République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
ⵓⵖⵉⵣⵀ ⵏ ⵓⵙⵉⵎⵙⵍ ⵙⵏⵣⵅ ⵏ ⵙⵍⵓⵍⵙ ⵙⵓⵓⵍⵍ

Référentiel de la Formation Initiale de Troisième Cycle

« Fondements et Techniques de Programmation »

2024 - 2025

Comité national d'encadrement, de suivi et d'exécution du programme de formation initiale du 3ème cycle dans les établissements de l'enseignement supérieur.

- M. BENYAMINA Saïd — Président de la commission nationale
- Mme. BOUALLOUCHE Rachida — Directrice de la Formation Doctorale
- M. ZEBOUCHI Mohamed Abderraouf — Sous-directeur de la Formation Doctorale
- M. BOUAROURI Djaafar — Président de la Conférence Régionale des Universités du Centre
- M.LATRECHE Mohamed El Hadi — Président de la Conférence Régionale des Universités de l'Est
- M. CHAALAL Ahmed — Président de la Conférence Régionale des Universités de l'Ouest

Comité pédagogique national de la matière « Fondements et Techniques de Programmation »:

- Dr. Said GADRI — Président
- Pr. Abdelkader LAOUID — Membre Expert
- Dr. Abdelkader GHAZLI — Membre Expert
- Pr. Karim SEHABA — Membre Expert
- Dr. Abdelhakim HANNOUSSE — Membre Expert
- Dr. Samia ZOUAOUI — Membre Expert
- Dr. Mohamed Cherif BOUKALA — Membre Expert
- Dr. Belkacem KHALDI — Membre Expert
- Pr. Safia NAIT-BAHLOUL — Membre Expert

Table des matières

1. Préambule

Les défis du monde contemporain à l'ère de la programmation, de l'intelligence artificielle et des nouvelles orientations technologiques, imposent aux chercheurs une adaptation constante. La programmation est aujourd'hui au cœur des innovations et des avancées scientifiques, impactant tous les domaines de recherche.

L'acquisition des compétences en programmation est désormais indispensable pour les doctorants, quelle que soit leur spécialité. Elles leur permettent d'exploiter efficacement les outils numériques et d'optimiser leurs travaux de recherche. Dans cette optique, un programme de formation en programmation a été élaboré afin de leur fournir les connaissances et les outils essentiels dans ce domaine clé.

Dans le prolongement de cette démarche, l'intégration de concepts avancés tels que la pensée informatique, la modélisation computationnelle et le raisonnement logique vise à permettre aux doctorants d'adopter une approche méthodique et structurée dans la résolution des problématiques scientifiques. Cette formation leur permet non seulement de comprendre les fondements théoriques des langages de programmation et des structures de données, mais aussi de les appliquer concrètement dans leurs travaux de recherche, en exploitant les outils et plateformes adaptés à leurs disciplines respectives.

De plus, l'accent est mis sur le développement de compétences transversales essentielles, telles que l'autonomie dans l'apprentissage des nouvelles technologies, la collaboration interdisciplinaire et l'optimisation des processus de recherche. En maîtrisant ces compétences, les doctorants seront mieux préparés à relever les défis complexes liés à leurs thématiques de recherche, à innover et à proposer des solutions pertinentes aux problématiques émergentes dans leurs domaines respectifs.

2. Objectifs de la formation

- Permettre au doctorant d'acquérir de nouveaux outils de communication avec l'ordinateur dans les domaines de la programmation.
- Renforcer la pensée computationnelle et le raisonnement logique pour analyser, structurer et résoudre des problèmes complexes.
- Donner au doctorant les compétences nécessaires pour comprendre, concevoir et développer des algorithmes et des applications fiables et

efficaces, afin de résoudre des problèmes relevant de leur sujet de recherche.

- Explorer les langages de programmation et les exploiter dans la résolution des problèmes de recherche.
- Maîtriser les outils de manipulation et d'analyse de bases de données en utilisant les techniques de programmation.
- Acquérir des compétences en programmation collaborative.
- Acquérir des compétences en programmation orientée objet et les utiliser pour développer des programmes et des applications fiables et efficaces.
- Développer le sens de l'innovation et de la créativité technologique par la pratique de la programmation.
- Favoriser l'utilisation des langages de programmation open source tels que Python.

3. Organisation de la formation

La formation est organisée en cinq volets complémentaires, abordant à la fois les bases essentielles et les concepts avancés de la programmation, avec un accent particulier sur le langage Python :

- **Volet 1 :** Introduction à la programmation
- **Volet 2 :** Notions et concepts de base de la programmation en Python
- **Volet 3 :** Manipulation des fichiers et des bases de données
- **Volet 4 :** Programmation structurée, modulaire et orientée objets
- **Volet 5 :** Collaboration et bonnes pratiques de la programmation

Chaque volet est accompagné d'une collection de supports pédagogiques interactifs, incluant :

- *Conférences et séminaires* pour approfondir les concepts clés.
- *Ateliers* pour appliquer les connaissances sur des cas concrets.
- *Études de cas et mini-projets* favorisant l'apprentissage par l'expérimentation.

L'objectif est de renforcer la culture de la programmation chez les doctorants et de leur fournir les compétences nécessaires pour concevoir et développer des solutions logicielles efficaces adaptées à leur domaine de recherche.

Étant donné que le programme de formation s'adresse à des doctorants issus de divers domaines et spécialités, ayant des niveaux de compétence en programmation variés et des parcours académiques différents en développement

informatique, nous avons conçu une structure modulaire et flexible permettant une répartition adaptée des volets du programme. Cette organisation vise à adapter l'apprentissage aux besoins spécifiques de chaque domaine tout en garantissant une base commune en programmation. Ainsi, le programme est structuré en fonction de deux grands domaines disciplinaires :

Doctorants en Sciences Humaines et Sociales (SHS) :

- Les volets 1, 2 et 3 sont obligatoires afin de leur fournir une initiation complète aux bases de la programmation et aux outils de développement, leur permettant de concevoir des programmes simples pour résoudre leurs problématiques de recherche.
- Les volets 4 et 5 sont optionnels et peuvent être suivis en fonction des besoins spécifiques liés à leurs travaux de recherche et à l'intégration des concepts avancés de programmation.

