



# Phirio

## IA - analyse et production de code informatique

IA053

Durée: 2 jours

1 610 €

31 janvier au 1er février  
23 au 24 mai

17 au 18 octobre  
2 au 3 décembre

### Public :

Développeurs, chefs de projet

### Objectifs :

Comprendre quels sont les apports de l'IA dans le process de développement, les principaux outils et savoir faire le choix adéquat selon le projet.

### Connaissances préalables nécessaires :

Connaissances de Python ou d'un autre langage de programmation structuré. Expériences en développement.

### Programme :

#### Apports de l'IA

Interventions à plusieurs stades du process de développement  
Vérification de la qualité du code, détection d'erreurs, de failles de sécurité, vérification de la syntaxe, des règles de développement,  
Analyse de code pour générer de la documentation, pour la gestion de sources,  
Génération de tests automatisés.  
Utilisation de modèles d'apprentissage automatique.  
Proposition de modules de codes, autocomplétion, génération de codes complets.  
Intérêts : aide aux développeurs, gains de temps, contrôle exhaustif, meilleure documentation, ..  
Quelques outils phares d'analyse de code : Pylint, Checkstyle, pycodestyle, Black, CodeQL,  
d'autocomplétion : Tabnine, Kite,  
et de production de code : Alphacode, GitHub Copilot, Codex.

#### pycodestyle

Objectif : vérification de respect des conventions d'écriture PEP8 (Python Enhancement Proposal 8),  
Fonctionnement : architecture modulaire, outil léger,  
Intégration aux principaux IDE (VS Code, Pycharm, JupyterNotebook, ...)  
renvoi direct aux erreurs lors de l'édition du code.  
Configuration en mode utilisateur ou projet.

#### Atelier : installation de pydecoestyle,

mise en oeuvre sur un programme simple,  
affichage des erreurs dans un code source,  
intégration à JupyterNotebook



# Phirio

---

## Pylint

Objectif : vérification du respect de la PEP8, détection d'erreurs de programmation, aide au refactoring, configuration des priorités utilisateur, intégration continue, intégration avec les principaux éditeurs et ide.

**Atelier : installation de Pylint,**

configuration de la détection d'erreurs, désactivation des vérifications de règles d'écriture et du refactoring, test sur des programmes caractéristiques.

---

## CodeQL

Objectif : recherche de failles de sécurité dans du code

Fonctionnement :

génération d'une base de données à partir du code, exécution de requêtes sur cette base pour détecter les failles.

Langages traités, architecture : soit en local avec CodeQL-CLI, soit sur GitHub, Principe d'intégration continue

**Atelier : mise en oeuvre de CodeQL pour Python**

Ecriture de requêtes basiques pour du code Python  
Utilisation de CodeQL library for Python

---

## Alphacode

Présentation du projet Google deepmind  
Principe de fonctionnement  
Résultats obtenus sur CodeForces

---

## Copilot

Présentation du projet Git alimenté par OpenAI Codex.  
Fonction : assistant virtuel en programmation.  
Fourniture de suggestions de lignes entières de codes ou de fonction entières  
Langages supportés

**Atelier : démonstration sur GitHub de l'utilisation de Copilot**

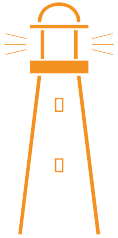
---

## Codex

Projet : OpenAI Codex, module du projet OpenAI, production de code informatique à partir de requêtes exprimées en langage naturel  
Ressources disponibles : bibliothèque de codes en ligne, hackathons, librairies Python, etc ...

**Atelier : démonstration avec OpenAI Codex**

Génération de code Python à partir d'un cahier des charges simple  
Tests et amélioration du code depuis l'interface en langage naturel.



# — Phirio —

---

## Les limites et risques

---

Nécessité d'une nouvelle organisation des tâches de développeurs.  
Définition du problème à résoudre, définition des contraintes, des jeux d'essai, etc ...  
Potentiels risques de sécurité, risques juridiques : origine des solutions utilisées, propriété du code/  
Erreurs de compréhension, d'analyse du problème à résoudre.  
Importance des contrôles humains à mettre en place.