CPP 2020

Rapport sur le projet pédagogique

Ph. Langlois

1er avril 2021

Table des matières

1 Synthèse

Ce rapport présente les avancées du projet pédagogique réalisé dans le cadre d'un CPP au premier semestre de l'année universitaire 2020-2021.

L'objectif de ce projet pédagogique est de concevoir et mettre en oeuvre des contenus et des outils logiciels qui permettent un meilleur apprentissage de l'algorithmique et de la programmation pour des étudiantes et des étudiants de licence 1 d'informatique ou de mathématiques (semestre 2). Dans la nouvelle offre de formation 2021-2025 de l'établissement, ce projet a pour cadre une unité d'enseignement (UE) de 72h étudiant au semestre 2 des licences d'informatique et de mathématiques.

L'objectif de ce projet, défini au premier semestre académique 2019-2020, est globalement atteint tout en intégrant les adaptations nécessaires au contexte sanitaire actuel et son impact sur nos méthodes et nos conditions d'enseignement depuis mars 2020. Ce projet a aussi significativement bénéficié des formations et du suivi du CAP de l'établissement, en particulier pour répondre au mieux à la déclinaison de nos diplômes en termes de compétences.

Les principaux résultats à ce jour sont la définition et l'intégration de quizz exploités comme une véritable nouvelle activité pédagogique ainsi que de TP auto-corrigés pour permettre une auto-évaluation de/par l'étudiant ainsi que son évaluation formative et sommative. La définition et l'intégration de ces activités découlent de l'analyse et de la reformulation préalable sous forme de compétences des objectifs de l'UE ciblée par ce projet, et aussi de l'identification de certaines faiblesses de l'approche pédagogique jusque-là mise en place.

Ces résultats sont en cours d'expérimentation progressive, et ce autant que la situation actuelle le permette (semestre 2 2020-2021). Cette expérimentation, complétée du retour d'expérience des étudiants de L2 et L3 sur cette UE fondamentale à la suite de leur formation, permettra les ajustements et les développements complémentaires à la réussite de la mise en place de cette UE ainsi améliorée pour l'année académique 2021-2022.

Ce rapport s'articule en deux parties. Les aspects à ce jour les plus significatifs du projet sont d'abord présentés de façon synthétique. Neuf annexes plus illustratives ou plus techniques complètent ensuite ce document.

2 Le projet initial et son évolution

L'objectif général de ce projet est de concevoir et mettre en oeuvre des contenus et des outils logiciels pour un meilleur apprentissage de l'algorithmique et de la programmation par les étudiantes et les étudiants de licence 1 d'informatique ou de mathématiques (semestre 2). Il a pour cadre une unité

d'enseignement unique de 72h étudiants issue de la refonte de 2 UE actuelles (87h étudiants au semestre 2 aussi).

Le projet dans sa forme initiale (janvier 2020) est rappelé dans la dernière annexe de ce document.

Trois directions avaient été définies en ce sens :

— permettre une pratique améliorée,
— permettre un scénario d'apprentissage adapté au projet personnel de formation de l'étudiant.e,

— favoriser le présentiel comme amplificateur du travail personnel.

Les deux premières directions ont été significativement influencées, d'abord par l'expérience du premier confinement sanitaire (de mars 2020 à la fin du second semestre académique durant laquelle la moitié environ de ces UE avaient été réalisées en distanciel), puis par les formations et l'accompagnement proposés par le CAP de l'UPVD, et enfin par le second semestre 2021 actuellement en cours sous forme (au mieux) hybride pour l'enseignement concerné par ce projet.

Mon intention initiale était *surtout* de favoriser l'autonomie d'apprentissage de l'étudiant, à l'aide de contenus et d'outils dans l'esprit de ceux utilisés pour la préparation des compétitions d'algorithmique et d'informatique comme Algoréa ou les Olympiades d'informatique accessibles par la plateforme de France-IOI. Aujourd'hui, ma lecture de cet objectif a évolué pour mieux répondre aux attentes de la majorité de *notre public étudiant* et aux *contraintes techniques imposées* par la situation actuelle.

En effet, **l'autonomie de l'apprentissage ne se décrète pas. Elle s'apprend ! En particulier, l'évaluation guide l'apprentissage et participe ainsi à l'apprentissage de cette autonomie.**

Pour cela, les attendus de l'apprentissage doivent être :

1. d'abord très clairement définis (en terme de compétences par exemple),
2. assortis des repères d'évaluation qui les ancreront (déclinaison formative) ou les valideront (déclinaison sommative) et
3. du scénario (pédagogique et technique) qui accompagnera l'étudiant dans cet objectif.

Ce préalable a notablement enrichi le projet initial. En effet, les formations et l'accompagnement du CAP de l'UPVD m'ont permis de mettre à jour certaines faiblesses de la démarche pédagogique mise en place jusque-là – au moins dans ces UE sous ma responsabilité depuis quelques années.

D'abord, l'organisation pédagogique souffrait de l'absence d'une explicitation, précise pour l'étudiant, des attendus *au plus près* du déroulement de l'UE. Le planning de l'apprentissage était globalement orchestré par les deux contrôles de connaissances, à mi-parcours et en fin d'UE. Malgré la mise en place de deux grilles de lecture des cours, des exercices de TD et de TP (deux objectifs d'apprentissage définis selon l'importance de ces matières pour le projet personnel de l'étudiant), *trop peu d'évaluations* étaient proposées *pour accompagner* l'étudiant dans *les premières étapes de son apprentissage* (les étapes

“connaitre” et “comprendre” de la taxonomie de Bloom). Le contexte sanitaire récent a accentué cet aspect : cet accompagnement était trop diffus, trop informel car essentiellement basé sur la dynamique d’interaction enseignant-étudiant possible lors des séances de travaux dirigés en présentiel.

Au delà de son intérêt pédagogique, cette démarche est désormais nécessaire pour **la nouvelle offre de formation 2021-2025 de l’établissement** définie en terme de compétences selon les recommandations du Ministère.

3 Principaux résultats

3.1 Approche par compétences

Le chapitre thématique est la brique de base du séquençage de cet enseignement. J’ai donc défini et explicité les compétences (savoirs, savoirs-faire et attitudes) et des grilles critériées **au niveau de chaque chapitre**. Ceci complète de façon adaptée l’approche suivie pour les diplômes (licence d’informatique et licence de mathématiques) où les compétences générales, extraites du référentiel des compétences des mentions de licence ou la fiche 24514 et projetées dans chaque UE, doivent pouvoir être validées.

Ce travail, à première vue formel, s’est avéré fécond. Expliciter de façon conjointe les attendus et leurs évaluations a conduit à la définition d’un volume important de quizz, à la mise à jour du cours et de son support avec l’ajout de chapitres nouveaux pour satisfaire des compléments “non-dits” de pré-requis insuffisamment acquis au semestre précédent, et au re-découpage ou à la réorganisation de certains chapitres en cohérence avec l’évaluation définie.

Les compétences ciblées par cet UE et un exemple d’identification-explicitation effectuée par chapitre sont respectivement proposés en Annexe 1 et 2.

3.2 Quizz : une véritable nouvelle activité pour l’apprentissage de l’étudiant et son auto-évaluation

L’usage de quizz était absent de cet enseignement – tout comme de l’ensemble de ma formation universitaire et plus généralement des premiers et seconds cycles universitaires des années 80-90 (au moins en informatique et mathématiques). La démarche d’identification-explicitation précédente a créé un cadre qui incite naturellement à développer de tels exercices.

J’ai défini **des quizz organisés selon les compétences de chaque chapitre** précédemment identifiées **et quantifiés** (notés) de façon à obtenir une grille critériée par chapitre. Ceci permet d’introduire une véritable nouvelle activité pour l’apprentissage de l’étudiant et son auto-évaluation.

Ces quizz permettent bien sûr de réduire significativement la faiblesse de l'accompagnement de l'étudiant dans de les premières étapes de son apprentissage préalablement identifiée. Ils sont utiles pour la réactivation en début de séance des acquis de la séance précédente.
J'exploite aussi ces quizz accompagnés de *sondages interactifs* ou *asynchrones* sur la *perception par l'étudiant* de son niveau d'acquisition. Ces sondages utilisent les grilles critériées qui dirigent ces quizz. En fin de chapitre par exemple, un sondage interactif et anonyme est effectué avant de "découvrir" le quizz, puis après son exploitation en temps limité. De façon plus asynchrone, un ou deux sondages nominatifs sont demandés plus longtemps après laissé le quizz ouvert. L'association du quizz et du sondage interactif (en cours ou en travaux dirigés) permet d'identifier rapidement les points à reprendre pour la majorité des étudiants (en séance ou d'une séance sur l'autre). L'association du quizz et du sondage asynchrone nominatif permet une interaction mieux ciblée durant les séances de travaux dirigés, en particulier en mode distanciel.

Mais ces quizz sont surtout **un outil d'auto-évaluation pour l'étudiant.** Ils l'accompagnent dans sa propre évaluation de son niveau d'acquisition des compétences de la matière.
La notation du quizz est définie compétence par compétence et pour chacune des compétences explicitées pour le chapitre. Chaque étudiant dispose ainsi d'une explicitation quantifiée de son niveau d'acquisition dans chaque compétence visée, *ie.* de sa projection dans la grille critériée du chapitre. Cette quantification est – ou devrait être – le reflet des résultats du sondage asynchrone effectué par l'étudiant. En effet, l'association quizz+sondage asynchrone place l'étudiant dans une posture d'auto-évaluation : il est en mesure de *mieux* répondre à la question "où en suis-je dans l'acquisition de telle ou telle compétence ?". Il est ainsi mieux accompagné dans sa *démarche d'autonomie d'apprentissage.*

En pratique, ces quizz restent ouverts sous moodle sans limite d'essais. Les étudiants sont relancés au moins deux fois sur ces sondages asynchrones (une semaine puis trois à quatre semaines après la fin du chapitre correspondant). D'autres ressources de ce type sont aussi signalées lorsqu'elles sont disponibles sur le net et adaptées avec les objectifs et la progression de l'UE.

Les sondages interactifs et les sondages asynchrones sont respectivement réalisés avec l'outil [mentimeter] (https://www.mentimeter.com) et l'outil "sondage moodle classique".

Un exemple de ces quizz est proposé en Annexe 3.
Un exemple de sondage interactif basé sur la grille critériée d'un chapitre est proposé en Annexe~4.

3.3 Des TP auto-corrigés formatifs ou sommatifs

L'objectif initial principal de ce projet était la mise en place d'outils logiciels adaptés aux spécificités de l'apprentissage de l'algorithmique et de la programmation, apprentissage qui serait réalisé avec un maximum d'autonomie et en cohérence avec l'hétérogénéité des pré-acquis des apprenants et de leur projet.

Après un état de l'art des différentes solutions techniques possibles (par exemple les environnements de "juge automatique" comme camisole ou domjudge, et aussi de CodeRunner, alternative sous la forme d'un plug-in moodle récemment proposée à l'UPVD), complété par l'expérience du confinement de mars 2020 – et la crainte aujourd'hui justifiée d'une nouvelle situation similaire –, j'ai choisi de *ne pas rajouter de la difficulté technique à la difficulté d'apprentissage.*

J'ai donc développé une démarche d'**auto-correction dans les notebooks jupyter** afin de conserver **ce support unique** déjà utilisé pour les cours, les exercices et les TP tout au long du semestre (voir section 1.6.1 du projet initial). L'étudiant est ainsi conforté dans son apprentissage *de*, et *avec*, un unique environnement logiciel – dont la maîtrise est un des savoirs-faire technique et transversal visé par cette UE (compétence transversale). Ce choix réduit aussi significativement une des difficultés majeures apparues lors du premier confinement et par exemple rapportée par J.-M. Gilliot (IMT Nantes) lors de la journée Pandémique : lorsque pandémie et informatique se rencontrent organisée par la Société Informatique de France en novembre 2020. Les travaux (dits) pratiques (ce sont plutôt des travaux dirigés sur ordinateurs) sont indispensables à l'apprentissage de l'Informatique. Leur mise en place, leur suivi et leur gestion est apparu comme le principal frein à l'apprentissage de la programmation dans le contexte distanciel – un autre étant l'évaluation certificative face à la fraude induite par ce contexte distanciel.

En pratique, chaque question d'un sujet de TP est accompagnée de "cellules de test" visibles et exécutables par l'étudiant. Ces tests valident ou non la solution qu'il propose : un test qui échoue signifie que la solution de l'étudiant est incorrecte. Le traitement est alors interrompu pour permettre à l'étudiant de corriger et re-soumettre une solution, et ce sans limitation d'essai. Deux niveaux d'auto-correction sont possibles : le concepteur du sujet peut définir un test avec une réponse fermée ("solution correcte : oui/non") ou une réponse plus informative qui permet à l'étudiant d'identifier l'origine de la version erronée du traitement qu'il propose.

De façon complémentaire et similaire à l'approche proposée par l'environnement nbgrader développé à U.C. Berkeley, ce mécanisme d'auto-correction peut être aussi adapté pour produire des sujets avec des tests visibles (auto-correction) mais aussi des tests cachés (validation). Ces derniers contribuent à une première phase de correction automatisée (de certaines parties) des devoirs des étudiants et aider à produire plus vite un *feed-back* individualisé lorsque c'est opportun.