Doctorants en Sciences et Technologies (ST) :

- Les volets 1 et 2 sont optionnels, car ces doctorants possèdent généralement déjà des bases en programmation. Cependant, ils peuvent les suivre s'ils souhaitent renforcer leurs compétences sur certains aspects fondamentaux.
- Les volets 3 à 5 sont obligatoires, car ils couvrent des concepts avancés, des méthodologies et des outils essentiels au développement d'applications complexes adaptées aux exigences de la recherche scientifique et technique de ce domaine.

Cette structuration garantit une formation adaptée aux profils variés des doctorants tout en maintenant une cohérence pédagogique et un équilibre entre les exigences disciplinaires et les compétences transversales nécessaires à la recherche scientifique.

4. Équipe de formation

La formation est destinée aux doctorants et supervisée par des enseignants-chercheurs et des chercheurs permanents, ayant des compétences dans le domaine de la programmation au niveau de chaque établissement universitaire ainsi qu'au niveau régional et national.

5. Formation de formateurs

Afin d'assurer un déroulement fluide et efficace de la formation doctorale tout en favorisant un engagement actif et une contribution accrue des différents établissements universitaires à l'encadrement des doctorants, deux types de formations de formateurs sont proposés :

5.1. Séminaire :

Un séminaire de 2 à 3 heures sera organisé en ligne à l'intention des formateurs. Il visera à présenter les objectifs pédagogiques de la formation, les thématiques abordées, ainsi que les modalités de suivi et d'évaluation. Ce séminaire constituera un espace d'échange permettant aux formateurs de se familiariser avec les outils et ressources disponibles, d'échanger sur les approches pédagogiques à adopter et de clarifier leurs éventuelles interrogations. Cette session sera essentielle pour assurer une mise en œuvre cohérente et homogène de la formation dans les différents établissements.

5.2. Formation courte et intensive :

Certains établissements ne disposant pas de spécialistes en programmation pourront bénéficier d'une formation intensive de 3 à 5 jours. Cette formation intensive vise à doter les enseignants formateurs des compétences fondamentales en programmation, leur permettant ainsi d'assurer efficacement la formation des doctorants. À l'issue de cette formation, les participants auront acquis les bases nécessaires pour encadrer et accompagner les doctorants dans leur apprentissage, garantissant ainsi une montée en compétence progressive et rendant les établissements plus autonomes dans l'enseignement de la programmation.

6. Langue et mode d'enseignement

6.1. Langue d'enseignement :

En adéquation avec l'orientation de ministère de tutelle, l'usage de l'anglais dans cette formation est fortement encouragé. Toutefois, d'autres langues peuvent être utilisées selon l'appréciation du formateur.

6.2. Mode d'enseignement :

Le déroulement du cours se fait selon une approche hybride. Ainsi, la partie théorique, composée de supports de cours sous différents formats (ouvrages,

présentations PowerPoint, vidéos, exemples de code, etc.), sera accessible aux doctorants *en ligne* via une plateforme dédiée.

La partie *en présentiel* sera supervisée par des enseignants du domaine, désignés par l'établissement, et prendra la forme d'ateliers où des exercices pratiques et des mini-projets seront menés en prenant en considération les spécialités et les niveaux du groupe. Au début de ces ateliers, les enseignants pourront rappeler les concepts fondamentaux de la matière en s'appuyant sur le référentiel fourni. Le tableau ci-après résume les exigences de la formation doctorale en *« Fondements et Techniques de Programmation »*.

Structure de supervision	Matériel et plateformes	Profil de formateurs
• Établissements universitaires • Conférences régionales • Formation à distance	• Salles de formation équipées • Plateformes de formation disponibles (Padoc, Telum, Moodle, etc.) et d'autres à développer si nécessaire. • Connexion internet	Des enseignants universitaires et chercheurs permanents répondant aux critères suivants : • MCA ou plus pour les enseignants universitaires • MRA ou plus pour les chercheurs permanents. • Possédant des compétences en programmation.

7. Programmation avec les outils open source et logiciels libres

L'open source, ou encore le logiciel libre est bien plus qu'un choix économique ou technique, c'est un mode de pensée qui se focalise sur l'accessibilité et la transparence de l'information. Les logiciels open source représentent une famille de logiciels dont tout ou une partie du code est accessible à l'utilisateur ou au programmeur, modifiable et exploitable par un contributeur ce qui permet de l'adapter aux besoins spécifiques de chacun.

Les logiciels open source et les logiciels libres deviennent aujourd'hui la norme dans plusieurs domaines, en particulier le domaine de l'enseignement pour plusieurs raisons, notamment :

- Accès libre et gratuit
- Indépendance et pérennité
- Bons outils pour l'auto-apprentissage
- Développement rapide de compétences en programmation

- Respect de la vie privée et de l'éthique
- Compatibilité et interopérabilité
- Développement de l'écosystème interne de l'établissement d'enseignement

Pour tous ces avantages, on a choisi d'utiliser des outils de programmation open source y compris le langage Python, les différentes bibliothèques associées, les éditeurs de programmes, etc., afin de créer un environnement d'apprentissage de la programmation, libre, indépendant, efficace, collaboratif et innovant.

