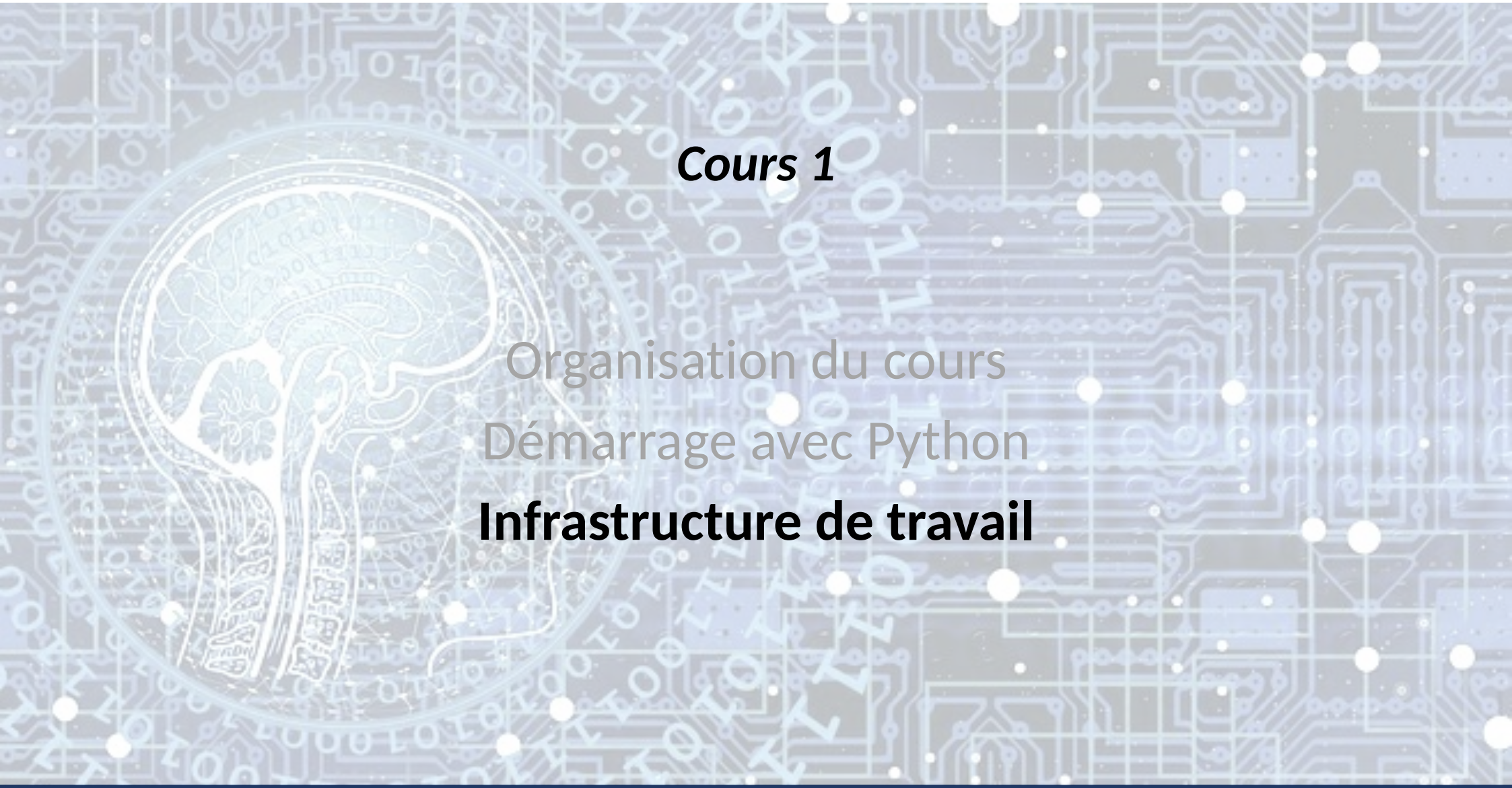
```
while (item := get_next()) is not None:
    ...
```

… mais ceci oui, en utilisant **:=** *plutôt que* **=** *pour l'affectation!*

Note: l'affectation avec **:=** *ne peut pas remplacer* **=** *dans une affectation «simple» du style* item = value *sans rien d'autre sur la ligne*

Résumé du cours d'aujourd'hui

- **Python est un langage dynamiquement typé**
  - La syntaxe est plus simple — symboles comme () ; en moins
  - L'indentation est significative
  - Les types peuvent quand même être indiqués
- **On retrouve les structures de contrôles de base**
  - if – elif – else
  - while, for (avec des différences)
- **On peut facilement définir et appeler des fonctions**
  - Flexibilité dans la manière de définir les paramètres et de passer des arguments



Cours 1

Organisation du cours

Démarrage avec Python

Infrastructure de travail

Séance d’exercices: sur les machines virtuelles

Démo

Autoévaluation — objectifs

- **Je suis capable de…**
  - me connecter sur Renku, démarrer et accéder à ma VM, ouvrir les notebooks
  - affecter et lire des variables, utiliser les opérateurs arithmétiques standards
  - utiliser les *f-strings* en insérant des expressions entre accolades
  - écrire des *if*, avec ou sans partie *elif*, avec ou sans *else*
  - écrire des boucles *for* qui itèrent à travers différentes *ranges*
  - déclarer et appeler des fonctions avec ou sans paramètres (avec ou sans valeur par défaut), avec ou sans valeur de retour
  - rechercher de la documentation en ligne pour découvrir seul·e de nouvelles fonctions