Cette mise en oeuvre nécessite en amont de sensibiliser les étudiants à la démarche de tests unitaires du logiciel – qui constitue aussi un des savoirs-faire disciplinaires de la Licence – et aussi d'étudier le mécanisme des exceptions (ici en python) un peu plus tôt dans le déroulement "habituel" du semestre.

De façon globale, cette solution à base de notebook jupyter répond aux objectifs de **pratique améliorée** définis par les améliorations listées dans la section 1.4.1 du projet initial. Comme les aspects précédents, cette nouvelle forme de TP est expérimentée lors du semestre en cours et dans un contexte distanciel,

à la fois de façon formative et aussi sommative (contrôle continu).

Un exemple de ces TP auto-corrigés (contrôle continu) est proposé en Annexe 4.

3.4 Satisfaire l'hétérogénéité du public apprenant

Le projet initial insistait sur l'augmentation de l'hétérogénéité des acquis des étudiants primo-entrants à partir de la prochaine rentrée universitaire 2021. En effet, la nouvelle spécialité de baccalauréat "Numérique et Science Informatique" (NSI) introduit environ 300 heures d'enseignement d'informatique pour les lycéens concernés en première et terminale. Le programme correspondant, très ambitieux, couvre une partie très importante de l'UE de ce projet. En pratique, il est à ce jour difficile d'apprécier l'impact réel qu'aura cette spécialité sur notre public étudiant (nombre et pourcentage d'étudiants concernés, niveau d'acquisition effectif, ...).

Après un état de l'art des ressources NSI (sites institutionnels et bibliographie assez exhaustive), j'ai adapté le programme et les supports d'enseignements pour satisfaire au mieux l'hétérogénéité de ces acquis. Concrètement, cela se traduit par l'**ajout de ressources** (chapitres, devoirs, sujets de TP), **des changements de choix pédagogiques** (comme par exemple la représentation des tableaux par des listes-python) et l'**utilisation importante de l'outil pythontutor** de visualisation interactive de l'exécution d'un code.

Un exemple de pythontutor est présenté en Annexe 6. Une photographie des ressources NSI est présentée en Annexe 8.

3.5 Un sondage type EEE pour les L2 et L3

Dans la dynamique des formations du CAP, j'ai préparé un sondage de type EEE (évaluation des enseignements) **à l'attention des étudiants de L2 et de L3** (licence de mathématiques et licence d'informatique) et qui concerne ces UE fondamentales **vues en L1**. Il s'agit de pouvoir profiter de leur analyse à la fois avec un recul suffisant et en distinguant leur projet professionnel.

La première version de ce sondage est présentée en Annexe 7. Pour profiter du meilleur recul, ce sondage sera diffusée après les vacances de printemps et les réponses seront analysés en juin (2021).

4 Ce qu'il reste à faire

A ce jour, environ 80% du contenu de l'UE a été repris selon la démarche décrite. La transformation de l'ensemble des ressources de l'UE est en cours d'achèvement. Une seconde passe de cette transformation profitera de l'expérimentation et de sondages en cours.

4.1 Exploiter le premier retour d'expérience du semestre en cours et des sondages des L2 et L3 pour adapter et compléter la démarche entamée pendant ce CPP

L'expérimentation en cours ce semestre s'effectue dans un contexte particulier et assez défavorable. Le premier confinement de mars 2020 avait (simplement) nécessité d'adapter aux conditions distancielles une dynamique d'apprentissage mise en place en présentiel la moitié du semestre 2 et dans la continuité du semestre 1. Cette adaptation avait été techniquement assez simple pour cette UE déjà entièrement dématérialisée sous moodle. La très grande majorité des étudiants avaient pu profiter du début du semestre pour configurer correctement leurs ordinateurs personnels de façon à pouvoir continuer à travailler comme dans les salles d'informatique de l'UPVD, et surtout, être dans une dynamique d'apprentissage adaptée car lancée de façon classique. Les conditions actuelles sont significativement différentes. Aucune dynamique d'apprentissage n'a pu être mise en place par du présentiel depuis le début de l'UE. Aucune préparation du matériel personnel des étudiants n'a pu être effectuée (il y avait d'autres priorités). A ce jour, la majorité des étudiants de L1 a, depuis un an, étudié en présentiel seulement 10 semaines environ. Les étudiants les plus en difficulté sont dans une situation encore plus défavorable qu'en période normale, tout comme ceux "entre deux eaux" qui étaient maintenus à flot grâce à un soutien important et régulier du présentiel-enseignant.
Dans ce contexte, le retour de l'expérience actuelle sera certainement plus limité qu'il aurait pu être et l'analyse de son efficacité et de sa pertinence sera assurément biaisée.

Il faudra donc attendre le retour à une situation plus favorable pour profiter du retour d'expérience *en temps réel*. En revanche, les sondages recueillis auprès des L3 seront certainement les plus instructifs, à la fois par leurs reculs et les conditions plus habituelles de l'apprentissage en L1. Par chance, la promotion de L2 concernée par ce sondage est d'un très bon niveau global. Cela devrait (leur) permettre de relativiser l'impact des deux confinements et ainsi fournir d'un retour d'expérience aussi très intéressant et que je ne manquerai pas d'exploiter pour pour adapter et compléter la démarche entamée pendant ce CPP.

Dans un second temps, la question suivante sera examinée.

4.2 Comment mieux affiner la quantification des quizz en fonction du projet personnel de l'étudiant ?

En amont de ce projet et depuis quelques années, j'avais introduit globalement et explicitement pour ce module, *deux parcours pédagogiques différents*. Un parcours dit "objectif 10" constitué du "minimum à avoir compris pour ne pas perdre de points", parcours destiné aux étudiants en difficulté, ou sans intérêt important de la matière pour réaliser leur projet personnel de formation et professionnel. Un parcours dit "objectif 20" défini pour les étudiants qui souhaitent continuer en informatique et composé des notions à maîtriser pour réussir au moins jusqu'en L3 Informatique. L'orientation vers un

objectif ou un autre est au choix de l'étudiant que l'on conseille selon ses résultats du semestre 1 et au fur et à mesure des séances de travaux dirigés : certains étudiants migrent de l'un à l'autre selon les notions et … leur investissement.

Une première distinction de ces deux objectifs a été introduite dans la quantification des quizz mais de façon assez globale. Les questions sont notées de façon principalement uniforme et conduisent à une note qui guide le *feed-back* global suivant.

- $100\% = note$: très bon travail de niveau objectif 20
- $100\% > note \geq 80\%$:
  - Objectif 20 : cours à approfondir et/ou questions à relire avec attention
  - Objectif 10 : excellent
- $80\% > note \geq 60\%$:
  - Objectif 20 : cours à reprendre en profondeur
  - Objectif 10 : certaines notions du cours sont à reprendre pour garantir le 10
- $40\% > note$: Veuillez reprendre la lecture de la totalité du cours, noter les points qui vous posent des difficultés et en parler à votre chargé de TD dès la prochaine séance. Ne laissez pas le temps passer, ne baissez pas les bras !

Cette première solution est un peu trop globale. Elle nécessite souvent un travail un peu artificiel d'équilibrage de la quantification des questions afin de conduire à un *feed-back* cohérent. Elle ne permet pas facilement de relativiser l'acquis des compétences évaluées selon l'objectif poursuivi, ni de distinguer aisément le poids relatif de chaque compétences au sein d'un chapitre, ni l'importance du chapitre selon l'objectif poursuivi. En effet, la ligne verticale médiane du sondage ou les 50% de la notation du quizz actuellement utilisés n'ont pas toujours la même signification selon les chapitres et l'objectif poursuivi par l'étudiant. Au delà de la dimension technique ou du choix d'un mode de calcul plus sophistiqué, cette amélioration nécessite au préalable d'affiner les grilles critériées qui dirigent ces quizz pour mieux prendre en compte le projet personnel de l'étudiant.

5 Annexes

5.1 Les compétences des licences d'informatique et de mathématiques ciblées par l'UE

Lors de la conception de la nouvelle offre de formation 2021-2025, les compétences générales extraites du référentiel des compétences des mentions de licence ou la fiche 24514 pour la licence d'informatique et la licence de mathématiques ont été projetées dans chaque UE.

Cette UE d'algorithmique et programmation *contribue* aux compétences suivantes.

Compétences disciplinaires

- Choisir, sur des critères objectifs, les structures de données et construire les algorithmes les mieux adaptés à un problème donné.
- Analyser et interpréter les résultats produits par l'exécution d'un programme.
- Se servir aisément de plusieurs styles/paradigmes algorithmiques et de programmation (approches impérative, fonctionnelle, objet et multitâche) ainsi que plusieurs langages de programmation.

Compétences transversales

- Utiliser les outils numériques de référence et les règles de sécurité informatique pour acquérir, traiter, produire et diffuser de l'information ainsi que pour collaborer en interne et en externe.
- Identifier et sélectionner diverses ressources spécialisées pour documenter un sujet.
- Analyser et synthétiser des données en vue de leur exploitation.

5.2 Un exemple de compétences par chapitre

Cet exemple concerne le chapitre **Fonctions**.

5.2.1 Avoir les idées claires

- Distinguer définition et paramètres formels *vs.* appel et paramètres effectifs
- Distinguer spécification (en-tête, signature) : spécifier pour utiliser, pour vérifier *vs.* corps : implémentation du traitement
- Distinguer appelant *vs.* appelé : le rôle de l'appel, le rôle du return
- Distinguer variables locales *vs.* variables plus globales, identifier la portée des variables
- Distinguer appel = changement de contexte : trace de l'exécution *vs.* séquentialité des instructions écrites, distinguer dynamique *vs.* statique

5.2.2 Ce qu'il faut savoir faire

Cadre : en/pour python

- Définir et écrire la spécification d'une fonction qui réalise un traitement décrit en français, ou qui résoud un problème (simple) décrit en français
- Définir et écrire des appels simples (tests unitaires)
- Définir et écrire l'implémentation d'une fonction associée à une spécification

5.2.3 Pré-requis technique

Dans/avec un notebook jupyter :

- savoir mettre en oeuvre de la programmation simple en python (niveau semestre 1) dans un notebook jupyter
- savoir documenter cette programmation (énoncés, descriptions, …) avec markdown

5.3 Un exemple de quizz

Cet exemple concerne le chapitre **Fonctions**. Par souci de lisibilité de ce document, nous présentons ici ce quizz sous une forme différente du quizz moodle proposé aux étudiants.

Cette annexe continue avec 7 pages dont 6 numérotées hors document.

Programme du quizz

1. Distinguer définition et paramètres formels vs. appel et paramètres effectifs
2. Distinguer spécification (en-tête, signature) : spécifier pour utiliser, pour vérifier vs. corps : implémentation du traitement
3. Distinguer appelant vs. appelé : le rôle de l'appel, le rôle du return
4. Distinguer variables locales vs. variables plus globales, identifier la portée des variables
5. Distinguer appel = changement de contexte : trace de l'exécution vs. séquentialité des instructions écrites, distinguer dynamique vs. statique , distinguer dynamique vs. statique

Rappels.