8. Critères d'évaluation

En adéquation avec les modalités hybrides susmentionnées, l'évaluation portera à la fois sur la partie théorique en ligne et sur la partie pratique en présentiel. En effet, les exercices, sous différentes formes (QCM, codage, etc.), seront mis en ligne sur la plateforme, et les doctorants devront y répondre. Quant à la partie pratique en présentiel, elle sera évaluée par le formateur de l'établissement dans le cadre d'une formation locale et portera sur la réalisation de mini-projets. La répartition de la note globale est la suivante :

- 20% pour la partie théorique en ligne.
- 80% pour la partie pratique en présentiel, dont :
  - 50% correspondent à une évaluation continue des ateliers.
  - 30% sont attribués à un mini-projet de fin de formation.

Il est à noter que l'évaluation porte uniquement sur les volets obligatoires. Pour le calcul de la note globale, on opte pour le système en pourcentage (80%, 75%, etc.) avec la fixation d'un seuil de validation.

9. Contenue de la formation

Cette partie présente les cinq volets du programme de formation, chacun étant structuré en quatre sous-sections : la première donne un aperçu du volet, la deuxième définit ses objectifs pédagogiques, la troisième précise les prérequis en lien avec les autres volets, et la quatrième détaille le contenu.

Volet 1 – Introduction à la programmation

Présentation du volet :

Ce premier volet de la formation est consacré à l'introduction à la programmation à travers le langage visuel *Blockly*. Il a pour objectif de

présenter les concepts fondamentaux de la programmation en abordant progressivement les notions de programme et de langage, les variables et types de données, les entrées/sorties, ainsi que les structures conditionnelles et les boucles. Afin d'en faciliter l'assimilation, chaque partie sera illustrée par des exemples concrets.

Objectifs du volet :

- Comprendre les notions de programme informatique et de langage de programmation.
- Découvrir les bases de la programmation à l'aide d'un langage visuel (*Blockly*).
- Apprendre les concepts fondamentaux : variables, types de données, entrées/sorties, structures conditionnelles et boucles avec *Blockly*.
- Transition progressive vers les langages de programmation textuels.

Prérequis :

- Aucun

Programme détaillé du volet :

- Notion du programme
  - Comprendre la logique séquentielle d'un programme à travers l'exécution ordonnée des instructions.
  - *Programme illustratif 1* : Atteindre une cible sous *Blockly-labyrinthe*.
- Notion de langage de programmation
  - Définition de la notion de langage de programmation.
  - Langage visuel contre langage textuel.
- Variables et types de données
  - Comprendre l'importance des types de données dans la gestion de la mémoire.
  - Introduction aux variables et aux types de données.
  - Types de données élémentaires : *entier*, *réel*, *chaîne de caractères*, *booléen.*
  - Types de données complexes : *tableau*, *dictionnaire*, *liste*, *objet.*
  - *Programme illustratif 2* : Addition de deux nombres.
- Entrées et sorties
  - Comprendre comment interagir avec l'utilisateur.

- *Programme illustratif 3* : Afficher le message ("*Bonjour* [*Nom*] [*Prénom*]"), les nom et prénom sont saisis par l'utilisateur.

- Structures conditionnelles
  - Comprendre les structures conditionnelles `if` et `if-else`.
  - *Programme illustratif 4* : Affichage conditionnel : "*Bonjour Monsieur* [*Nom*] [*Prénom*]" ou "*Bonjour Madame* [*Nom*] [*Prénom*]" en fonction du sexe.
- Boucles
  - Introduction aux boucles (`for`, `while do`, `do-while`).
  - *Programme illustratif 5* : Exécuter une instruction plusieurs fois pour calculer la somme 1+2+3…100.
- Transition vers les langages textuels
  - Traduction du code visuel en code textuel (de *Blockly* vers *Python*, *Javascript*, etc.).

Volet 2 – Notions et concepts de base de la programmation en Python

Présentation du volet :

Python est un langage de programmation interprété, polyvalent et facile à apprendre, réputé pour sa syntaxe claire et concise. Il est largement utilisé en développement Web, en intelligence artificielle, en science des données, et en automatisation. Grâce à ses nombreuses bibliothèques et outils puissants, Python est un choix privilégié aussi bien pour les débutants que pour les développeurs expérimentés. Dans ce volet, nous faisons le lien entre la programmation visuelle et textuelle en traduisant les concepts fondamentaux introduits avec *Blockly* (Volet 1) en Python, illustrant ainsi comment la logique de base s'adapte à un environnement de programmation réel.

Objectifs du volet :

- Promouvoir la logique de programmation chez le doctorant en apprenant les structures fondamentales de Python.
- Faciliter l'apprentissage de la programmation grâce à la syntaxe simple et lisible de Python.
- Renforcer progressivement l'autonomie par la résolution des exercices de plus en plus complexes et la réalisation de projets.

- Préparer à des compétences professionnelles en adoptant un langage largement utilisé dans des domaines divers : *éducation*, *santé*, *industrie*, *recherche scientifique*.
- Favoriser l'utilisation des langages de programmation open source tels que Python.

Prérequis :

- **Volet 1 :** Introduction à la programmation

Programme détaillé du volet :

- Introduction au langage de programmation Python
  - Pourquoi choisir Python ?
  - Bref historique de Python.
  - Positionnement de Python.
  - Outils essentiels utilisés pour la programmation en Python.
  - Installation et exécution de Python.
  - Premier programme en Python.
- Éléments fondamentaux de la programmation en Python
  - Fonction de sortie `print()`.
  - Commentaires en Python.
  - Variables et expressions.
  - Types de données simples : `int`, `float`, `str`, `bool`, `complex`.
  - Identification des types de données.
  - Formatage de données.
  - Affectation.
  - Conversion de types de données.
  - Opérateurs arithmétiques et priorité des opérateurs.
  - Opérateurs logiques.
  - Fonctions mathématiques fréquentes.
  - Fonction `input()`.
- Contrôle du flux d'exécution : Structures conditionnelles et Boucles
  - Instruction `if`.
  - Instruction `if-else`.
  - Instruction de conditions multiples.
  - Boucle `for` : simple et en utilisant la fonction `range()`.
  - Boucle `while`.
  - Instructions pour le contrôle des boucles : `break`, `continue`, `pass`.
- Structure de données complexes en Python

- Listes : principe et fonctions de manipulation.
- Utilisation des listes comme piles.
- Tableaux 1D : principe et manipulation.
- Tableaux 2D : principe et manipulation.
- Ensembles : principe et manipulation.
- Tuples : principe et manipulation.
- Dictionnaires : principe et manipulation.

