MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

Concours externe de l'agrégation du second degré

Section informatique

Programme de la session 2022

L'agrégation d'informatique a été créée par le JO du 13 juin [1] qui définit également ses épreuves :

> *Les épreuves 1° et 2° d'admissibilité portent sur les programmes d'enseignement de la spécialité « numérique et sciences informatiques » (NSI) du cycle terminal de la voie générale du lycée, ceux des classes préparatoires scientifiques aux grandes écoles « mathématiques, physique, ingénierie et informatique » (MP2I) et « mathématiques, physique, informatique » (MPI), auxquels s'ajoute un programme complémentaire publié sur le site internet du ministère chargé de l'éducation nationale.*
>
> *L'épreuve 3° d'admissibilité porte sur les programmes mentionnés ci-dessus auxquels s'ajoute un programme complémentaire spécifique à chacun des sujets au choix (étude de cas informatique ou fondements de l'informatique). Ce programme complémentaire fait également l'objet d'une publication sur le site internet précité. Pour l'ensemble du programme, il est attendu du candidat un recul correspondant au niveau master.*

Ce document définit les programmes complémentaires pour ces épreuves d'admissibilité. On rappelle que ce programme est **complémentaire** et que le programme de l'agrégation d'informatique contient en premier lieu les programmes de NSI et d'informatique en MP2I et MPI. Certains éléments de ces programmes sont rappelés dans la présentation qui suit, afin de donner au présent document une forme cohérente.

Conformément aux programmes en vigueur, la connaissance des langages OCaml, C et Python est attendue, ainsi que la connaissance du langage de requêtes SQL. Les programmes des classes MP2I/MPI fixent dans leur annexe les éléments des langages C et OCaml, dont la connaissance est exigible. Ils fixent également, dans la partie 7, des éléments du langage SQL qui sont complétés ci-dessous dans la rubrique « Bases de données ». L'annexe A du présent programme fixe les éléments exigibles pour le langage Python.

1 Épreuves 1 et 2 d'admissibilité

Architecture

- Circuits combinatoires/séquentiels, machine de Mealy, machine de Moore.
- Description et fonctionnement d'une machine de von Neuman (introduction à la programmation assembleur : instructions de calcul, instructions de branchement, accès à la mémoire et aux registres, instructions système).
- Exécution d'un appel de fonction, concept de pile.
- Hiérarchie mémoire.
- Typologie des machines parallèles (classification de Flynn, classification de Raina, machines multicoeurs, supercalculateurs).
- Représentation des nombres à virgule flottante. Problèmes de précision des calculs flottants et de dépassement de capacité. Notion de mode d'arrondi.

Bases de données

- Création, suppression, modification de tables au travers du langage de requêtes SQL.
- Opérateurs de l'algèbre relationnelle et leurs propriétés : application à l'optimisation de requêtes.

1. `https://www.legifrance.gouv.fr/jorf/id/JORFTEXT000043648279`

MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

Concours externe de l'agrégation du second degré

Section informatique

Programme de la session 2022

Calculabilité, complexité

— Modèle de calcul. Machines de Turing : définition, principales variantes (ruban biinfini vs infini, machine à plusieurs rubans). La machine de Turing est le modèle de calcul retenu pour l'étude des notions qui suivent.
— Calculabilité : universalité, décidabilité, indécidabilité. Problème de l'arrêt.
— Complexité : complexité en temps et en espace, classe P. Acceptation par certificat, classe NP. Réduction polynomiale. NP-complétude. Théorème de Cook.

La notion de machine de Turing non déterministe n'est pas exigible aux épreuves 1, 2 et 3.a de l'écrit, ni aux épreuves orales.

Chaîne de compilation

— Analyse lexicale, analyse syntaxique (principes de l'analyse descendante).
— Analyse sémantique élémentaire (arbre de syntaxe abstraite, environnement, analyse de portée, typage).
— Génération de code vers une machine à pile.

Génie logiciel

Production de logiciel : cycle de vie, cycle de développement, agilité.

Informatique et société

— Points clés du RGPD.
— Notions de Données d'Intérêt Général (DIG).
— Impact environnemental du numérique. Notion d'écoconception.
— Points clés de la loi n° 2016-1321 du 7 octobre 2016 pour une République numérique.
— Enjeux d'éthique du numérique et valeurs sous-jacentes.