- Vocabulaire : spécification = signature = en-tête

1 Distinguer définition et paramètres formels *vs.* appel et paramètres effectifs

```python
def f(n : int) -> int:
    if n % 2 == 0:
        return n
    else:
        return n+1

x = 5
y = f(x)
print(x, y)
```

Le code précédent contient : (au moins un choix)

- la définition de la fonction f (x)
- la définition de paramètre formel x
- la définition du paramètre effectif n
- un appel de f (x)
- un appel de f pour le paramètre effectif 5 (x)
- un appel de f pour le paramètre formel x
- un seul appel de fonction
- deux ou plus de deux appels de fonction (x)
- provoque une erreur à l'exécution
- affiche "x, y"
- affiche "5 5"

- affiche “5 6” (x)
- affiche “5 f(5)”
- n’affiche rien

2 Distinguer spécification (en-tête, signature) : spécifier pour utiliser, pour vérifier *vs.* corps : implémentation du traitement

```python
def f(n : int) -> int:
    if n % 2 == 0:
        return n
    else:
        return n+1

x = 5
y = f(x)
print(x, y)
```

La spécification de la fonction f : (au moins une réponse)

- est complète
- n’est pas complète : il manque le paramètre formel x
- n’est pas complète : il manque le type du paramètre formel
- n’est pas complète : il manque le type du retour
- n’est pas complète : il manque le docstring (x)
- n’est pas complète : il manque la description du rôle de f (x)
- n’est pas satisfaisante : le type du paramètre résultat est faux
- n’est pas satisfaisante : je n’arrive pas à utiliser f sans comprendre son traitement

La définition de la fonction f comporte : (réponse unique)

- 1 ligne (x)
- 2 lignes
- 3 lignes
- 4 lignes
- 5 lignes
- aucune ligne : f n’est pas définie

Le corps de la fonction f comporte : (réponse unique)

- 1 lignes

- 2 lignes
- 3 lignes
- 4 lignes (x)
- 5 lignes
- aucune ligne : il n'est pas défini

Le corps de la fonction f satisfait sa spécification :(réponse unique)

- oui
- non
- je ne sais pas (x)

3 Distinguer appelant *vs.* appelé : le rôle de l'appel, le rôle du return

```python
def f(n : int) -> int:
    if n % 2 == 0:
        return n
    else:
        return n+1

x = 5
y = f(x)
print(x, y)
```

La fonction f : (au moins une réponse)

- n'est pas correcte car il y a 2 `return`

- est correcte car il y a un au moins un `return` pour chaque trace d'exécution possible (x)
- peut effectuer le même traitement avec un seul `return`(x)

L'appel suivant vérifie la spécification de f : (oui/non)

- f(-1)
- f(0)

- f(-1.0) (n)

L'appel suivant déclencher une erreur à l'exécution : (oui/non)

- f(-1) (n)

- f(0) (n)
- f(-1.0) (n)

4 Distinguer variables locales *vs.* variables plus globales, identifier la portée des variables

```python
def f(n : int) -> int:
    if n % 2 == 0:
        return n
    else:
        return n+1

x = 5
y = f(x)
print(n, x, y)
```

Dans le code précédent : (au moins un choix)

- la fonction f n'a aucune variable locale (x)
- la fonction f a une variable locale : n
- la fonction f a deux variables locales : n et n+1
- la fonction f a quatre variables locales : n, 2, 0 et n+1
- son appel modifie la variable globale x
- son appel modifie la variable globale y (x)
- son appel modifie la variable locale n
- provoque une erreur à l'exécution (xx)
- affiche "n, x, y"
- affiche "5 5 5"
- affiche "5 5 6"
- affiche "n 5 f(5)"
- n'affiche rien

```python
def f(n : int) -> int:
    if n % 2 == 0:
        return n
    else:
        return n+1

for i in range(5):
```

```
8       res = f(i)
9       print(res)
```

Le code précédent comporte : (au moins un choix)

- 1 appel de f
- 2 appels de f
- 4 appels de f
- 5 appels de f (x)
- 1 appel de fonctions
- 2 appels de fonctions
- 5 appels de fonctions
- 6 appels de fonctions
- 10 appels de fonctions
- 11 appels de fonctions (x)

```
1  def g(x : float, n : int) -> float:
2      '''g fait son taff, croyez-moi'''
3
4      def h(m : int) -> int:
5          return 2*m + 1
6
7      res = x
8      for i in range(n):
9          res = res + float(h(i))
10     return res
```

L'appel suivant est conforme à la spécification de g : (oui/non)

- g(0, 0) (n)
- g(0.0, 0)
- g(0.0, 1.0) (n)
- g(0.0, f(0))
- g(0.0, h(0)) (n)

`print(g(0.0, 1))`: (choix unique)

- n'est pas une appel de fonction
- est 1 appel de fonction
- est 2 appels de fonctions (x)

L'appel suivant déclenche une erreur à l'exécution : (oui/non) (non partout)

- g(0, 0)
- g(0.0, 0)
- g(0.0, 1.0)
- g(0.0, f(0))
- g(0.0, h(0))

`print(g(0.0, 1)`: (choix unique)

- produit l'affichage : 0
- produit l'affichage : 0.0
- produit l'affichage : 1
- produit l'affichage : 1.0 (x)
- produit l'affichage : 3
- produit l'affichage : 3.0
- déclenche une erreur

L'exécution de `print(g(0.0, 1))` provoque : (choix unique)

- 0 appel de fonctions
- 1 appel de fonctions
- 2 appels de fonctions
- 3 appels de fonctions
- 4 appels de fonctions
- 5 appels de fonction (x)
- 6 appels de fonctions

5.3.1 Sur la mise en place de quizz avec moodle

L'enregistrement de quizz sous forme d'activités *Test* de moodle est une tâche largement reconnue comme étant longue et fastidieuse : trop de clics, de cadres ou d'options à renseigner, de requêtes vers le serveur, … le tout demandant un temps inutilement important, ce qui est vite décourageant, voire inenvisageable et rédhibitoire pour l'enseignant qui souhaiterait les utiliser intensément.

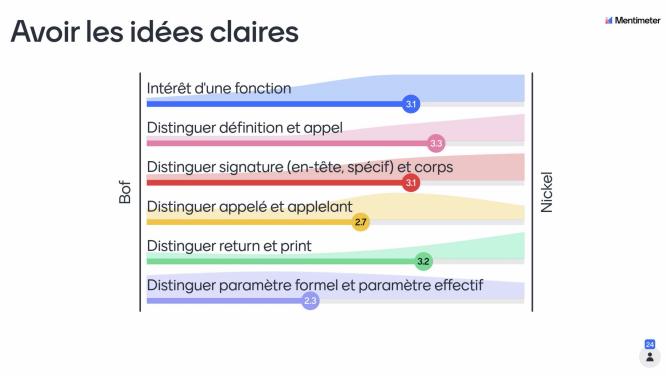
Il faut pouvoir définir le quizz dans *un fichier* que l'on modifie ou complète *dans un éditeur de texte*. Ce fichier est ensuite importé et post-traité automatiquement dans et par moodle (fichier *uploadé* en une fois sous moodle et charge à une "moulinette" ad hoc de mettre à jour les bases de données moodle nécessaires à l'activité Test). Après avoir sensibilisé le service Pl@tinium sur ce besoin, j'ai effectué une recherche des solutions existantes, en particulier de solutions gratuites et open-source.

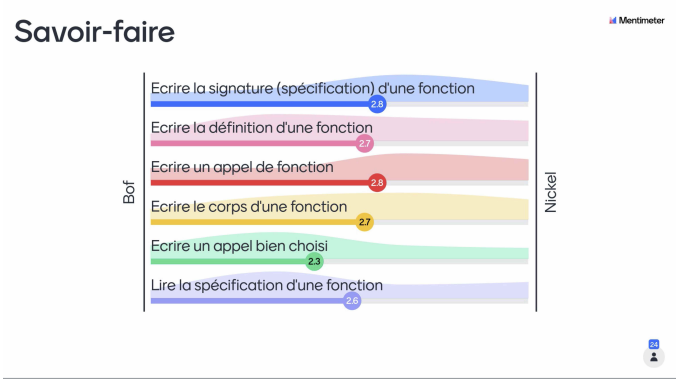
Le terme "GIFT" est le mot-clé utile dans cette démarche. En effet, les tests moodle sont écrits en format GIFT et l'import par moodle de tels fichiers permet le post-traitement automatique ad hoc. Il existe des sites web qui proposent de générer de tels fichiers à partir d'une entrée dans un éditeur intégré à une fenêtre en ligne, comme par exemple text2gift. Il existe aussi des solutions à partir de fichiers excel mais hélas à ce jour uniquement opérationnelles dans l'environnement Windows (et bien sûr assujeties à ces logiciels aux licences payantes). De façon plus spécialisée, citons l'existence de facilités pour générer ces fichiers GIFT à partir de markdown par exemple – ce qui est intéressant pour notre projet car les notebook jupyter travaillent avec le langage de mise en forme markdown. Pour ce qui me concerne, j'écris directement dans mon éditeur de texte préféré selon ce format GIFT qui est rapidement assez simple à manipuler. Le gain de temps est d'environ un facteur 3 à 5 par rapport à la saisie moodle classique.

5.4 Un exemples d'enquête dynamique sur l'acquisition de compétences

Cet exemple de sondage dynamique concerne le chapitre **Fonctions**. Il illustre aussi la grille critériée associée au chapitre et au quizz des annexes précédentes.

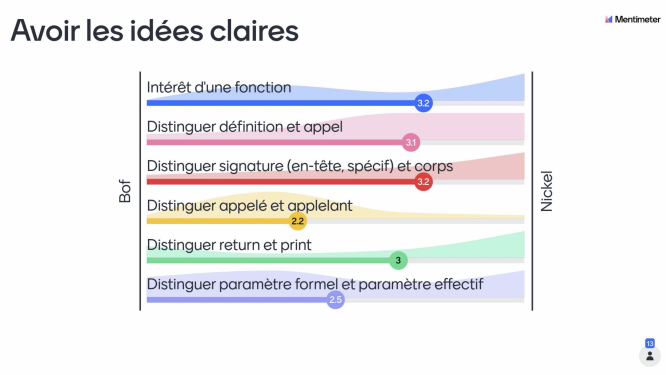
5.4.1 Avant l'activité quizz

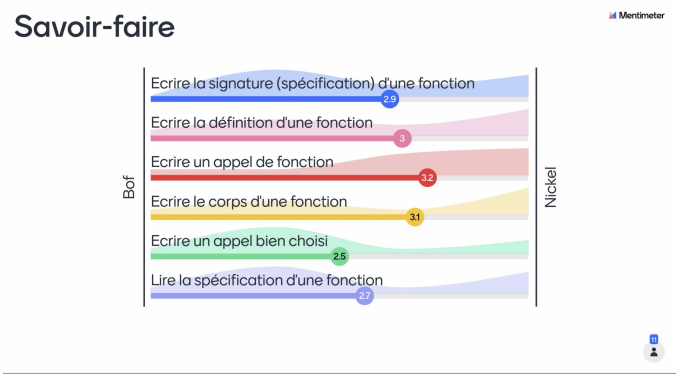




5.4.2 Après l'activité quizz

L'activité quizz de ce chapitre avait été proposée en séance.





Cette enquête permet par exemple de constater pour ce chapitre :

— que certains messages très souvent répétés sont bien passés (exemple des distinctions d'édition/appel, return/print),
— qu'il faut revenir sur la distinction appelée/appelant, à la fois dans le cours et dans le quizz,
— et que la phase d'analyse (au sens de la taxonomie de Bloom) n'est pas encore atteinte pour la majorité des étudiants ("écrire un appel bien choisi" est l'activité la plus abstraite ici) ; ce point devrait s'améliorer dans le sondage asynchrone à effectuer trois à quatre semaines plus tard.

— Et que le quizz rassure globalement les étudiants sur leurs savoir-faire (moyenne globalement meilleures *avant/après le quizz*),
— tout en permettant à l'étudiant de mieux quantifier ses acquis et ainsi mieux se situer dans le choix "Objectif 10" *vs.* "Objectif 20" au moins pour ce chapitre (courbes globales avec des doubles bosses plus nettes *après le quizz*)
— que cette tendance semble plus nette pour les étudiants de type "Objectif 10" (apparition plus importante de "bosses côté gauche" des courbes *après le quizz*).

5.5 Un exemple de TP auto-corrigé formatif-sommatif

5.5.1 Un sujet

Cet exemple est le premier TP auto-corrigé sommatif proposé en mars 2021 pour une épreuve de contrôle continu en temps limité.

Pour chaque question, les séquences d'auto-correction sont définies par les cellules où apparaissent les instructions `assert`.

Des exemples de ces mécanismes d'auto-corrections dans les contextes formatif et sommatif sont présentés dans les sections suivantes.

Cette annexe continue avec 14 pages dont 13 numérotées hors document.

L1 Math-Info UPVD – Semestre 2
Programmation
TP

Bien lire ce qui suit.

ENREGISTRER REGULIEREMENT VOTRE TRAVAIL !

Le sujet contenu dans l'archive zip est composé :

- du notebook `tp-adn.ipynb`,
- de sa version `html` pour plus de facilité de lecture globale,
- les 8 fichiers-texte suivants :
  - adn_1.txt
  - adn_2.txt
  - adn_3.txt
  - adn_4.txt
  - adn_fifi.txt
  - adn_humain.txt
  - gene_bleu.txt
  - gene_marron.txt

A. CONSIGNES TECHNIQUES

Le notebook se manipule classiquement.

1- La plupart des cellules sont dédiées à vos réponses en `python`. Elles se reconnaissent aisément par ce message :

```
# ENTRER VOTRE CODE A LA PLACE DE CES 2 LIGNES
raise NotImplementedError()
```

- La première ligne est interprétée comme un commentaire – que vous pouvez laisser ou supprimer.
- Ainsi **vous écrivez votre réponse à la place de `raise NotImplementedError()`**
- Si vous n'avez pas de réponse à donner dans cette cellule, 2 choix :
  - Si vous souhaitez relancer votre noyau et exécuter des cellules situées après celle-ci, alors effacer la ligne `raise NotImplementedError()` afin d'éviter que cette exception bloque une éventuelle exécution de l'ensemble du notebook après relance du noyau (voir dernière section).

- Sinon ne modifiez pas cette cellule.

2- D'autres cellules sont dédiées à vos réponses textuelles, en `markdown` par exemple. Elles se reconnaissent avec le message explicite :

> VOTRE REPONSE A LA PLACE DE CE TEXTE (format texte ou markdown)

B. VERIFICATIONS A EFFECTUER AU MOINS AVANT LA FIN DE VOTRE TRAVAIL.

Assurez-vous que tout fonctionne comme prévu.

1- **Redémarrer le noyau** : - Dans le menu Noyau (ou Kernel), choisir →Redémarrer & tout exécuter (ou Restart and run all) - Votre notebook doit s'exécuter entièrement comme prévu et sans erreur nouvelle ! - L'exécution d'un notebook avec des cellules sans réponse et non modifiées peut être interrompue par l'exception `NotImplementedError()` (relire le point 2 plus haut).

