Ce document a été mis en ligne par l'organisme FormaV®.

Toute reproduction, représentation ou diffusion, même partielle, sans autorisation préalable, est strictement interdite.

Pour en savoir plus sur nos formations disponibles, veuillez visiter :
formav.co/explorer

BACHELOR UNIVERSITAIRE DE TECHNOLOGIE

INFORMATIQUE

ÉVALUATION LOCALE : COMPÉTENCE C1, RÉALISER UN DÉVELOPPEMENT D'APPLICATION

SESSION 2026

Durée : 2 h 30

SPÉCIALITÉ / UNITÉ	COEFF
INFORMATIQUE	3,0 INDICATIF, NON NATIONAL

Moyens de calculs et matériel autorisés :

L'usage de la calculatrice n'est pas autorisé.

Aucun document ou matériel autorisé en dehors de ceux fournis avec le sujet.

Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Ce sujet comporte 9 pages numérotées de 1/9 à 9/9.

Les documents réponses en fin de sujet sont à rendre avec la copie.

DOSSIER SUJET

MISE EN SITUATION PROFESSIONNELLE ET CAHIER DES CHARGES : ATELIER NUMÉRIQUE DES RIVES, ENTREPRISE FICTIVE SITUÉE À NANTES

Vous intervenez au sein de l'Atelier Numérique des Rives, qui souhaite disposer d'un petit programme de gestion des prêts de matériel informatique à ses salariés. Le programme sera réalisé en Python 3.11, sans bibliothèque externe. Il devra enregistrer des équipements, créer des prêts, restituer le matériel et produire quelques indicateurs de suivi.

Le travail demandé porte sur la conception, l'implémentation et la vérification d'une application simple. Les documents techniques fournis dans le dossier contiennent toutes les règles métier nécessaires. Les dates sont représentées par des chaînes au format **AAAA-MM-JJ**. Les identifiants d'équipement et de salarié sont des chaînes non vides.

Cette évaluation est construite à l'échelle locale à partir de la compétence C1 du référentiel du BUT Informatique. Le référentiel national ne définit pas d'épreuve écrite terminale nationale intitulée « C1 - Réaliser », ni de durée ou de coefficient national associé à une telle épreuve.

PARTIE 1 : Analyse fonctionnelle et règles de gestion (6 points)

Comprendre le besoin, identifier les données et formaliser le comportement attendu

À partir du cahier des charges et des documents DT 1 et DT 2, analysez le fonctionnement attendu avant de proposer ou d'écrire du code. Toute réponse doit s'appuyer sur les règles de gestion fournies.

Entités et règles métier

Élément	Attributs connus	Règle ou contrainte
Équipement	id, libellé, état	Un équipement disponible peut faire l'objet d'un prêt.
Prêt	id équipement, salarié, date de prêt, date de retour prévue, date de retour effective	Un équipement ne peut avoir qu'un seul prêt en cours.
Restitution	id équipement, date de retour effective	La restitution clôt le prêt en cours.

Synthèse des données nécessaires à l'application, à compléter pour les questions 1.1 et 1.2.

1.1 (2 points) À partir du cahier des charges, identifiez les principales opérations que l'application doit proposer. Pour chacune, précisez les données qu'elle reçoit et son résultat attendu. Présentez au moins trois opérations distinctes, dont l'enregistrement d'un prêt et la restitution d'un équipement.

1.2 (2 points) Formalisez les règles de validation d'une demande de prêt. Votre réponse doit notamment traiter l'existence de l'équipement, sa disponibilité, l'existence du salarié et la cohérence des dates. Distinguez les conditions qui entraînent le refus de la demande.

1.3 (2 points) Proposez une représentation textuelle des données d'un équipement et d'un prêt en utilisant des dictionnaires Python. Indiquez les clés retenues et la valeur utilisée pour représenter un prêt encore en cours lorsque la date de retour effective est inconnue.

PARTIE 2 : Conception et implémentation (8 points)

Écrire des fonctions correctes et produire des résultats conformes au cahier des charges

Le programme utilise les structures et règles précisées dans DT 1 et DT 2. Sauf indication contraire, vous pouvez présenter du code Python exécutable ou un algorithme suffisamment précis pour être directement transcrit en Python. Les fonctions doivent respecter les signatures données et ne doivent pas modifier les listes reçues lorsqu'une copie de travail est demandée.