Intelligence artificielle

— Mesures de similarité pour l'apprentissage machine.
— Données d'entraînement et données de test, choix des descripteurs.
— Enjeux d'éthique (biais d'apprentissage, transparence).

Programmation

— Programmation objet : objets, classes, héritage, polymorphisme de sous-typage.
— Programmation fonctionnelle : ordre supérieur, structures immuables, polymorphisme paramétrique.

Réseaux

- Caractéristiques des réseaux et performances associées :
  - réseaux d'accès, réseaux de coeur ;
  - topologies de réseaux : point à point, à diffusion ;
  - performances : débit de transmission, délai, taux de perte.
- Modélisation en couches : TCP/IP, encapsulation.
- Transmission :
  - Adressage : adresses MAC, IPv4, IPv6 ;
  - Routage : principes ; routage à vecteur de distance (algorithme de Bellman-Ford), routage par état de liens (algorithme de Dijkstra) ;
  - Solutions de transport : principes ; TCP, UDP.
- Programmation réseau : API Sockets en Python et en C à l'aide d'un aide-mémoire fourni.

Systèmes d'exploitation

- Séquence de démarrage : de l'initialisation aux processus utilisateur.
- Abstractions fournies par le système : accès au matériel, répartition équitable des ressources.
- Liens entre système d'exploitation et applications : adressage physique et virtuel, notion de pagination, MMU, interruptions, appels systèmes, gestion des processus, gestion du temps (ticks et implémentation du temps partagé).
- Isolation et interaction entre les processus : espace mémoire d'une application, communication entre applications (`pipe`, `mmap`).
- Concurrence : modèles de cohérence (forte, faible, PRAM et au relâchement) et d'équité. Construction des mutex et sémaphores à partir des instructions atomiques *test and set*. Schéma lecteurs rédacteurs.

2 Épreuve 3.a – étude de cas informatique

Génie logiciel

- Analyse : identification et spécification des besoins fonctionnels, définition des scénarios d'utilisation (séquence d'interactions).
- Conception : architecture logicielle (en couches), conception modulaire et approche par patron de conception (*design pattern*). Diagramme de classes / de tables.
- Maintenance : test unitaire, *refactoring*.

Une connaissance exhaustive des patrons de conception n'est pas exigible à cette épreuve.

Programmation web

- JavaScript : Accès et modification du DOM, événements (mode de propagation et réaction aux évènements), requêtes HTTP (envoi et réception en HTML et en JavaScript), programmation asynchrone (`async` / `await`).

L'annexe B du présent programme complémentaire fixe les éléments pour le langage JavaScript dont la connaissance est exigible pour cette épreuve.

Systèmes d'exploitation

— Émulation et virtualisation : types d'hyperviseurs (type 1, type 2) et conteneur.
— Virtualisation matérielle et para-virtualisation.

3 Épreuve 3.b – fondements de l'informatique

Calculabilité, complexité

— Machines de Turing non déterministes : équivalence avec les machines de Turing du point de vue de la calculabilité. Définition de la classe NP comme la classe des problèmes résolubles en temps polynomial par une machine de Turing non déterministe, équivalence avec l'acceptation par certificat.
— Définitions et caractérisations des ensembles récursifs, récursivement énumérables.
— Fonctions primitives récursives : définition, schémas primitifs, minimisation bornée. Fonctions récursives : définition, équivalence avec les machines de Turing.
— Lambda-calcul pur comme modèle de calcul : définition, propriétés (dont confluence), stratégies. Équivalence avec les machines de Turing et les fonctions récursives.
— Thèse de Church-Turing.

Logique

— Logique du premier ordre (aspects syntaxiques) : langages, termes, formules, variables libres et variables liées, substitutions, capture de variables.
— Logique du premier ordre (aspects sémantiques) : interprétation d'une formule dans un modèle, validité, satisfiabilité, théories cohérentes, théories complètes, théories décidables, indécidables. Exemples de théories : égalité, arithmétique de Peano.
— Bases de données : calcul relationnel et théorème de Codd.