2- **Déposer votre travail qui est constitué de 1 (*UN SEUL*) fichier** : - Votre fichier `tp-adn.ipynb` complet **sans changer son nom** et **sans l'inclure dans une archive zip ou autre** - Le dépot de ce fichier s'effectue dans le moodle *Programmation Python (S2)* à l'endroit où vous avez téléchargé le sujet.

```
1 #-*- coding: utf8  -*-
```

1 Pour être accompagné par des tests "automatiques"

Dans cette version 2021, des cellules de tests sont proposées pour une vérification "automatique" de vos solutions au fur et à mesure de votre travail. "automatique" est entre guillemets car il faut quand même exécuter la cellule qui le contient (directement ou lors de la relance du noyau avec exécution) !

Si votre développement ne vérifie pas le test, l'exception `AssertionError` est levée. Vous pouvez alors corriger votre solution ou continuer à avancer … à vos risques et périls.

Exemple. Corrigez la cellule suivante pour que l'exception `AssertionError` ne soit plus levée.

```
1 assert 1 + 1 == 3
```

2 Et si on parlait ADN ?

> L'**acide désoxyribonucléique**, ou **ADN**, est une macromolécule biologique présente dans toutes les cellules ainsi que chez de nombreux virus. L'ADN contient toute l'information génétique, appelée génome, permettant le développement, le fonctionnement et la reproduction des êtres vivants.

En biologie, un problème courant consiste à comprendre la structure des molécules d'ADN, et le rôle de structures spécifiques dans le fonctionnement de la molécule. Pour arriver à lire l'ADN, le biologiste utilise un séquenceur d'ADN qui permet de présenter l'ADN sous forme de chaine de caractères. Ainsi, un brin d'ADN est représenté par une suite $c_0 c_1 c_2 \dots c_n$ de lettres choisies parmi les quatre lettres 'A', 'C', 'G' et 'T', qui désignent les nucléotides adenine (A), cytosine (C), guanine (G) et thymine (T). Par exemple, la chaîne de caractères 'AAACAACTTCGTAAGTATA' représente un brin d'ADN. Cependant, il arrive que le séquenceur d'ADN ne parvienne pas à récupérer l'entièreté du brin d'ADN, il place alors un (X) pour marquer un nucléotide inconnu.

En savoir plus sur l'ADN : Wkipédia

2.1 Fonctions usuelles sur les brins d'ADN

Dans le cadre de ce TP, un brin d'ADN est une chaine de caractères qui ne peut contenir que les cinq lettres : * A : adenine * C : cytosine * G : guanine * T : thymine * X : nucléotide inconnu.

On rassemble ces caractères dans le tuple constant suivant – qui n'est pas nécessaire mais qui peut être utile.

```
NUCLEOTIDE = ('A', 'C', 'G', 'T', 'X')
```

2.1.1 est_adn(adn)

Écrire une fonction `est_ADN()` qui vérifie si la chaine de caractères passée en paramètre est un brin d'ADN. Elle retourne `True` si c'est bien un brin d'ADN, et `False` sinon.

```
def est_adn(adn : str) -> bool:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert est_adn("AAACAACTTXXXAAGTXXXA") == True
assert est_adn("Chaine de caractéres") == False
```

2.1.2 cherche_base(adn, nucleotide)

Écrire une fonction `cherche_base()` qui retourne le rang où apparaît pour la première fois le nucléotide dans le brin d'ADN, tous deux passés en paramètres. Si le nucléotide n'apparaît pas dans le brin d'ADN, la fonction devra renvoyer -1.

Attention. Le nucléotide est représenté par une chaine de caractère ne contenant qu'une seule lettre.

```
def cherche_base(adn : str, nucleotide : str) -> int:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert cherche_base("AAACAACTTXXXAAGTXXXA", "C") == 3
assert cherche_base("AACGXXXA", "T") == -1
```

2.1.3 proportion(adn)

Écrire une fonction `proportion()` qui calcule les proportions de présence des nucléotides A, C, G et T dans un brin d'ADN. Pour calculer la proportion d'un nucléotide, compter le nombre d'occurences de ce dernier dans le brin d'ADN et le diviser par le nombre total de nucléotides (nucléotide inconnu 'X' inclus).

```
def proportion(adn : str, opt = True) -> tuple:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert proportion("AAACAACTTXXXAAGTXXXA") == (0.4, 0.1, 0.05, 0.15)
```

2.1.4 complement(adn)

Écrire une fonction `complement()` qui retourne le complémentaire d'un brin d'ADN défini par les transformations suivantes : A devient T, T devient A, C devient G, G devient C.

```
def complement(adn : str) -> str:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert complement("AAACAACTTXXXAAGTXXXA") == "TTTGTTGAAXXXTTCAXXXT"
```

2.2 Traitement sur des parties d'ADN

2.2.1 Rappels et compléments de python

Rappel 1. En python, l'opérateur `in` peut s'appliquer à deux valeurs du type prédéfini `str` et retourne `True` si l'opérande de gauche est inclus dans l'opérande de droite, et `False` sinon.

Cet opérateur s'applique aux conteneurs de type *séquence* : `str`, `list` et `tuple`.

Pour des opérandes `str`, on comprend "inclus" au sens : la chaine de caractères opérande-gauche apparait (sans modification) ou est contenue (sans modification) dans la chaine de caractères opérande-droite.

Exemples :

```
print("to" in "toto")
print("ti" in "toto")
print("tt" in "toto")
```

Rappel 2. En python, on peut définir une tranche (*slice*) de `s` une chaîne de caractères `str` avec la notation `s[i:j]`.

La chaine de caractères `s[i:j]` désigne la chaine *des* caractères de `s` contenus de l'indice `i` à l'indice `j` *non compris* pour `i < j`.

Attention de ne pas utiliser d'indice négatif pour l'instant.

Exemples :

```
s = "abcdefgh"
print("La tranche est :",s[0:3])
print("La tranche est :",s[3:6])
print("La tranche est :",s[3:len(s)])
print("La tranche est :",s[3:3])
print("La tranche est :",s[5:3])
```

2.2.2 Présentation des sous-chaines

Une **sous-chaine** S de longueur s est une partie d'une chaine T de longueur t $\geq$ s. En d'autres termes, la sous-chaine S est contenue (au sens du `in`) dans la chaine T.

Exemple:

```
        est_sous_chaine("ACXXXTCGG","TCG") -> True
        est_sous_chaine("ACXXXTCGG","ACA") -> False
```

On commence par écrire 3 implémentations différentes en python d'une fonction `est_sous_chaine()` dont la spécification est la suivante : > cette fonction prend deux brins ADN en paramètres et retourne `True` si la seconde est une sous-chaine de la première et `False` sinon.

On notera donc successivement ces 3 implémentations différentes : - `est_sous_chaine_0()` - `est_sous_chaine_1()` - `est_sous_chaine_2()`

qui retourneront bien sûr les mêmes résultats.

2.2.3 est_sous_chaine_0(adn1, adn2)

En utilisant l'opérateur `in`, écrire la fonction `est_sous_chaine_0()` qui prend deux brins ADN en paramètres et retourne `True` si la seconde est une **sous-chaine** de la première ou `False` sinon.

```
def est_sous_chaine_0(adn1 : str, adn2 : str) -> bool:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert est_sous_chaine_0("AAACAACTTXXXAAGTXXXA", "AAGT") == True
assert est_sous_chaine_0("AAACAACTTXXXAAGTXXXA", "GTA") == False
assert est_sous_chaine_0("AAACAACTTXXXAAGTXXXA", "XAT") == False

s = "CGAGTAAGTXXXAAGC"
assert est_sous_chaine_0(s,s) == True
assert est_sous_chaine_0(s,'') == True
```

2.2.4 est_sous_chaine_1(adn1, adn2)

Sans utiliser l'opérateur `in` mais en utilisant des tranches de chaînes de caractères, écrire la fonction `est_sous_chaine_1()` qui prend deux brins ADN en paramètres et retourne `True` si la seconde est une **sous-chaine** de la première ou `False` sinon.

```
def est_sous_chaine_1(adn1 : str, adn2 : str) -> bool:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert est_sous_chaine_1("AAACAACTTXXXAAGTXXXA", "AAGT") == True
assert est_sous_chaine_1("AAACAACTTXXXAAGTXXXA", "GTA") == False
assert est_sous_chaine_1("AAACAACTTXXXAAGTXXXA", "XAT") == False

s = "CGAGTAAGTXXXAAGC"
```

```
assert est_sous_chaine_1(s,s) == True
assert est_sous_chaine_1(s,'') == True
```

2.2.5 est_sous_chaine_2(adn1, adn2)

Sans utiliser ni l'opérateur `in`, ni aucune tranche de chaîne de caractères, écrire la fonction `est_sous_chaine_2()` qui prend deux brins ADN en paramètres et retourne `True` si la seconde est une **sous-chaine** de la première ou `False` sinon.

```
def est_sous_chaine_2(adn1 : str, adn2 : str) -> bool:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert est_sous_chaine_2("AAACAACTTXXXAAGTXXXA", "AAGT") == True
assert est_sous_chaine_2("AAACAACTTXXXAAGTXXXA", "GTA") == False
assert est_sous_chaine_2("AAACAACTTXXXAAGTXXXA", "XAT") == False

s = "CGAGTAAGTXXXAAGC"
assert est_sous_chaine_2(s,s) == True
assert est_sous_chaine_2(s,'') == True
```

2.2.6 sous_chaine_commune(adn1, adn2, long)

Écrire une fonction `sous_chaine_commune()` qui prend en paramètres deux brins ADN et une longueur `k` et retourne l'indice dans `adn1` de la première lettre **d'une sous-chaine de longueur `k` et commune** aux brins `adn1` et `adn2`. Cette fonction retourne -1 si aucune sous-chaine commune de longueur `k` n'est trouvée.

Avant de commencer à écrire la fonction, vous décrirez votre algorithme en quelques phrases.

VOTRE REPONSE A LA PLACE DE CE TEXTE (format texte ou markdown)

```
def sous_chaine_commune(adn1 : str, adn2 : str, k : int) -> int:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

2.2.7 Vérification de `sous-chaine_commune()`

ATTENTION. A la différence des fonctions précédentes, le résultat de cette fonction dépend (sauf cas d'absence) de son algorithme. Il peut exister plusieurs sous-chaines communes d'une longueur donnée

(et même de plus grande longueur, si elle est connue). Donc certaines vérifications peuvent-être fausses alors que votre fonction est correcte : l'indice renvoyé par une solution de référence n'est pas celui calculé par votre algorithme bien que celui-ci désigne le premier indice d'une occurrence de longueur `k` commune à `adn1` et `adn2`.

Par exemple, `AXGT` et `TTGA` sont deux sous-chaines de taille `4` communes à `AXGTCTTGA` et `TTGACTGAXGTGA`. Il faut donc faire attention aux exemples que vous choisissez pour vos test unitaires.

Les vérifications suivantes prennent seulement des cas extrèmes, à savoir qu'il n'existe au maximum qu'une seule sous-chaine qui respecte les paramètres.

```
assert sous_chaine_commune("AAACAACTTXXXAAGTXXXA", "CGAGTAAGTXXXAAGC",
    9) == -1
assert sous_chaine_commune("AAACAACTTXXXAAGTXXXA", "CGAGTAAGTXXXAAGC",
    8) == 12
assert sous_chaine_commune("AAACAACTTXXXGTXXXATT", "CGAGTAAGTXXXAAGC",
    7) == -1
assert sous_chaine_commune("AAACAACTTXXXGTXXXATT", "CGAGTAAGTXXXAAGC",
    6) == 12
assert sous_chaine_commune("AAACAACTTGTATTAAACAACTTGTATT", "
    TXXXTXXXTXXXTXXX", 2) == -1
assert sous_chaine_commune("AAACAACTTGTATTAAACAACTTGTATT", "
    XXXXXXXXXXXX", 1) == -1

s = "CGAGTAAGTXXXAAGC"
assert sous_chaine_commune(s,s, len(s)) == 0
assert sous_chaine_commune(s,'', 0) == 0
```

2.2.8 plus_longue_sous_chaine_commune(adn1, adn2)

Écrire une fonction `plus_longue_sous_chaine_commune()` qui identifie et retourne **une plus longue** sous-chaine commune à deux brins ADN passés en paramètres.

Avant de commencer à écrire la fonction, vous décrirez votre algorithme en quelques phrases.

VOTRE REPONSE A LA PLACE DE CE TEXTE (format texte ou markdown)

```
def plus_longue_sous_chaine_commune(adn1 : str, adn2 : str) -> str:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert plus_longue_sous_chaine_commune("AAACAACTTXXXAAGTXXXA", "
    CGAGTAAGTXXXAAGC") == "AAGTXXXA"
assert plus_longue_sous_chaine_commune("AAACAACTTXXXGTXXXATT", "
    CGAGTAAGTXXXAAGC") == "GTXXXA"
assert plus_longue_sous_chaine_commune("AAACAACTTGTATTAAACAACTTGTATT",
    "TXXXTXXXTXXXTXXX") == "T"
assert plus_longue_sous_chaine_commune("AAACAACTTGTATTAAACAACTTGTATT",
    "XXXXXXXXXXXX") == ""

s = "CGAGTAAGTXXXAAGC"
assert plus_longue_sous_chaine_commune(s,s) == s
assert plus_longue_sous_chaine_commune(s,'') == ''
```

2.3 Analyse de l'ADN humaine

Votre séquenceur d'ADN a décomposé l'ADN humaine et a stocké tout ceci dans le fichier `adn_humain.txt`.

2.3.1 extraire_donnees()

Écrire une fonction `extraire_donnees()` qui récupère l'ADN d'un fichier texte et retourne son contenu comme une chaine de caractères.

Cette fonction prendra soin de vérifier que l'extraction retourne bien un brin d'ADN.