Signatures des fonctions à réaliser

Fonction	Signature	Résultat attendu
Recherche d'un équipement	chercher_equipement(equipements, identifiant)	L'équipement correspondant ou None.
Création d'un prêt	creer_pret(equipements, prets, identifiant, salarie, date_pret, date_retour_prevue)	Un message de résultat et, en cas de succès, l'ajout du prêt.
Calcul des retards	lister_retards(prets, date_du_jour)	Une nouvelle liste contenant les prêts en retard.

Les noms et paramètres sont imposés pour faciliter l'intégration des fonctions.

2.1 (2 points) Écrivez la fonction **chercher_equipement(equipements, identifiant)**. Elle parcourt la liste d'équipements et renvoie le dictionnaire de l'équipement dont la clé **id** correspond à **identifiant**. Si aucun équipement ne correspond, elle renvoie **None**. Précisez le comportement de la fonction si la liste est vide.

2.2 (2 points) Écrivez une fonction **est_date_valide(date)** qui renvoie **True** si la chaîne respecte exactement le format **AAAA-MM-JJ** et représente une date existante dans le calendrier grégorien, et **False** sinon. Vous pouvez utiliser le module standard **datetime**. Indiquez comment sont traitées une chaîne vide et une date impossible telle que **2026-02-30**.

2.3 (2 points) Écrivez la fonction **creer_pret(equipements, prets, identifiant, salarie, date_pret, date_retour_prevue)**. Elle doit vérifier que l'équipement existe, qu'il est disponible, que **salarie** est une chaîne non vide, que les deux dates sont valides et que la date de retour prévue est postérieure ou égale à la date de prêt. En cas de succès, elle ajoute à **prets** un dictionnaire contenant les clés **equipement_id**, **salarie**, **date_pret**, **date_retour_prevue** et **date_retour_effective**, cette dernière étant initialisée à **None**. La fonction renvoie un message explicite en cas de refus et un message de confirmation en cas de succès.

2.4 (2 points) Écrivez la fonction **lister_retards(prets, date_du_jour)**. Elle renvoie une nouvelle liste des prêts dont la date de retour prévue est strictement antérieure à **date_du_jour** et dont la date de retour effective vaut **None**. Les prêts restitués ne doivent jamais être inclus. La fonction ne doit pas modifier la liste initiale. Donnez également sa complexité en temps en fonction du nombre *n* de prêts et justifiez-la brièvement.

PARTIE 3 : Vérification, tests et qualité de l'implémentation (6 points)

Construire des tests pertinents et diagnostiquer un défaut de programme

Utilisez les données des documents DT 2 et DT 3. Les tests doivent vérifier les cas nominaux et les cas limites. Les dates sont comparées au format ISO indiqué dans le contexte.

Extrait du jeu de données fourni

Équipement	État	Prêt en cours	Date de retour prévue
PC-014	disponible	non	Sans objet
TAB-006	emprunté	oui	2026-06-11
VID-002	disponible	non	Sans objet

Données de départ utilisées pour contrôler les fonctions.

3.1 (3 points) Complétez le document réponse DR 1 avec six cas de test pour les fonctions **est_date_valide**, **creer_pret** et **lister_retards**. Les six cas doivent couvrir au minimum : une date valide, une date impossible, un équipement disponible, un équipement déjà emprunté, une restitution déjà enregistrée et un prêt en retard non restitué. Pour chaque cas, indiquez les données d'entrée et le résultat attendu.

3.2 (2 points) Un développeur propose le code suivant : **if pret["date_retour_prevue"] < date_du_jour: retards.append(pret)**. Expliquez pourquoi cette condition ne respecte pas entièrement le cahier des charges. Donnez la condition logique corrigée pour ne retenir que les prêts en retard et non restitués.

3.3 (1 point) Pour une liste de *n* prêts, une fonction parcourt chaque élément une seule fois et effectue pour chacun un nombre constant d'opérations. Donnez sa complexité en temps en notation asymptotique.