- Traitement des chaînes de caractères
  - Détermination de la longueur d'une chaîne.
  - Concaténation des chaînes.
  - Extraction des sous-chaînes.
  - Formatage de chaînes.
  - Conversion entre nombres et chaînes.

Volet 3 – Manipulation des fichiers et des bases de données

Présentation du volet :

Ce volet se compose de deux parties. La première partie introduit les différentes techniques de gestion et de traitement des fichiers sous des formats courants (TXT, CSV, JSON, XML). Les doctorants y apprendront les principes de lecture, d'écriture et de manipulation de ces fichiers à l'aide du langage Python. La deuxième partie est consacrée à l'apprentissage des bases de données et à leur gestion en utilisant Python. Elle couvre les concepts fondamentaux des bases de données relationnelles et non relationnelles, ainsi que leur exploitation à travers des bibliothèques et outils spécifiques en Python.

Objectifs du volet :

- Apprendre les principes fondamentaux de la gestion des fichiers.
- Initier aux principales méthodes de gestion des fichiers : lecture, écriture, modification et suppression des fichiers.
- Développer la capacité à exploiter efficacement les outils et bibliothèques dédiés à la gestion des fichiers.
- Apprendre à créer, manipuler et interroger des bases de données avec *SQL* et *NoSQL*.
- Maîtriser l'utilisation de *SQLite*, *MySQL* et *PostgreSQL* avec Python.
- Utiliser des bibliothèques comme *sqlite3*, *SQLAlchemy*, *pymysql* et *psycopg2* pour la gestion des bases de données.

- Travailler avec des bases de données *NoSQL* telles que *MongoDB* avec *pymongo*.
- Développer des applications interactives exploitant des bases de données.

Prérequis :

- **Volet 1 :** Introduction à la programmation
- **Volet 2 :** Notions et concepts de base de la programmation en Python

Programme détaillé du volet :

- **Manipulation des fichiers en Python**
  - Introduction à la manipulation des fichiers et formats de données
    - Différences entre fichiers texte et fichiers binaires.
    - Présentation et usages des formats courants : `TXT`, `CSV`, `YAML`, `JSON`, `XLSX`, `XML`.
  - Création et gestion des fichiers en Python (`TXT`, `CSV`, `JSON`, `XML`)
    - Création et ouverture de fichiers.
    - Lecture, exploration et extraction de contenu.
    - Écriture et modification de fichiers.
    - Suppression des fichiers.
  - Opérations avancées sur les fichiers
    - Fusion et concaténation de plusieurs fichiers.
    - Division et segmentation d'un fichier en plusieurs parties.
    - Conversion et transformation entre différents formats de fichiers.

- **Manipulation des bases de données en Python**
  - Introduction aux bases de données
    - Définition et types de bases de données (relationnelles vs. non-relationnelles).
    - Concepts clés : *tables*, *relations*, *clés primaires* et *étrangères*.
    - Comparaison *SQL* vs. *NoSQL*.
  - Manipulation de bases de données *SQL* avec Python
    - Introduction à *SQLite* : *sqlite3* en Python.
    - Création et gestion de bases de données *SQLite*.
    - Introduction à *MySQL* et *PostgreSQL*.
    - Utilisation de *pymysql* et *psycopg2* pour l'interaction avec *MySQL/PostgreSQL*.

  - Exécution de requêtes SQL : `SELECT`, `INSERT`, `UPDATE`, `DELETE`.
  - Gestion des transactions et sécurité des requêtes avec *SQLAlchemy*.
- Bases de données *NoSQL* avec Python
  - Introduction aux bases de données *NoSQL*.
  - MongoDB et son intégration avec Python via *pymongo*.
  - Création, mise à jour et suppression de documents dans *MongoDB*.
  - Requêtes avancées et indexation dans *MongoDB*.
- Connexion et intégration dans des applications Python
  - Connexion d'une application Python à une base de données.
  - Gestion des erreurs et optimisation des requêtes.
  - Utilisation d'un ORM (Object Relational Mapping) avec *SQLAlchemy*.
  - Stockage et récupération de données à l'aide d'*APIs REST*.

Volet 4 – Programmation structurée, modulaire et orientée objets

Présentation du volet :

Dans un monde où la complexité des logiciels ne cesse de croître, écrire des programmes fonctionnels n'est plus suffisant : ils doivent également être bien structurés, modulaires, réutilisables et évolutifs. Dans ce volet, les doctorants apprendront à structurer leurs programmes en exploitant les fonctions, modules et packages, garantissant ainsi une meilleure lisibilité et maintenabilité. Ils exploreront ensuite les concepts fondamentaux de la POO, afin de concevoir des programmes flexibles et évolutifs. Une attention particulière sera accordée aux bonnes pratiques de développement, notamment l'application de patrons de conception pour résoudre des problèmes de conception récurrents de manière élégante et efficace.

Objectifs du volet :

- Maîtriser la structuration efficace des programmes Python à travers l'utilisation de *fonctions*, *modules* et *packages*.
- Développer une expertise en programmation orientée objet (POO) en manipulant les concepts clés de ce paradigme afin de créer des programmes robustes et évolutifs.