```
def extraire_donnees(nom_fichier : str) -> str:
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert len(extraire_donnees("adn_humain.txt")) == 5000
```

2.3.2 Couleurs des Yeux

La couleurs des yeux est déterminée par un gène qui est présent dans l'ADN. Ainsi le brin d'ADN associé à ces gènes sont définis dans les fichiers `gene_bleu.txt` et `gene_marron.txt` pour les couleurs bleu et marron.

Vérifier que la personne dont l'ADN est enregistrée dans `adn_humain.txt` possède bien le gène des yeux bleus et non celui des yeux marrons.

```
# ENTRER VOTRE CODE ICI
```

2.3.3 Mesure de la performance du traitement

Préalable La fonction `perf_counter()` du module `time` (définie depuis python3.3 et disponible avec l'exécution de la cellule suivante) permet une mesure fiable, précise et simple à mettre en oeuvre du temps d'exécution d'une séquence de traitements.

```
import time
```

On procède en deux appels qui encadrent la portion de code à mesurer.

- `t0 = time.perf_counter()`: le premier appel initialise une valeur initiale `t0` juste avant la portion de code à mesurer
- `t1 = time.perf_counter()`: le second appel mesure `t1` à l'issue de l'exécution de la portion de code
- `t1 - t0` est la mesure de ce temps d'exécution

Exemple. Sur ma machine, l'exécution de la cellule suivante donne les résultats :

```
temps d'exécution de l'appel à la fonction PLSC: 5.670e-04 sec.
temps d'exécution de l'appel à la fonction PLSC: 5.702e-04 sec.
temps d'exécution de l'appel à la fonction PLSC: 1.005e-03 sec.
temps d'exécution de l'appel à la fonction PLSC: 9.898e-04 sec.
temps d'exécution de l'appel à la fonction PLSC: 1.369e-03 sec.
```

On voit ainsi l'intérêt de répéter ces mesures et d'en dégager des valeurs plus significatives, comme la moyenne arithmétique par exemple.

Remarque (qui fait l'objet de compléments dans un chapitre de cours à venir). Il peut même être nécessaire de *boucler autour* de la portion à analyser et de *mesurer à l'extérieur* de cette boucle et d'effectuer les traitements statistiques adaptés à ces échantillons de mesures.

```
for nb in range(5):
    t0 = time.perf_counter() # début de la mesure
    res = plus_longue_sous_chaine_commune("AAACAACTTXXXAAGTXXXA", "
        CGAGTAAGTXXXAAGC")
    t1 = time.perf_counter() # fin de la mesure
    delta = t1 - t0
    print("Temps d'exécution de l'appel à la fonction PLSC:", f"{delta
        :1.3e} sec.")
```

Remarque. Le format d'affichage de la mesure `delta` à l'aide de la construction `"f{valeur:format } unité"` permet d'afficher `valeur` en notation scientifique (`e`) avec respectivement 1 et 3 chiffres avant et après la virgule – ici cette mise en forme (formattage) est aussi complétée par l'`unité` de `valeur`.

Les différentes versions de `est_sous_chaine()` On va mesurer la performance en temps d'exécution de nos trois versions de la fonction `est_sous_chaine()`. Cette expérience nous permettra de choisir celle de ces implantations qu'il est le plus intéressant à utiliser dans les fonctions `sous_chaine_commune()` et `plus_longue_sous_chaine_commune()`. Il n'est pas interdit d'aller modifier ces fonctions après avoir analysé les résultats de cette expérimentation.

On considère les deux traitements suivants : - (T1) : vérifier la présence du gène des yeux bleus dans l'ADN enregistrée dans `adn_humain.txt` - (T2) : vérifier l'absence du gêne des yeux marrons dans l'ADN enregistrée dans `adn_humain.txt`

Pour *chacun* de ces traitements : 1. effectuer 5 mesures de leur temps d'exécution 2. calculer la moyenne arithmétique de ces 5 mesures 3. afficher cette moyenne ainsi que la longueur des chaînes concernées par ce traitement.

Remarque. (T2) est un traitement plus coûteux que (T1).

```
1  # ENTRER VOTRE CODE ICI
```

(T1) avec `plus_longue_sous_chaine_commune()` On effectue maintenant des tests similaires avec cette fonction plus coûteuse. D'abord le traitement (T1) qui vérifie une présence.

```
1  # ENTRER VOTRE CODE ICI
```

(T2) avec `plus_longue_sous_chaine` : influence de la longueur de la chaine Pour (T2), le gêne des yeux marrons est absent de l'ADN enregistrée dans `adn_humain.txt`. Mais il y a certainement au moins une sous-chaine commune à ces deux séquences. Quelle est la longueur maximale d'une telle sous-chaine ? Elle est bien sûr strictement inférieure à celle du gêne des yeux marrons.

Répondre à cette question avec notre implémentation de `plus_longue_sous_chaine()` est en fait un traitement *coûteux* pour le temps imparti aujourd'hui. On va s'en convaincre en observant le temps mis pour identifier cette plus longue sous-chaine commune *en se limitant aux n premiers caractères* de la séquence de l'ADN humain et au gêne de yeux marrons, et ce en faisant varier cette longueur n de façon adaptée à notre *propre* environnement.

Indications. - N'effectuez que 2 mesures pour une taille n fixée - Commencez par n qui varie de $n_{min} = 200$ à $n_{max} = 500$ par pas de 100 - Augmentez n_{max} de façon progressive et adaptée en observant les mesures obtenues.

Attention. Les calculs peuvent durer une ou deux minutes, pendant ce temps vous pouvez commencer à travailler sur la question suivante, relire votre notebook, ou simplement aller chercher un café. Si toutefois cela dure trop longtemps, annulez l'exécution avec `Kernel`, puis `Interrupt` et passez à la question suivante.

```
# ENTRER VOTRE CODE ICI
```

Question. Qu'en déduire ?

VOTRE REPONSE A LA PLACE DE CE TEXTE (format texte ou markdown)

2.3.4 Identification ADN

Gérard travaille à la police scientifique mais il est un peu tête en l'air. Il a mélangé 4 fichiers d'ADN, dont celui de **Riri**. Trouvez quel est le fichier correspondant à **Riri** sachant que l'ADN de son père **Fifi** est enregistré dans le fichier `adn_fifi.txt`. Les fichiers sont nommés `adn_1.txt`, `adn_2.txt`, `adn_3.txt` et `adn_4.txt`.

Note. Les enfants ont une grande similitude ADN avec leurs parents.

Nous voulons aider Gérard mais au vu des piètres performances de notre fonction `plus_longue_sous_chaine_commu` , son supérieur se rendra compte de l'erreur de Gérard.

Pour cela, nous allons écrire une nouvelle fonction plus performante.

plus_longue_sous_chaine_commune_performante(adn1, adn2) Pour augmenter les performances de votre fonction, vous avez plusieurs possibilités : - Réduire le nombre de boucles ainsi que leurs tailles : ceci réduit la complexité algorithmique du traitement. - Réduire le nombre d'appel aux fonctions `est_sous_chaine` et `sous_chaine_commune` : ceci réduit le temps "mort" du processeur (allocation de mémoire, copie de données, etc.)

```
def plus_longue_sous_chaine_commune_performante(adn1,adn2):
    # ENTRER VOTRE CODE ICI
    raise NotImplementedError()
```

```
assert plus_longue_sous_chaine_commune("AAACAACTTXXXAAGTXXXA", "
    CGAGTAAGTXXXAAGC") == "AAGTXXXA"
assert plus_longue_sous_chaine_commune("AAACAACTTXXXGTXXXATT", "
    CGAGTAAGTXXXAAGC") == "GTXXXA"
```

```
assert plus_longue_sous_chaine_commune("AAACAACTTGTATTAAACAACTTGTATT",
    "TXXXTXXXTXXXTXXX") == "T"
assert plus_longue_sous_chaine_commune("AAACAACTTGTATTAAACAACTTGTATT",
    "XXXXXXXXXXXX") == ""

s = "CGAGTAAGTXXXAAGC"
assert plus_longue_sous_chaine_commune(s,s) == s
assert plus_longue_sous_chaine_commune(s,'') == ''
```

Identification de Riri Maintenant que nous avons une fonction plus performante, nous allons pouvoir aider Gérard à trouver le fichier ADN de Riri parmi les quatre fichiers : `adn_1.txt`, `adn_2.txt`, `adn_3.txt` et `adn_4.txt`.

```
# ENTRER VOTRE CODE ICI
```

Question. Que dire à Gérard ?

VOTRE REPONSE A LA PLACE DE CE TEXTE (format texte ou markdown)

5.5.2 Un exemple de faute signalée dans le contexte formatif

Dans ce contexte, l'erreur détectée est accompagnée d'indications sur son origine et en quoi la solution proposée diffère de la solution attendue. L'étudiant est guidé pour localiser et corriger cette erreur. Sa nouvelle solution sera évaluée de la même façon et sans limitation.

2.1.2 cherche_base(adn, nucleotide)

Écrire une fonction `cherche_base()` qui retourne le rang où apparaît pour la première fois le nucléotide dans le brin d'ADN, tous deux passés en paramètres. Si le nucléotide n'apparaît pas dans le brin d'ADN, la fonction devra renvoyer -1.

Attention. Le nucléotide est représenté par une chaine de caractère ne contenant qu'une seule lettre.

[20]:
```
1  def cherche_base(adn : str, nucleotide : str) -> int:
2      '''retourne le rang de la première occurrence de nucleotide dans adn.
3      Retourne -1 si absent'''
4      res = -1
5      for i in range(len(adn)):
6          if adn[i] == nucleotide:
7              res = i
8      return res
```

[21]:
```
1  assert_equal(cherche_base("AACGXXXA", "T"), -1)
2  assert_equal(cherche_base("AAACAACTTXXXAAGTXXXA", "C"), 3)
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-21-d96b4a1c29ad> in <module>
      1 assert_equal(cherche_base("AACGXXXA", "T"), -1)
----> 2 assert_equal(cherche_base("AAACAACTTXXXAAGTXXXA", "C"), 3)

/usr/local/Cellar/python/3.7.7/Frameworks/Python.framework/Versions/3.7/lib/python3.7/
unittest/case.py in assertEqual(self, first, second, msg)
    850         """
    851         assertion_func = self._getAssertEqualityFunc(first, second)
--> 852         assertion_func(first, second, msg=msg)
    853
    854     def assertNotEqual(self, first, second, msg=None):

/usr/local/Cellar/python/3.7.7/Frameworks/Python.framework/Versions/3.7/lib/python3.7/
unittest/case.py in _baseAssertEqual(self, first, second, msg)
    843             standardMsg = '%s != %s' % _common_shorten_repr(first, second)
    844             msg = self._formatMessage(msg, standardMsg)
--> 845             raise self.failureException(msg)
    846
    847     def assertEqual(self, first, second, msg=None):

AssertionError: 6 != 3
```

Ici par exemple, la dernière ligne de l'auto-correction indique à l'étudiant que sa réponse n'est pas correcte car elle calcule la valeur 6 alors que la valeur 3 est attendue. Ce traitement incorrect apparaît dans le second test (ligne 2 indiquée en haut de l'auto-correction). Cependant, le premier test (ligne 1) valide une partie du traitement proposé par l'étudiant (que celui-ci peut identifier comme étant) lorsque la condition de la ligne 6 n'est pas vérifiée. L'erreur provient donc du traitement effectué lorsque cette condition est vérifiée et se situe donc à la ligne 7 de la solution incorrecte proposée. En

effet, la ligne 7 qui modifie `i` tant que cette condition est vérifiée détermine la *dernière* occurrence de `nucleotide` cherché dans la séquence `adn` et non la *première* occurrence demandée.

5.5.3 Un exemple de faute signalée dans le contexte sommatif

Dans ce contexte, l'erreur détectée est signalée sans indication explicite de la solution attendue. L'étudiant sait que sa solution est inexacte et dans un contexte d'épreuve en temps limité, il peut choisir ou non de la corriger.

2.1.2 cherche_base(adn, nucleotide)

Écrire une fonction `cherche_base()` qui retourne le rang où apparaît pour la première fois le nucléotide dans le brin d'ADN, tous deux passés en paramètres. Si le nucléotide n'apparaît pas dans le brin d'ADN, la fonction devra renvoyer -1.

Attention. Le nucléotide est représenté par une chaine de caractère ne contenant qu'une seule lettre.

[18]:
```
1 def cherche_base(adn : str, nucleotide : str) -> int:
2     '''retourne le rang de la première occurrence de nucleotide dans adn.
3     Retourne -1 si absent'''
4     res = -1
5     for i in range(len(adn)):
6         if adn[i] == nucleotide:
7             res = i
8     return res
```

[19]:
```
1 assert cherche_base("AACGXXXA", "T") == -1
2 assert cherche_base("AAACAACTTXXXAAGTXXXA", "C") == 3
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-19-4bf483bf9e45> in <module>
      1 assert cherche_base("AACGXXXA", "T") == -1
----> 2 assert cherche_base("AAACAACTTXXXAAGTXXXA", "C") == 3