Fondements de la programmation

— Preuve de programmes : correction, terminaison. Méthodes élémentaires : assertions, préconditions et postconditions, invariants et variants de boucles, logique de Hoare, induction structurelle.
— Sémantique des langages de programmation : sémantiques opérationnelles. Application à un langage impératif restreint (IMP).
— Systèmes de types : types simples. Application à un langage fonctionnel simple (mini-ML sans polymorphisme), sûreté du typage.

MINISTÈRE
DE L'ÉDUCATION
NATIONALE,
DE LA JEUNESSE
ET DES SPORTS
Liberté
Égalité
Fraternité

Concours externe de l'agrégation du second degré

Section informatique

Programme de la session 2022

A Langage Python

Cette annexe liste limitativement les éléments du langage Python (version 3) dont la connaissance, est exigible des candidats à l'agrégation d'informatique. Ces éléments s'inscrivent dans la perspective de lire et d'écrire des programmes en Python ; aucun concept sous-jacent n'est exigible au titre de la présente annexe. Aucune connaissance sur un module particulier n'est exigible des candidats.

A.1 Traits et éléments techniques à connaître

Les éléments et notations suivants du langage Python doivent pouvoir être compris et utilisés par les candidats sans faire l'objet d'un rappel, y compris lorsqu'ils n'ont pas accès à un ordinateur.

Traits généraux

- Typage dynamique : l'interprète détermine le type à la volée lors de l'exécution du code.
- Principe d'indentation.
- Portée lexicale : lorsqu'une expression fait référence à une variable à l'intérieur d'une fonction, Python cherche la valeur définie à l'intérieur de la fonction et à défaut la valeur dans l'espace global.
- Appel de fonction par valeur : l'exécution de $f(e)$ évalue d'abord l'expression e puis exécute f avec la valeur obtenue. Certaines valeurs sont des scalaires (booléens, entiers, etc.) d'autres des adresses (listes, objets, etc.).

Types de base

- Opérations sur les entiers (`int`) : `+`, `-`, `*`, `//`, `**`, `%` avec des opérandes positifs.
- Opérations sur les flottants (`float`) : `+`, `-`, `*`, `/`, `**`.
- Opérations sur les booléens (`bool`) : `not`, `or`, `and` (et leur caractère paresseux).
- Comparaisons `==`, `!=`, `<`, `>`, `<=`, `>=`.

Types structurés

- Structures indicées immuables (chaînes, n-uplets) : `len`, accès par indice positif valide, concaténation `+`, répétition `*`, extraction de tranche.
- Listes : création par compréhension `[`e `for` x `in` s`]`, par `[`e`]` `*` n, par `append` successifs ; `len`, accès par indice positif valide ; concaténation `+`, extraction de tranche, copie (y compris son caractère superficiel) ; `pop` en dernière position. Identification des tableaux et des listes.
- Dictionnaires : création `{`c_1 `:` v_1`,` ..., c_n `:` v_n`}`, accès, insertion (d`[`k`]` `=` v), présence d'une clé (k `in` d), `len`, `copy`.

Structures de contrôle

- Instruction d'affectation avec `=`. Dépaquetage de n-uplets.
- Instruction conditionnelle : `if`, `elif`, `else`.

— Boucle `while` (sans `else`). `break`, `continue`, `return` dans un corps de boucle.
— Boucle `for` (sans `else`) et itération sur `range(`a`, `b`)`, une chaîne, un n-uplet, une liste, un dictionnaire au travers des méthodes `keys` et `items`.
— Définition d'une fonction `def` f`(`x_1`, ..., `x_n`)`, `return`.
— Exceptions : lever une exception avec `raise`, la rattraper avec `try/except`.

Programmation objet

— Classe, constructeur (`__init__` avec un premier argument `self`), attributs.
— Méthodes (avec un premier argument `self`).
— Encapsulation de données dans un objet.
— Application : construction de structures chaînées (cellules de listes) et arborescentes (nœuds d'arbres). Utilisation de `None` comme pointeur nul, comparaison avec `is None`. Encapsulation de la structure dans une autre classe.
— Héritage. On se limite à de l'héritage simple.

Divers

— Introduction d'un commentaire avec `#`.
— Utilisation simple de `print`, sans paramètre facultatif.
— Importation de modules avec `import` *module* ou `import` *module* `as` *alias* ou `from` *module* `import` $f, g, \ldots$
— Assertion : `assert` (sans message d'erreur).