- Concevoir un code maintenable et scalable en appliquant les principes SOLID et les patrons de conception, garantissant ainsi une architecture logicielle propre et efficace.

Prérequis :

- **Volet 1 :** Introduction à la programmation
- **Volet 2 :** Notions et concepts de base de la programmation en Python

Programme détaillé du volet :

- Introduction à la programmation structurée et modulaire
  - Principe de la programmation modulaire.
  - Définition et utilisation des fonctions en Python.
  - Organisation du code avec des modules, des packages et des bibliothèques.
  - Exploitation des bibliothèques standard et externes (ex. *matplotlib*, *Tkinter*, *scipy*).
- Programmation orientée objet avec Python
  - Principe de la programmation orientée objet.
  - Comparaison avec la programmation procédurale.
  - Concepts fondamentaux : *classes*, *objets* et *constructeurs.*
  - *Attributs* et *méthodes d'instance* et de classe, et *méthodes statiques.*
  - Encapsulation et modificateurs d'accès : *public*, *protégé* et *privé.*
  - Relations et interactions entre classes : *héritage*, *association*, *agrégation* et *composition*.
  - Polymorphisme et surcharge de méthodes : *classes et méthodes abstraites*, *redéfinition des méthodes.*
  - Diagrammes de classes : modélisation et visualisation avec *PlantUML.*
  - Applications pratiques de la POO,
- Architecture logicielle et patrons de conception
  - Introduction aux principes SOLID.
  - Les patrons de conception : définition, utilité et présentation des patrons de conception clés (*Singleton*, *Stratégie*, *Chaîne de responsabilité*, *Adaptateur*, *Composite* et *Décorateur*).
  - Études de cas et mise en œuvre pratiques.

Volet 5 – Collaboration et bonnes pratiques de la programmation

Présentation du volet :

Ce volet met l'accent sur l'utilisation d'outils de débogage en Python pour identifier et corriger efficacement les erreurs dans le code source. Il vise également à familiariser les doctorants avec des environnements de développement collaboratifs, optimisant la gestion du code et favorisant le travail en équipe.

Objectifs du volet :

- Développer une expertise en codage en appliquant les meilleures pratiques pour améliorer les compétences en programmation et concevoir des programmes de haute qualité.
- Maîtriser l'utilisation d'outils de détection d'erreurs pour optimiser le développement des programmes.
- Découvrir et appliquer divers types de tests logiciels, notamment des tests unitaires, des tests d'intégration, des tests de bout en bout et des tests fonctionnels.
- Explorer des outils DevOps pour la gestion du code source et le contrôle des versions.
- Utiliser des plateformes cloud pour stocker et partager du code, ainsi que pour favoriser le travail collaboratif.

Prérequis :

- **Volet 1 :** Introduction à la programmation
- **Volet 2 :** Notions et concepts de base de la programmation en Python
- **Volet 3 :** Manipulation des fichiers et des bases de données
- **Volet 4 :** Programmation structurée, modulaire et orientée objets

Programme détaillé du volet :

- Débogage, vérification et optimisation du code Python
  - Principes et définitions.
  - Débogage du code avec *debugpy*.
  - Amélioration de la qualité du code avec *flake8*.
  - Développement et exécution de tests avec *pytest*.
- Programmation cloud et développement collaboratif

  - Définition et concepts fondamentaux.
  - Développement cloud et programmation interactive en Python avec *Google Colab.*
  - Outils de codage collaboratif : *Codeshare*, *JupyterHub* et *Visual Studio Live Share.*
- Collaboration et progression avec *GitHub*
  - Introduction à *Git* et *GitHub* : principes, fonctionnalités et utilisation.
  - Services proposés par GitHub : *collaboration*, *revue de code*, etc.
  - Installation de *Git* et création d'un compte *GitHub.*
  - Création et utilisation d'un référentiel (repository).
  - Gestion des branches dans un système de contrôle de version.
  - Commandes indispensables de *Git* : `clone`, `status`, `branch`, `merge`, `pull`, `push`, `commit`.
  - Découverte et exploration de projets open source.
  - Élaboration d'un portfolio professionnel pour développeur.

10. Tableau des conférences

Objectifs	Thèmes	Modalité	Compétences	VH
Volet 1 : Introduction à la programmation				
- Comprendre les notions de programme informatique et de langage de programmation à l'aide d'un langage visuel (*Blockly*). - Apprendre les concepts fondamentaux à l'aide de *Blockly*.	**1.** Déclaration et manipulation des variables et types de données élémentaires **2.** Entrées/sorties à travers des opérations de lecture et écriture **3.** Structures conditionnelles simples et imbriquées **4.** Boucles sous leurs différentes formes **5.** Traduction du code visuel vers un code textuel	En ligne	C18, C19, C20	2h
Volet 2 : Notions et concepts de base de la programmation en Python				
- Présenter les concepts de base du langage Python - Comprendre les structures de données de Python (simples et complexes) - Maîtriser les structures de contrôle du langage (conditions et boucles)	**1.** Introduction au langage de programmation Python : notions et concepts de base **2.** Structures de données simples en Python **3.** Structures conditionnelles et boucles **4.** Structures de données complexes	En ligne	C17, C18, C19, C20	3h
Volet 3 : Manipulation des fichiers et des bases de données				
- Apprendre les bases de la manipulation de fichiers de différents formats en Python - Appliquer diverses opérations sur les	**1.** Introduction aux fichiers et formats de données **2.** Gestion et opérations avancées sur les fichiers	En ligne	C17, C18, C19, C20	