AssertionError:
```

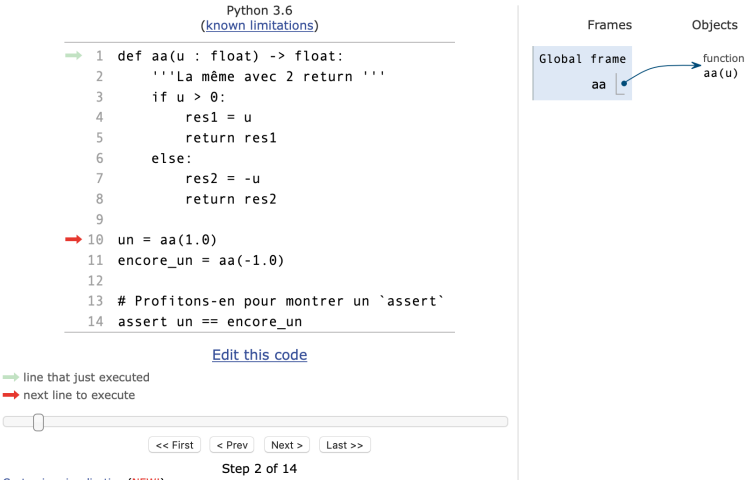
Ici par exemple, l'auto-correction indique à l'étudiant que sa réponse n'est pas correcte. Il sait uniquement que sa solution passe avec succès le premier test. Il n'a aucune autre information sur ce qui provoque la non validation de sa solution par ce second test, ni même si le succès du premier test n'est pas fortuit.

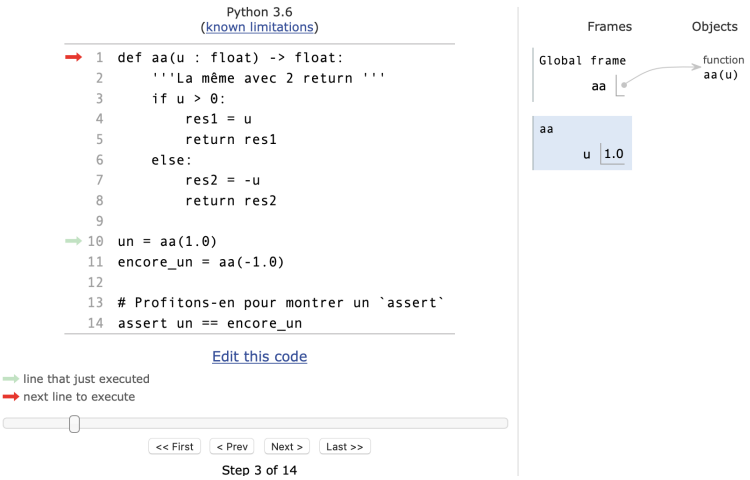
5.6 Un exemple de `pythontutor`

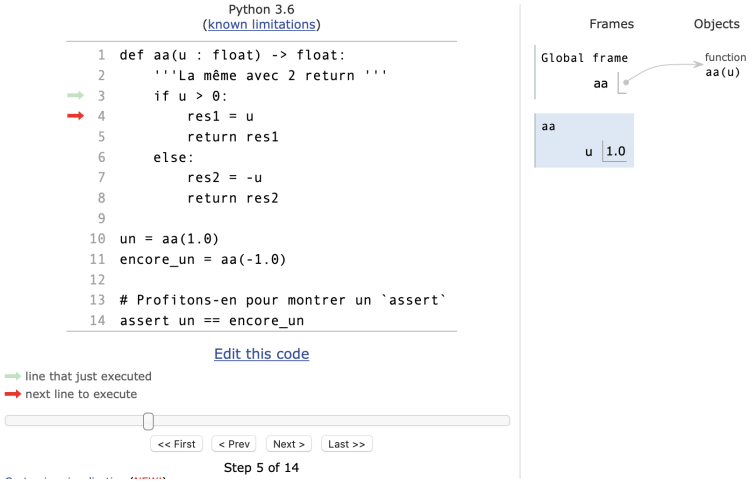
Cet exemple est extrait du chapitre **Fonctions**.

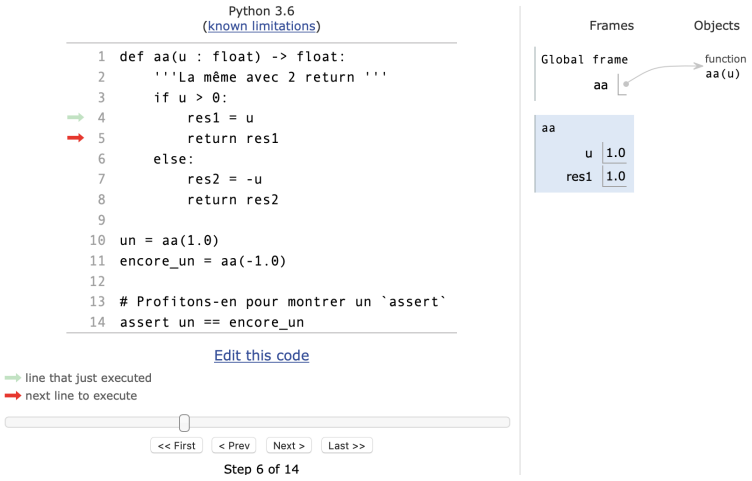
Nous présentons 6 des 14 étapes de cet exemple qui permet de visualiser les distinctions déjà explicitées dans les compétences de ce chapitre (Annexe 2), et plus particulièrement le rôle de 2 **`return`** dans une fonction, leurs effets respectifs lors de l'appel et les changements de contexte associés.

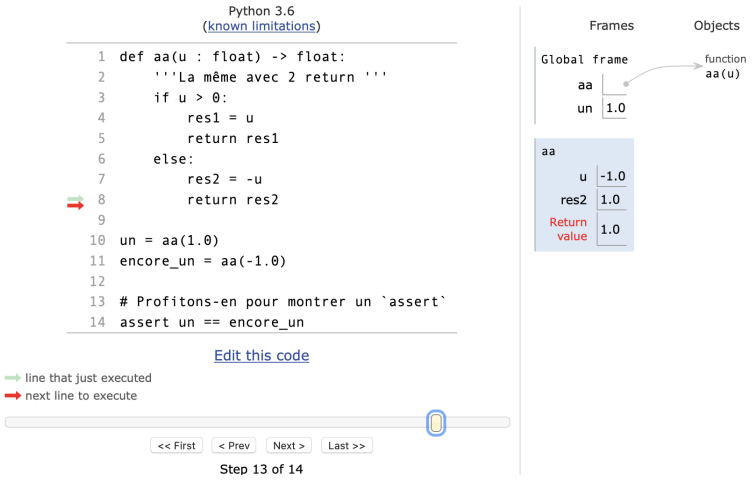
Le code est défini dans la fenêtre de gauche. Son exécution est réalisée pas à pas : "avancée séquentielle" des flèches rouge et verte. Cette exécution génère 14 changements d'état visualisables pas à pas dans la fenêtre de droite.











La version en ligne complète de cet exemple est accessible avec ce lien

5.7 Le sondage de “type EEE” pour les L2 et L3

Cette annexe continue avec 8 pages non numérotées.

Évaluation des enseignements

Page 1

1 * Après la L1, j'ai poursuivi ma formation en :

- ○ en L2 informatique
- ○ en L3 informatique
- ○ en L2 mathématiques
- ○ en L3 mathématiques

2 * Ces UE d'algorithmique et de programmation (python) proposaient deux parcours (Objectif 10 ou 20) selon les objectifs d'apprentissage visés. J'ai suivi le parcours :

- ○ objectif 20 en entier
- ○ plutôt objectif 20
- ○ plutôt objectif 10
- ○ objectif 10 en entier

ORGANISATION DE L'ENSEIGNEMENT

Dans cette série de questions, le vocable "*cours*" regroupe l'ensemble des formes d'enseignement : cours en amphi, TD, TP sur machine, sujets, devoirs, contrôles de connaissances, ... D'autres séries de questions sont spécifiquement consacrées aux cours en amphi, aux TD, aux TP sur machine, ...

3 * Les objectifs du cours sont clairs

- ○ oui
- ○ plutôt oui
- ○ plutôt non
- ○ non
- ○ sans avis

4 * Le cours est structuré en cohérence avec les objectifs

- ○ oui
- ○ plutôt oui
- ○ plutôt non
- ○ non
- ○ sans avis

5 * Les notions importantes sont clairement identifiées et suffisamment abordées pour être assimilées

- ○ oui
- ○ plutôt oui
- ○ plutôt non
- ○ non
- ○ sans avis

6 * Les supports de cours disponibles sous Moodle facilitent la compréhension de ces notions

- ○ oui
- ○ plutôt oui
- ○ plutôt non
- ○ non
- ○ sans avis

7 * La charge de travail est adéquate par rapport au nombre de crédits ECTS accordés à ce cours (1 crédit ECTS = 25-30 heures de travail, y compris la présence en classe et la préparation des examens)

oui plutôt oui plutôt non non sans avis

8 * Les modalités d'évaluation ont été clairement expliquées (type d'évaluation, durée, documentation autorisée, etc.)

oui plutôt oui plutôt non non sans avis

BILAN GLOBAL PERSONNEL

Dans cette série de questions, le vocable "*cours*" regroupe encore l'ensemble des formes d'enseignement (cours en amphi, TD, TP sur machine, sujets, devoirs, contrôles de connaissances, ...). D'autres séries de questions sont spécifiquement consacrées aux cours en amphi, aux TD, aux TP sur machine, ...

9 * Le cours est adapté à mes connaissances préalables

oui plutôt oui plutôt non non sans avis

10 * Ma réflexion est stimulée, le cours m'amène à réfléchir

oui plutôt oui plutôt non non sans avis

11 * Le cours est adapté à mon projet de formation, à mes objectifs d'apprentissage

oui plutôt oui plutôt non non sans avis

12 * Je réalise des apprentissages significatifs dans ce cours (méthodes nouvelles, connaissances, savoir-faire ...)

oui plutôt oui plutôt non non sans avis

13 * Ce cours permet d'améliorer mes techniques et mes méthodes d'apprentissage

oui plutôt oui plutôt non non sans avis

14 * Avec un peu de recul (au semestre suivant au moins), la proportion de ce cours que j'ai effectivement assimilé une fois le contrôle de fin de semestre passé, est de :

plus de 90% entre 66% et 90% entre 50% et 65%
entre 33% et 49% moins de 32%

15 * Les notions importantes d'**algorithmique** m'ont été utiles dans la suite de ma formation, en particulier dans les UE suivantes suivies ce semestre :

Unités du processeur Structures et algorithmes
Programmation avancée en C Robots mobiles Dans aucune UE

16 Les notions importantes de **programmation** m'ont été utiles dans la suite de ma formation, en particulier dans les UE suivantes suivies ce semestre :

- ○ Unités du processeur
- ○ Structures et algorithmes
- ○ Programmation avancée en C
- ○ Robots mobiles
- ○ Dans aucune UE

17 Quels sont les points forts de ce cours?

18 Quels sont les aspects de ce cours à améliorer ?

19 Remarques et suggestions complémentaires :

Page 3

LE COURS

Dans cette série de questions, le vocable "*cours*" désigne le cours en amphi (promotion complète). Les TD et TP font l'objet d'autres séries de questions.

20 * Le cours est présenté de façon stimulante

oui plutôt oui plutôt non non sans avis

21 * Il y a suffisamment d'illustrations : exemples, applications, expériences, etc.

oui plutôt oui plutôt non non sans avis

22 * L'utilisation des notebooks jupyter aide à mieux comprendre les notions présentées

oui plutôt oui plutôt non non sans avis

23 * Le rythme du cours est adapté

oui plutôt oui plutôt non non sans avis

24 * L'expression orale de l'enseignant-e est bonne (élocution, clarté, volume de voix)

oui plutôt oui plutôt non non sans avis

25 * L'enseignant-e suscite de l'intérêt pour son enseignement

oui plutôt oui plutôt non non sans avis

26 * L'enseignant incite les étudiants à être actifs

oui plutôt oui plutôt non non sans avis

27 * L'enseignant répond aux questions des étudiants

oui plutôt oui plutôt non non sans avis

28 * Les retours de l'enseignant me sont utiles

oui plutôt oui plutôt non non sans avis

29 * Il est possible de dialoguer avec l'enseignant durant le cours

oui plutôt oui plutôt non non sans avis

30 * L'enseignant est disponible pour répondre aux questions en dehors du cours

oui plutôt oui plutôt non non sans avis

31 * L'enseignant adapte ses méthodes pédagogiques pour aider les étudiants

oui plutôt oui plutôt non non sans avis

32 Remarques et suggestions complémentaires :

Page 4

LES TD

Cette série de questions concerne exclusivement les séances de TD.

33 * Le rythme des TD est adapté à la compréhension des notions importantes

Oui Plutôt oui Plutôt non Non Sans avis

34 * Le contenu des TD est adapté à la compréhension des notions importantes

oui plutôt oui plutôt non non sans avis

35 * La difficulté des exercices de TD est adaptée à mon projet de formation, à mes objectifs d'apprentissage

Oui Plutôt oui Plutôt non Non Sans avis

36 * Les TD préparent aux épreuves du contrôle des connaissances

Oui Plutôt oui Plutôt non Non Sans avis

37 * Les sujets des contrôles de connaissances des années précédentes sont utiles

oui plutôt oui plutôt non non sans avis

38 * Travailler en TD de façon individuelle ou en tout petit groupe est adapté à votre projet de formation

Oui Plutôt oui Plutôt non Non Sans avis

39 * L'enseignant incite les étudiants à être actifs

oui plutôt oui plutôt non non sans avis

40 * L'enseignant répond aux questions des étudiants

oui plutôt oui plutôt non non sans avis

41 * Les retours de l'enseignant me sont utiles

oui plutôt oui plutôt non non sans avis

42 * L'expression orale de l'enseignant-e est bonne (élocution, clarté, volume de voix)

oui plutôt oui plutôt non non sans avis

43 Remarques et suggestions complémentaires :

LES TP

Cette série de questions concerne exclusivement les séances de travaux pratiques (TP sur machine).

44 * Les TP sont en nombre suffisant et suffisamment variés pour votre objectif d'apprentissage

Oui | Plutôt oui | Plutôt non | Non | Sans avis

45 * Le rythme des TP est adapté à la compréhension approfondie des notions importantes

Oui | Plutôt oui | Plutôt non | Non | Sans avis

46 * Le contenu des TP est adapté à la compréhension approfondie des notions importantes

oui | plutôt oui | plutôt non | non | sans avis

47 * Les sujets de TP et des années précédentes sont adaptés à mon projet de formation, à mes objectifs d'apprentissage

oui | plutôt oui | plutôt non | non | sans avis

48 * La difficulté des TP et le temps à y consacrer sont adaptés à mon projet de formation

Oui | Plutôt oui | Plutôt non | Non | Sans avis

49 * Les TP notés, à réaliser individuellement et en temps limité, sont adaptés à mon projet de formation

Oui | Plutôt oui | Plutôt non | Non | Sans avis

50 * L'enseignant incite les étudiants à être actifs

oui | plutôt oui | plutôt non | non | sans avis

51 * L'expression orale de l'enseignant-e est bonne (élocution, clarté, volume de voix)

oui | plutôt oui | plutôt non | non | sans avis

52 * Les retours de l'enseignant me sont utiles

oui | plutôt oui | plutôt non | non | sans avis

53 * L'enseignant répond aux questions des étudiants

oui plutôt oui plutôt non non sans avis

54 Remarques et suggestions complémentaires :

Fermer cette fenêtre

5.8 Etat des lieux des ressources NSI

Cette annexe fonctionnelle regroupe les liens vers les ressources utiles en amont de cette UE, et en particulier dans le cadre de la nouvelle spécialité NSI du lycée. Une bibliographie assez exhaustive à ce jour est aussi présentée.

De façon préalable, mentionnons que je n'ai hélas pu identifier sur la plateforme FUN-MOOC aucune ressource académique adaptée à cette UE. Le mot-clé "algorithmique" apparait (sans surprise) dans 29 MOOC dont la plupart sont très loins des objectifs pédagogiques de cette UE. Une seule exception : le très complet MOOC "Socle en informatique" du CNAM qui s'appuie sur le choix du langage de programmation C – et ce trop fortement par rapport à l'approche suivie dans notre UE.