A.2 Éléments techniques devant être reconnus et utilisables après rappel

Les éléments suivants du langage Python doivent pouvoir être utilisés par les candidats pour écrire des programmes dès lors qu'ils ont fait l'objet d'un rappel et que la documentation correspondante est fournie.

Types structurés

— Ensembles : création `set()`, insertion (`add`), présence d'un élément (e `in` s), `len`.

Divers

— Annotations de types. Vérification statique avec un outil tel que `mypy`.
— Manipulation de fichiers texte (la documentation utile de ces fonctions doit être rappelée ; tout problème relatif aux encodages est éludé) : `open`, `read`, `readline`, `readlines`, `split`, `write`, `close`. Construction `with`.
— Tirage pseudo-aléatoire avec la fonction `randint` du module `random`.

B Langage JavaScript

La présente annexe liste limitativement les éléments du langage JavaScript (7e édition du standard ECMA-262). Ces éléments s'inscrivent dans la perspective de lire et d'écrire des éléments de code javascript intégrés à des documents HTML. Aucun concept sous-jacent n'est exigible au titre de la présente annexe.

B.1 Traits et éléments techniques à connaître

Les éléments et notations suivants du langage JavaScript doivent être compris et utilisés par les étudiants sans faire l'objet d'un rappel, y compris lorsqu'ils n'ont pas accès à un ordinateur.

Traits généraux

- Typage dynamique.
- Passage par valeur.
- Délimitation des portées : par bloc pour `let`.
- Gestion de la mémoire : pile et tas, *garbage collector*.
- Retours à la ligne significatifs (*automatic semicolon insertion*) ; on privilégiera l'emploi du point-virgule.

Définitions et types de base

- Quatre types primitifs à considérer :
  - `boolean` (`true` / `false`). Opérateurs `!`, `&&`, `||`.
  - `number` (virgule flottante double précision). Opérateur `+`, `++`, `-`, `--`, `*`, `/` (point d'attention avec NaN).
  - `string` (chaîne de caractères UTF-16).
  - `undefined`.
- Opérateurs de comparaison `==`, `<`, `>`, `<=`, `>=`, `===` (égalité stricte).
- Définition de constantes (`const`), de variables (`let`).
- Définition de fonctions `function` *nom*`(`*param*`) { ... }`. Les paramètres non passés sont initialisés à `undefined`.

Types structurés

- Tableaux : `let` *t* `= [`*a*`,` *b*`,` *c*`]` ; lecture et écriture d'un terme de tableau par son indice *t*`[`*i*`]` ; fonctions `length`, `push`, `pop`, `forEach`.
- Objet : `let` *obj* `= { "`*nom_a*`" :` *valeur_a*`, "`*nom_b*`" :` *valeur_b* `}` ; lecture et écriture d'une propriété par son nom : *obj*`.`*nom_a*. Absence de valeur : `undefined`.

Structures de contrôle

— Conditionnelles : `if (`c`)` s et `if (`c`)` s `else` s'
— Boucles : `while (`c`)` s et `for (`$init$ `;` fin`;` $incr$`)` `s`

B.2 Éléments techniques devant être reconnus et utilisables après rappel

Manipulation des objets du DOM

— Objets du DOM (par exemple `Document`, `Node`, `Element`, `NodeList`)
— Requêtes sur le DOM (par exemple `document.getElementById()`, `document.querySelector()`, `document.querySelectorAll()`)
— Lecture / écriture du DOM :
  — en utilisant les propriétés (par exemple `id`, `src`, `value`, `style`, `innerHTML`).
  — en utilisant les méthodes (par exemple `createElement()`, `createTextNode()`, `getAttribute()`, `setAttribute()`, `appendChild()`).

Réaction aux événements DOM

— Les événements : l'objet `Event`, propagation des événements dans le DOM.
— Gestionnaire d'événements : notamment `addEventListener()`, `removeEventListener()`.

Asynchronisme et requêtes HTTP

— Principes : Code bloquant, Mono Thread, Fonction de CallBack.
— Programmation Asynchrone notamment avec Async et Await.
— Envoi de requêtes http (POST et GET) (par exemple avec la fonction asynchrone `fetch()` ou `XMLHttpRequest()`).