fichiers en Python - Maîtriser l'utilisation de *SQLite*, *MySQL* et *PostgreSQL* avec Python. - Utiliser des bibliothèques Python telles que *sqlite3*, *SQLAlchemy*, *pymysql* et *psycopg2* pour la gestion des bases de données relationnelles. - Travailler avec des bases de données *NoSQL*, notamment *MongoDB* avec *pymongo*.	**3.** Introduction aux bases de données relationnelles et NoSQL : concepts clés, SQL vs. NoSQL. **4.** Manipulation des bases de données avec Python : création, interrogation et gestion via SQLite, MySQL, PostgreSQL et MongoDB. **5.** Utilisation de bibliothèques Python : *sqlite3*, *SQLAlchemy*, *pymysql*, *psycopg2*, *pymongo*.			5h
Volet 4 : Programmation structurée, modulaire et orientée objets				
- Comprendre et appliquer les principes de la programmation structurée, modulaire et orientée objet pour améliorer la réutilisabilité du code. - Améliorer la maintenabilité du code en utilisant les patrons de conception.	**1.** Décomposition des programmes en fonctions, modules et packages **2.** Modélisation de systèmes réels à l'aide de classes et d'objets. **3.** Identification et implémentation des patrons de conception.	En ligne	C17, C18, C19, C20	3h
Volet 5 : Collaboration et bonnes pratiques de la programmation				
- Débogage, vérification et amélioration de la qualité du code - Exploration des outils pour la gestion du code et le contrôle des versions. - Développement collaboratif via des plateformes spécialisées.	**1.** Détection et correction des erreurs du code **2.** Amélioration de la qualité du code **3.** Création et manipulation de projets dans un environnement cloud **4.** Utilisation de plateformes de développement collaboratif.	En ligne	C17, C18, C19, C20, C21	3h

11. Tableau des séminaires

Volets	Objectifs	Thèmes	Modalité	Public Cible	Compétences	VH
Tous	- Présenter les objectifs pédagogiques, les thématiques et les modalités d'évaluation de la formation. - Se familiariser avec les outils et les ressources disponibles.	Présentation de la matière FTP : Objectifs, modalités d'évaluation et contenu	En ligne	Formateurs	C17, C18, C19, C20, C21	2h
Tous	- Comprendre les différentes familles de langages de programmation et leurs caractéristiques fondamentales. - Identifier les domaines d'application spécifiques de chaque famille en fonction des besoins et des contraintes. - Comprendre l'impact des logiciels libres et de l'open source dans le développement des logiciels (Cas du Python). - Découvrir les outils open source utilisés en programmation (IDE, frameworks, gestion de versions, etc.). - Favoriser l'engagement et la collaboration au sein de la communauté open source.	Les familles des langages de programmation, leurs domaines d'application et leur rôle dans le développement des logiciels libres et de l'open source.	En ligne	Doctorants	C17, C18, C19, C20, C21, C26	2h

12. Tableau des ateliers

Objectifs	Thèmes	Modalité	Compétences	VH
Volet 1 : Introduction à la programmation				
- Assimiler le principe des structures conditionnelles et les boucles à travers la réalisation des 10 niveaux du *Blockly labyrinthe*.	**1.** Structure conditionnelle simple et imbriquée **2.** Boucles dans ses différentes variantes	En présentiel	C18, C19, C20	1h 30
- Réaliser des programmes capables de communiquer avec l'utilisateur - Maîtriser les opérations et opérateurs de base en programmation (arithmétiques, logiques…) - Maîtriser les types de données élémentaires et quelques types composés	**1.** Lecture et écriture des données **2.** Opérateurs arithmétiques et logiques **3.** Types de données élémentaires (*entier*, *réel*, *chaîne de caractères*, *booléen*)	En présentiel	C18, C19, C20	1h 30
Volet 2 : Notions et concepts de base de la programmation en Python				
- Présenter les concepts de base du langage Python - Comprendre les structures de données simples de Python.	Présentation des concepts de base de python et des structures de données simples	En présentiel	C17, C18, C19, C20	2h
- Maîtriser les structures de contrôle du langage (conditionnelles, boucles)	Structures conditionnelles et boucles	En présentiel	C17, C18, C19, C20	1h 30
- Comprendre les structures de données complexes	Structures de données complexes	En présentiel	C17, C18, C19, C20	1h 30

Volet 3 : Manipulation des fichiers et des bases de données				
- Identifier les bibliothèques Python nécessaires pour manipuler différents formats de fichiers - Maîtriser des opérations fondamentales sur les fichiers - Réaliser des opérations avancées sur les fichiers.	Manipulation et gestion avancée des fichiers en Python	En présentiel	C17, C18, C19	2h
- Apprendre à créer, interroger et manipuler des bases de données SQL et NoSQL avec Python. - Utiliser les bibliothèques adaptées à chaque type de base de données. - Intégrer une base de données dans une application Python.	**1.** Introduction aux bases de données relationnelles et NoSQL. **2.** Utilisation de *sqlite3*, *SQLAlchemy*, *PyMySql*, *Psycopg2*, *PyMongo*. **3.** Requêtes SQL de base et avancées. **4.** Connexion d'une application Python à une base de données.	En présentiel		3h
Volet 4 : Programmation structurée, modulaire et orientée objets				
- Écrire des fonctions et des modules - Créer des graphiques avec *matplotlib* et des interfaces graphiques simples avec *Tkinter*. - Utiliser *SciPy* pour des calculs scientifiques.	Introduction à la programmation structurée et modulaire en Python	En présentiel	C17, C18, C19	2h
- Concevoir des programmes modulaires et évolutifs en appliquant la POO pour modéliser des cas concrets.	Programmation orientée objet et			