5.8.1 NSI au Lycée

- Première : 4h x 30 semaines (120h) dont 1h au moins pour projet en groupe
- Terminale : 6h x 30 semaines (180h)

5.8.2 Le portail institutionnel Eduscol

Le portail institutionnel eduscol est organisé sous forme de sites disciplinaires qui proposent une vision d'ensemble de la discipline, des actualités spécifiques et des présentations de ressources pédagogiques et éducatives.

Hélas, il n'existe pas (à ce jour) de site disciplinaire NSI – comme c'est le cas pour les spécialités mathématiques, technologie, STI …

5.8.3 Sites NSI à destination des lycées de la spécialité NSI

Parmi les nombreux liens identifiés par ce site, les suivants ont plus particulièrement retenu mon attention – sans chercher à l'exhaustivité.

MonLycéeNumérique propose des leçons très bien faites mais avec une interface moins agréable que mathweb. Lycée Blaise Pascal est très bien fait et complet.

On mentionne aussi le site de M. Bernon qui rassemble des QCMs interactifs issus de la banque de données des évaluations communes du baccalauréat – anciennement "épreuves communes de contrôle continu" (E3C).

5.8.4 Le site France IOI

Ce site déjà mentionné pour la préparation et l'organisation de concours d'algorithmique et de programmation s'est récemment enrichi d'une zone enseigner. Il est ainsi possible de créer un groupe d'apprenants dont on suit la progression. Les rappels de cours sont très succincts (ce qui n'est pas surprenant vu l'objectif général de ce site à vocation élitiste) *mais* les corrections des exercices proposés sont très intéressantes.

Dans les perspectives de ce projet, on pourrait envisager de définir une progression et une utilisation adaptée de ces ressources pour des compléments de formation à destination des étudiants les plus intéressés par l'algorithmique et la programmationinformatique.

5.8.5 Bibliographie pour NSI 2020

Les ressources bibliographiques sont assez nombreuses en regard de la très récente mise en place de la spécialité NSI. Il est intéressant de noter que certaines d'entre elles sont des références **très pertinentes pour nos étudiants de L1** en complément des supports de cette UE – et d'autres UE des 3 premiers semestres de la Licence.

Mentionnons par exemple que les deux premières références ci-après comportent respectivement 525 et 515 pages ! Malgré le volume horaire de la spécialité NSI en première et terminale, un nombre extrêmement réduit de lycéens doit pouvoir profiter du niveau visé par ces ouvrages.

Ellipses Titre : Spécialité Numérique et sciences informatiques : 30 leçons avec exercices corrigés - Première - Nouveaux programmes Auteur(s) : Balabonski Thibaut, Conchon Sylvain, Filliâtre Jean-Christophe, Nguyen Kim Editeur : Ellipses Collection Compétences Attendues ISBN : 9782340033641

Titre : Spécialité Numérique et sciences informatiques : 24 leçons avec exercices corrigés - Terminale - Nouveaux programmes Auteur(s) : Balabonski Thibaut, Conchon Sylvain, Filliâtre Jean-Christophe, Nguyen Kim Editeur : Ellipses Collection Compétences Attendues ISBN : 9782340038554

Titre : Spécialité Numérique et sciences informatiques - Première - nouveaux programmes Auteur(s) : Serge Bays (Auteur) Bertrand Hauchecorne (Direction) Editeur : Ellipses Collection Prépas Sciences Paru le 9 juillet 2019 Scolaire / Universitaire (broché) ISBN : 2340031729

Titre : Spécialité Numérique et sciences informatiques - Terminale - nouveaux programmes Auteur(s) : Serge Bays (Auteur) Bertrand Hauchecorne (Direction) Editeur : Ellipses Collection Prépas Sciences Paru le 28 juillet 2020 Scolaire / Universitaire (broché) ISBN : 2340038448

Spécialité Numérique et sciences informatiques - Première - nouveaux programmes Cécile Canu (Auteur) Editeur : Ellipses Collection Compétences Attendues Paru le 7 août 2019 Scolaire / Universitaire (broché) ISBN : 2340031788

Titre : Spécialité NSI - Numérique et sciences informatiques - Terminale - nouveaux programmes Auteur(s) : Jean-Christophe Bonnefoy (Auteur) Bertrand Petit (Auteur) Editeur : Ellipses Collection Compétences Attendues Paru le 12 mai 2020 Scolaire / Universitaire (broché) ISBN : 2340038154

Titre : Spécialité NSI (numérique et sciences informatiques) - Première - nouveaux programmes Auteur(s) : Legrand David Editeur : Ellipses Collection Parcours et méthodes Paru le 12.05.2020 ISBN : 9782340038578

Hatier Titre : NSI 1ère générale (spécialité) - Prépabac Cours & entraînement. Nouveau programme, nouveau bac (2020-2021) ‘ Auteur(s) : Céline Adobet (Auteur) Guillaume Connan (Auteur) Gérard Rozsavolgyi (Auteur) Laurent Signac (Auteur) Nouveau programme de Première (2020-2021) Editeur : Hatier Collection : Prepabac Entrainement Progress Paru le 11 septembre 2019 ISBN : 2401052305 Version e-book : 8,99€

Titre : NSI Tle générale (spécialité) - Prépabac Cours & entraînement. Nouveau programme, nouveau bac (2020-2021) ‘ Auteur(s) : Guillaume Connan (Auteur) Vojislav Petrov (Auteur) Gérard Rozsavolgyi (Auteur) Laurent Signac (Auteur) Editeur : Hatier Collection : Prepabac Entrainement Progress Paru le 16 septembre 2020 ISBN : 2401064613 Version e-book : 8,99€

Nathan Titre : Interros des Lycées Numérique Sciences Informatiques - Première Auteur(s) : Stéphane Pasquet (Auteur) Mikael Leopoldoff
Editeur : Nathan Collection: Interros des Lycées Paru le 4 juillet 2019 Scolaire / Universitaire (broché) ISBN : 2091574651

Titre : Interros des Lycées Numérique Sciences Informatiques - Terminale Auteur(s) : Stéphane Pasquet (Auteur) Mikael Leopoldoff Editeur : Nathan Collection: Interros des Lycées Paru le 9 juillet 2020 Scolaire / Universitaire (broché) ISBN : 2091575437

5.9 Projet initial : description de février 2020

Cette annexe continue avec 10 pages numérotées hors document.

Projet pédagogique - CPP 2020

Ph. Langlois

14 février 2020

Table des matières

1 Projet pédagogique

1.1 Contexte

Conception, développement et mise en oeuvre de contenus et d'outils logiciels pour un meilleur apprentissage de l'algorithmique et de la programmation.

1.2 Public visé

Tous les étudiants des UE suivantes.

Formation : Licence Informatique, Licence Mathématique
Niveau : L1, semestre S2
UE (mouture 2021-2025) : Algorithmique et programmation 2
Volume horaire : 72h étudiant répartis CM/TD=36/36
UE en évolution par rapport aux moutures précédentes.

Formation : Licence Informatique
Niveau : L2, semestre S4
UE (mouture 2021-2025) : Algorithmique des images, du texte et des données
Volume horaire : 36h étudiant répartis CM/TD=18/18
Nouvelle UE par rapport aux moutures précédentes.

1.3 Diagnostic et objectifs généraux

On se restreint dans un premier temps à l'analyse de l'existant (UE de Licence 1).

1. **Augmentation de l'hétérogénéité des acquis des étudiants.** Et ce pour plusieurs raisons dont une nouvelle.

    - Un nouveau public lycéen arrive à la rentrée 2021 suite au nouveau programme du lycée avec l'introduction de la nouvelle spécialité *Numérique et Sciences Informatique* (NSI) en première, et surtout en terminale (4h/semaine en 1ère et 6h/semaine en terminale).
      Le programme complet de NSI est très ambitieux : il couvre une partie très importante de l'UE ici concerné
      A ce jour, il est difficile 1) d'estimer le nombre d'étudiants qui auront suivi ces enseignements de façon complète (2 ans) ou partielle (1 an), 2) de juger de l'acquis effectif de cette catégorie d'étudiants.
      En revanche, il est clair que ces étudiants sont *de facto* intéressés par l'informatique. En

s'inscrivant en Licence d'Informatique, ils espèrent une formation universitaire adaptée à ce choix effectué 2 ans auparavant et confirmé depuis. Compte tenu des réserves indiquées, l'organisation générale de la Licence version 2021 n'intègre pas de parcours particulier pour ces étudiants.
Sans oublier la réserve 2), il est opportun de **mettre en place des déclinaisons particulières et avancées** de ces modules, en particulier pour le cas extrême des étudiants ayant un *acquis avéré* proche de celui attendu en fin de ces enseignements.

- Ce nouveau public va s'ajouter aux 2 « types » d'étudiants aujourd'hui présents au semestre 2 (S2). 1) Ceux qui ont acquis les notions élémentaires du semestre 1, qu'ils soient en Licence Math ou en Licence Info. 2) Ceux qui sont déjà en difficulté et dans ce cas, il faut dissocier le public de L1 Math de celui d'Info car cette UE est prioritaire pour les informaticiens et (en général) plus secondaire pour les matheux.

2. **Valeurs ajoutées réciproques de l'algorithmique et de la programmation.** La mouture 2021-2025 des Licences d'Informatique et de Mathématiques regroupe en une seule UE les enseignements d'algorithmique et de programmation, jusque-là traités dans 2 UE séparées. Ce regroupement permet d'**être plus qu'une simple concaténation des pratiques actuelles dans chacune des UE**. Il convient de proposer **un scénario d'apprentissage** où ces aspects sont **plus étroitement liés dans une approche par compétences**. Cette démarche, qui permet de répondre aux deux difficultés suivantes, s'appuie entre autres sur la définition de **mini-projets spécifiques**, *i.e.* qui intègrent les aspects indiqués ci-après.
3. **L'apprentissage de la programmation peut être chronophage.** En conseil de perfectionnement, les étudiants saluent le contrôle de connaissances de la programmation *effectué sur machine – vs.* sur copie. Sur les dernières années du contrat en cours, j'ai progressivement introduit des sujets de « mini-projets de programmation » à rendre, corrigés, comptant pour le CC et qui – surtout – préparent aux épreuves terminales sur machine individuelle et en temps limité. Ce type d'activité s'est avéré à la fois très formateur pour tout le public étudiant mais très chronophage pour certains d'entre eux. Il convient de permettre aux étudiants d'**adapter leur investissement** sur ce type d'activité **de façon plus individualisée, plus adaptée à leur projet personnel de formation**. Cet aspect complète aussi l'objectif précédent.