- Générer et visualiser des diagrammes de classes avec *PlantUML*. - Exploiter des patrons de conception utiles : *Singleton*, *Stratégie*, *Chaîne de responsabilité*, *Adaptateur*, *Composite* et *Décorateur*.	patrons de conception avec Python	En présentiel	C17, C18, C19	4h
Volet 5 : Collaboration et bonnes pratiques de la programmation				
- Explorer quelques outils de débogage et d'amélioration de la qualité du code.	Développement de code Python de bonne qualité grâce à l'utilisation de *debugpy*, *flake8* et *pytest*.	En présentiel	C17, C18, C19, C20, C21	1h 30
- Apprendre à développer dans des environnements cloud qui favorisent le partage, la gestion des versions et le travail collaboratif.	Mise en place et administration d'un projet sur GitHub.	En présentiel	C17, C18, C19, C20, C21	1h 30

13. Tableau récapitulatif de volume horaire

Volet	Conférences	Ateliers	Séminaires	Sous-total
Volet 1 : Introduction à la programmation	2h	3h	2h	5h
Volet 2 : Notions et concepts de base de la programmation en Python	3h	5h		8h
Volet 3 : Manipulation des fichiers et des bases de données	5h	5h		10h
Volet 4 : Programmation structurée, modulaire et orientée objets	3h	6h		9h
Volet 5 : Collaboration et bonnes pratiques de la programmation	3h	3h		6h
Total	16h	22h	2h	**40h**

14. Annexes

Annexe 1 : Compétences visées

Compétences	Position dans la grille	Volets
Développer de nouvelles compétences en programmation et en intelligence artificielle.	C17	Tous
Renforcer la pensée computationnelle et le raisonnement logique pour analyser et structurer des problèmes complexes.	C18	Tous
Développer des capacités permettant de comprendre, concevoir et développer des algorithmes et des applications fiables et efficaces pour résoudre des problématiques de recherche.	C19	Tous
Explorer les langages de programmation et les exploiter dans la résolution des problématiques de recherche.	C20	Tous
Acquérir des compétences en programmation collaborative et en travail collectif.	C21	Tous
Adopter l'esprit de partage, de collaboration et d'éthique par l'utilisation des logiciels libres et de l'Open Source.	C26	Tous

Annexe 2 : Répartition des volets selon les domaines (O : Obligatoire, F : Facultatif ou Optionnel)

Discipline	Part 1	Part 2	Part 3	Part 4	Part 5
Sciences Humaines et Sociales (SHS)	O	O	O	F	F
Sciences et Technologie (ST)	F	F	O	O	O

Annexe 3 : Ressources matérielles mobilisées (par établissement Universitaire) *

N.	Désignation	Caractéristiques minimales requises
1	Salles de formation équipées	
2	PCs	Processeur : I5, 6Gen, RAM : 8Go

3	Laptops (pour formateurs)	I5/I7, 10Gen, RAM v: 16Go
4	Data show	Epson, port WIFI
5	Panneau de projection	
6	Point d'accès Wi-Fi	
7	Tableau blanc	1.60x0.90
8	Pointeur Laser	

* L'annexe 3 illustre les exigences matérielles minimales par établissement. Chaque établissement universitaire doit déterminer ses exigences en fonction du nombre de ses doctorants.

Annexe 4 : Ressources humaines (par établissement universitaire)

N.	Désignation	Tâches principales
1	Instructeurs principaux	Enseignement des cours
2	Instructeurs assistants	Supervision des ateliers
3	Ingénieurs/Techniciens en informatique	Maintenance, installation

Annexe 5 : Ressources logicielles

Volets	Software Tools	Source/Link
Volet 1	*Blockly Labyrinthe*	https://blockly.games/maze?lang=fr
	Blockly demo	https://blockly-demo.appspot.com/static/demos/code/index.html
	Notepad ++	https://notepad-plus-plus.org/downloads/
	Sublime Text	https://www.sublimetext.com/
Volets 2-4	*Python & Python IDLE*	www.python.org
	Anaconda	www.anaconda.com
	PyCharm	https://www.jetbrains.com/pych arm/
	Jupyter	www.jupyter.org
	Spider	https://www.spyder-ide.org/

	PyTorch	https://pytorch.org/
Volet 4	*PlantUML*	https://plugins.jetbrains.com/plugin/7017-plantuml-integration
Volet 5	*Git*	https://desktop.github.com/download/
	debugpy	https://pypi.org/project/debugpy/
	flake8	https://pypi.org/project/flake8/
	pytest	https://pypi.org/project/pytest/

Annexe 6 : Fiche Descriptive du Séminaire

Titre :

Les familles des langages de programmation, leurs domaines d'application et leur rôle dans le développement des logiciels libres et de l'open source.

Présentation du séminaire :

Ce séminaire propose une exploration des différentes familles de langages de programmation et de leurs domaines d'application, en mettant particulièrement l'accent sur leur rôle dans le développement des logiciels libres et open source. Il introduira les principales familles et paradigmes de programmation avant d'examiner leurs divers domaines d'application. Une partie significative du séminaire sera consacrée à l'introduction des concepts fondamentaux des logiciels libres et open source, en clarifiant les distinctions entre ces deux approches et en analysant les différents modèles de licences. L'accent sera également mis sur l'utilisation des outils open source pour la programmation et le développement logiciel. Enfin, une session interactive permettra aux doctorants d'échanger sur les perspectives et tendances du développement open source, tout en encourageant leur engagement dans cette dynamique collaborative et innovante.

Objectifs du séminaire :

- Explorer les principales familles de langages de programmation.
- Analyser les domaines d'application des langages de programmation.
- Développer la capacité des doctorants à sélectionner le langage de programmation le plus adapté à un projet en fonction des exigences spécifiques, du domaine d'application et des contraintes techniques.

- Comprendre les différences entre les logiciels libres et open source ainsi que les types de licences associées.
- Comprendre l'impact des logiciels libres et open source dans la programmation et le développement des logiciels.