4. **L'apprentissage de l'algorithmique peut poser problème** aux étudiants ayant des difficultés d'abstraction et/ou de langage. L'intérêt d'une approche plus pratique, basée sur la **résolution systématique de problèmes concrets** (*learning by doing*) grâce à une utilisation plus intégrée de la programmation est à la fois adaptée et opportune.
5. **Opportunité de calendrier.** La mise en place en septembre 2021 de l'évolution de ce module pour les nouvelles moutures de Licence nécessite une préparation adaptée lors de l'année universitaire 2020-2021.

6. **Continuité du message pédagogique.** Aujourd'hui, les TD sont en grande partie effectués par des intervenants non permanents, doctorants ou ater. Ainsi la majorité de l'équipe pédagogique est renouvelée chaque année. Il convient donc de limiter les effets de ce renouvellement et d'**homogénéiser le message pédagogique**, à la fois **sur la durée et sur l'ensemble du public concerné**. Dans ce cadre, mentionnons que je suis en charge du CM et d'un groupe de TD dans les 2 UE actuelles concernées. Je rédige donc les supports de CM, de TD, les mini-projets et les sujets d'évaluation continue et terminale de ces 2 UE. Il s'agit donc d'améliorer la transmission orale du message pédagogique global lors des activités de groupe, activités essentielles pour une apprentissage efficace.
7. **Au-delà des informaticiens.** Cet enseignement de base en informatique est susceptible d'intéresser des étudiants d'autres filières que les Licences d'informatique ou de mathématique, en formation initiale, continue voire en apprentissage. Afin que ce **potentiel d'intérêt pour d'autres cursus** puisse être exploité à terme, la déclinaison de cet enseignement sous une forme adaptée doit être envisagée. Pour cela, on compte s'appuyer sur la richesse des supports qui seront utilisés (*moodle*, écosystème des *notebooks jupyter*), adaptés (plates-formes de « juge en ligne ») voire développés, pour décliner un enseignement le plus autonome possible.

1.4 Modalités de réalisation

Nous serons guidés par les objectifs suivants.

- Renforcer l'acquisition des notions de l'UE Algorithmique et programmation par une *pratique améliorée*.
- Définir des objectifs d'apprentissage adaptés au projet personnel de formation de l'étudiant.e et mettre en oeuvre le parcours qui y conduit.
- Améliorer la valeur ajoutée des séances en présentiel de TD-TDO par rapport à une séance de travail personnel isolée.

1.4.1 Une pratique améliorée

Objectif : renforcer l'acquisition des notions de l'UE Algorithmique et programmation par une **pratique améliorée**.

Cette amélioration consiste à :
- pratiquer plus et de façon plus souple — chez soi en particulier,
- profiter de validations automatiques ou interactives de la solution proposée,
- être guidé vers les parties de cours utiles à la réalisation de la solution proposée,

- quantifier son assimilation des connaissances,
- évaluer ses acquis par rapport à aux besoins de son projet personnel de formation.

Cette pratique améliorée est fortement dépendante des outils logiciels qui seront mobilisés, adaptés ou développés dans ce projet (Voir Section *Aspects techniques*).

1.4.2 Un scénario d'apprentissage adapté et personnalisé

Objectif : Définir des objectifs d'**apprentissage adaptés au projet personnel** de formation de l'étudiant et mettre en oeuvre le parcours qui y conduit.

On pourra déjà au moins distinguer les cinq directions principales suivantes de poursuite d'étude envisagée par l'étudiant pour réaliser son projet personnel de formation : master d'informatique *vs.* licence professionnelle en informatique *vs.* emploi direct dans le domaine informatique *vs.* poursuite de la formation en mathématique avec lien fort avec l'informatique *vs.* poursuite de la formation en mathématique avec lien faible avec l'informatique.

Pour chacune, on définira **une grille de parcours des activités** proposées en qualifiant les notions à acquérir (programme) et en quantifiant le niveau d'apprentissage adapté (compétence).

Cette grille de parcours permet aussi d'améliorer la lisibilité des compétences à acquérir, bien sûr pour les étudiants, mais aussi pour les enseignants d'autres matières (meilleure lisibilité des pré-requis des autres UE).

1.4.3 Le présentiel comme amplificateur du travail personnel

Objectif : Améliorer la **valeur ajoutée des séances en présentiel de TP-TDO** par rapport à une séance de travail personnel isolée.

Il s'agit d'accompagner l'étudiant pour qu'il valorise au mieux les séances de TP-TDO (et TD) en présentiel. Le rythme de diffusion des sujets d'études et le dépôt (moodle) des productions des étudiants seront adaptés (par exemple : 2 semaines de préparation et dépôt de la production en fin d'une séance de TDO consacrée aux réponses et résolutions de problèmes mineurs).

1.5 Résultats attendus

- Un ensemble de contenus significativement plus riche qu'aujourd'hui afin de couvrir une très large variété de situations algorithmiques et ainsi répondre aux différentes déclinaisons de l'enseignement. Ces contenus prendront principalement la forme de *notebooks jupyter* augmentés pour être exploités par un système de type « juge automatique ». Dans ce projet, ils couvriront les UE de L1 et L2 menitonnés en début de document.

- L'environnement informatique qui permet d'exploiter ces contenus de façon autonome et automatiquement assistée.
- Une compétence augmentée par les sujets orientés vers les applications (*learning by doing*, nouveaux savoirs et nouveaux savoir-faire).
- Valeur ajoutée professionnelle en favorisant le travail collaboratif des étudiants avec une perspective métier (savoir-être).
- Souplesse des thématiques et de la complexité des contenus traités.
- Homogénéiser, sur la durée et le public étudiant, l'appui des intervenants non permanents.
- Améliorer le taux de réussite et la satisfaction des étudiants.

1.6 Aspects techniques

Cette démarche s'appuie de façon cruciale sur une **infrastructure logicielle** (à développer) qui intégrera à la fois l'écosystème des *notebooks jupyter* et un environnement logiciel de type "juge automatique" comme ceux des sites d'entraînement aux compétitions de programmation : Castor informatique, Algoréa, Olympiades d'informatique.

Parmi les enjeux techniques, nous serons attentifs à sa meilleure intégration avec les ressources *moodle*, et ce avec l'appui avec des services compétents de l'UPVD.

1.6.1 L'écosystème des *notebooks jupyter*

Ces « cahiers informatiques » permettent d'intégrer **en un seul support** les textes et les schémas du cours, les diapositives utilisées dans les CM, les programmes (informatiques) des démonstrations interactives effectuées lors des séances de CM, les exercices, les solutions proposées par les étudiants, la vérification de ces solutions, la définition de telles vérifications à la fois par l'enseignant mais aussi par l'étudiant.

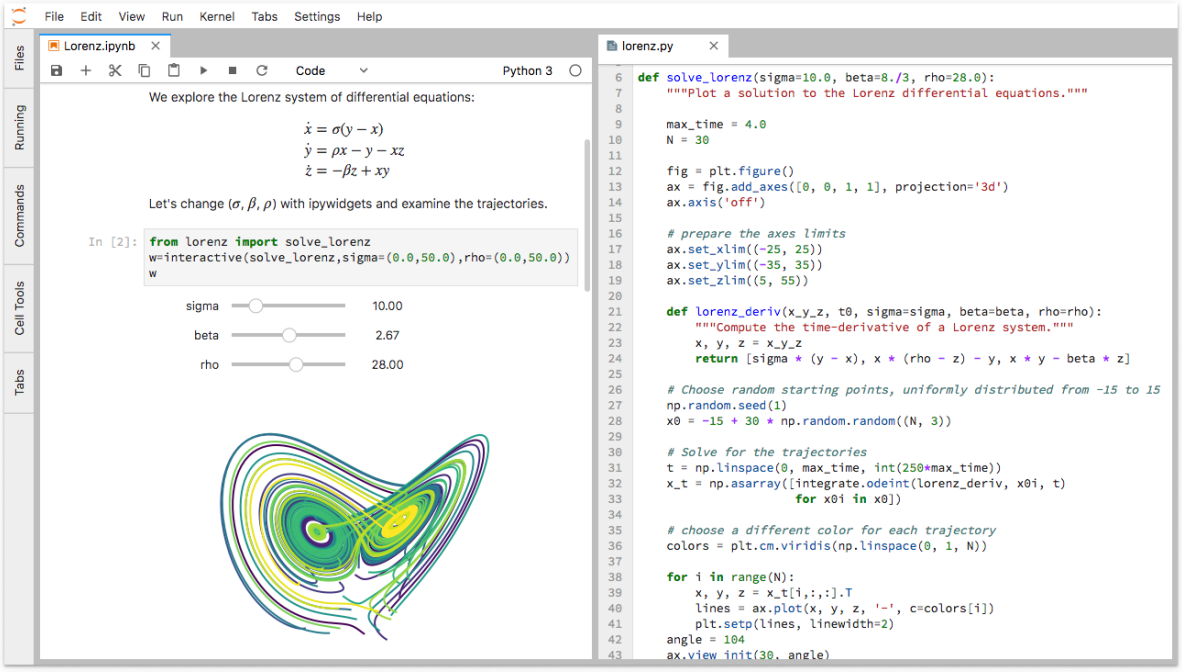


FIG. 1 : Un notebook jupyter

Au delà du regroupement en un document unique des différents éléments indiqués, il est important d'apprécier la valeur ajoutée de ces *notebooks* dans la cadre de l'apprentissage de l'algorithmique et de la programmation. En effet, les programmes écrits dans ces *notebooks* (par l'enseignant ou l'étudiant) s'exécutent (au sens informatique du terme) dans le *notebook* en interaction avec l'étudiant qui peut les modifier, les appliquer à de nouveaux jeux de tests qu'il définit, les réutiliser dans d'autres contextes applicatifs, … L'unité de support lui permet d'intégrer toutes les explications, descriptions, hypothèses qui jalonnent les différentes étapes de raisonnement nécessaires à l'écriture du programme qui répond à la question.

Ces *notebooks jupyter* que j'utilise depuis trois ans ont été salués lors de l'édition 2020 du colloque francophones de didactique de l'informatique Didapro-Didastic et décrits comme *« merveilleux … pour l'enseignement et des travaux pratiques en informatique »* (L. Besson, ENS Rennes).

1.6.2 Les environnements logiciels de "juge automatique"

L'écosystème des *notebooks jupyter* devra être articulé avec des environnements logiciels de "juge automatique" ou « juge en ligne ».

FIG. 2 : Exemple d'interface du juge en ligne de France-IOI

Pour un problème donné, l'apprenant soumet sa solution (qui est un programme informatique) à ce système logiciel qui la vérifie de façon automatique, la valide ou le cas échéant oriente l'apprenant vers les ressources adaptées. Les perspectives pédagogiques vers de meilleurs scénarios d'apprentissage sont incontestables, nombreuses et incroyablement motivantes.

En pratique et au-delà du choix des plates-formes logicielles les plus adaptées, ces validations s'appuient sur l'écriture de nombreux tests unitaires qui « augmentent » le *notebook jupyter* de façon transparente pour l'étudiant.e. Ces développements sont significativement chronophage. A titre personnel, j'ai pu mesurer que le temps d'écriture, de validation et d'intégration d'une version « augmentée » est environ 10 fois plus important que celui d'une simple correction (indépendamment de l'identification de la bonne solution).

1.7 Ressources

- Actes des colloques francophones de didactique de l'informatique Didapro-Didastic
- Société savante SIF
- Formations du CAP

- Outils et plates-formes logicielles : écosystème jupyter, camisole, domjudge

Dernière page du document :

title: “CPP 2020 - Rapport sur le projet pédagogique”

author: [Ph. Langlois]

date: 1er avril 2021