Programme détaillé du séminaire :

- Familles de langages de programmation
  - Langages d'assemblage.
  - Langages fonctionnels.
  - Langages impératifs et procéduraux.
  - Langages orientés objet.
  - Langages basés sur la logique.
  - Langages de script.
  - Langages déclaratifs.
  - Langages concurrents et parallèles.
- Domaines d'application des langages de programmation
  - Développement web et mobile.
  - Scientifique et calcul haute performance.
  - Programmation système et bas niveau.
  - Intelligence artificielle et data science.
  - Programmation embarquée et IoT.
  - Programmation polyvalente.
- Logiciels libres et open source
  - Logiciels libres vs. Logiciels open source.
  - Types de licences (GPL, MIT, Apache, etc.).
  - Avantages et défis des logiciels et outils open source.
- Programmation avec les outils open source
  - Environnements de développement.
  - Outils de gestion de version et collaboration.
  - Frameworks et bibliothèques open source populaires.
  - Cas pratiques (e.g., Python).

15. Bibliographie

Volet 1

1. Amber Lovett, *Coding With Blockly*, Cherry Lake Publishing, 2017

Volet 2

2. Al Sweigart, *Automate the Boring Stuff with Python: Practical Programming for Total Beginners*, No Starch Press, 2nd edition, 2019.
3. David Beazley and Brian K. Jones, *Python Cookbook*, O'Reilly Media, 3rd edition, 2013.
4. Eric Matthes, *Python Crash Course: A Hands-On – Project-Based Introduction to Programming*, No Starch Press, 2nd edition, 2019.
5. Florian Dedov, *The Python Bible 7 in 1: Volumes One to Seven (Beginner, Intermediate, Data Science, Machine Learning, Finance, Neural Networks, Computer Vision)*, Self-published, 1st edition, 2020.
6. Luciano Ramalho, *Fluent Python: Clear, Concise, and Effective Programming*, O'Reilly Media, 2nd edition, 2022.
7. Mark Lutz and David Ascher, *Learning Python*, O'Reilly Media, 5th edition, 2013.
8. Sweigart, A. *Invent Your Own Computer Games with Python,* No Starch Press, 4th edition, 2016
9. ليزا تاغليفيري، البرمجة بلغة بايثون: تعلم البرمجة وكتابة البرامج وتلقيحها بلغة بايثون. أكاديمية حاسوب. ترجمة: محمد بغات، عبد اللطيف أيمش. 2020
10. متاح على موقع PDFهديل محمد الطاهر. تعلّم البايثون للمبتدئين (بالعربي). مطبوعة . Foulabook.com

Volet 3

11. Severance C., *Python for Everybody: Exploring Data in Python,* CreateSpace Independent Publishing Platform, 2016. Available on : https://www.py4e.com/book
12. Barry P., Head *First Python: A Brain-Friendly Guide*, O'Reilly Media, 2nd edition, 2016
13. Panneerslvam R., *Databases and Python Programming: MySQL, MongoDB, OOP and Tkinter*, Amazon Digital Services LLC - Kdp, 2022
14. Albert Lukaszewski, *MySQL for Python*, Packt Publishing, 2010

Volet 4

15. Jeff Maynard, *Modular programming*, Auerbach Publishers, 1972
16. Erik Westra, *Modular Programming with Python*, Packt Publishing, 2016

17. Martin Abadi and Luca Cardelli, *A Theory of Objects (Monographs in Computer Science)*, Springer, Corrected edition, 1996
18. Brett McLaughlin, Gary Pollice and David West, *Head First Object-Oriented Analysis and Design: A Brain Friendly Guide to OOA&D*, O'Reilly Media, 1st edition, 2007
19. Steven F. Lott and Dusty Phillips, *Python Object-Oriented Programming: Build robust and maintainable object-oriented Python applications and libraries*, Packt Publishing, 4th edition, 2021
20. Mark Lutz, *Learning Python: Powerful Object-Oriented Programming*, O'Reilly Media; 6th edition, 2025
21. Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides and Grady Booch, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Professional, 1st edition, 1994
22. Eric Freeman and Elisabeth Robson, *Head First Design Patterns: Building Extensible and Maintainable Object-Oriented Software*, O'Reilly Media, 2nd edition, 2021
23. Kamon Ayeva and Sakis Kasampalis, *Mastering Python Design Patterns: Craft essential Python patterns by following core design principles*, Packt Publishing, 3rd edition, 2024

Volet 5

24. Downey A. B., *Think Python: How to Think Like a Computer Scientist*, O'Reilly Media, 2nd edition, 2015. Available on: https://greenteapress.com/wp/think-python-2e/
25. Slatkin B., *Effective Python: 90 Specific Ways to Write Better Python*, Addison-Wesley Professional, 2nd edition, 2019. Available on: https://effectivepython.com/
26. Mariot Tsitoara, *Beginning Git and GitHub: Version Control, Project Management and Teamwork for the New Developer*, Apress, 2nd edition, 2024

16. Webographie

Programmation en Python

27. https://www.freecodecamp.org/news/learn-python-free-python-courses-for-beginners/
28. https://www.udemy.com/topic/python/free/?srsltid=AfmBOorjgRsKTP-Sep3UKHhr8rafQSo6enqqdtxloqzkJWcrkM-6N7RF
29. https://www.youtube.com/watch?v=rfscVS0vtbw
30. https://www.youtube.com/watch?v=qwAFL1597eM
31. https://www.youtube.com/watch?v=eWRfhZUzrAc

32. https://www.youtube.com/watch?v=K5KVEU3aaeQ
33. https://www.youtube.com/watch?v=pdsc9SVW-S8
34. https://www.youtube.com/watch?v=6i3e-j3wSf0

Modularité, Débogage, Vérification, et Optimisation

35. https://code.visualstudio.com/docs/python/debugging
36. https://flake8.pycqa.org/en/latest/
37. https://docs.pytest.org/en/stable/
38. https://docs.github.com/en/get-started/git-basics
39. https://docs.github.com/en/get-started