DOSSIER TECHNIQUE ET RESSOURCES

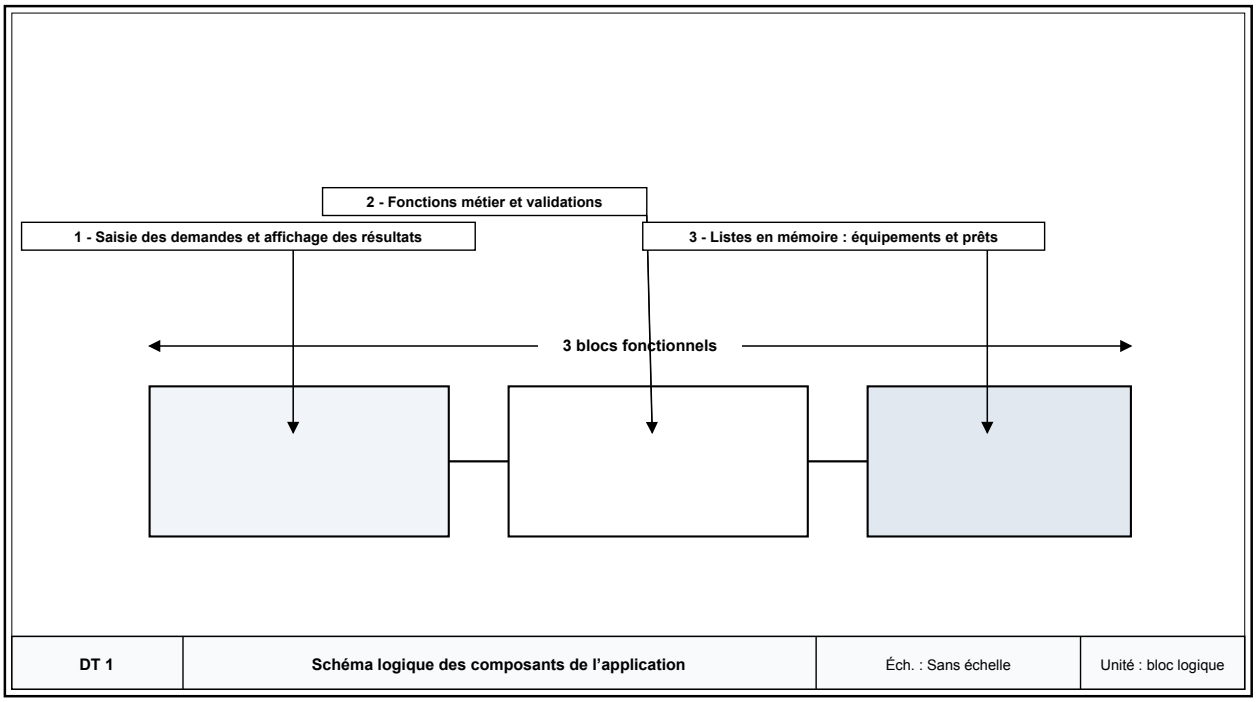
Documents ressources nécessaires à la résolution du sujet

DT 1 : ARCHITECTURE LOGIQUE ET RÈGLES FONCTIONNELLES DE L'APPLICATION

L'application est un programme local en Python 3.11. Elle ne communique avec aucun service externe et n'utilise aucune bibliothèque tierce. Les données sont chargées en mémoire sous forme de listes de dictionnaires. La persistance des données n'est pas demandée dans cette évaluation.

Règles fonctionnelles : un équipement possède un identifiant unique, un libellé et un état parmi **disponible** et **emprunté**. Un équipement est considéré comme ayant un prêt en cours si un prêt associé possède une valeur **None** pour **date_retour_effective**. Un prêt n'est accepté que si l'équipement existe et n'a aucun prêt en cours. Les dates sont des dates calendaires valides au format ISO **AAAA-MM-JJ**. La date de retour prévue peut être égale à la date du prêt, mais ne peut pas lui être antérieure.

Pour comparer deux dates valides au format ISO, l'application peut convertir les chaînes en objets date avec **datetime.date.fromisoformat**. Une exception de conversion indique une date invalide. Le traitement doit intercepter cette exception et produire un résultat contrôlé.



Représentation fonctionnelle sans échelle. Les dimensions indiquent l'ordre logique des blocs et ne représentent pas des dimensions physiques.

Structure des données

Structure	Clé	Type ou valeurs	Règle
Équipement	id	str	Identifiant unique et non vide.
Équipement	libelle	str	Description lisible de l'équipement.
Équipement	etat	str : disponible ou emprunté	État de disponibilité affiché.
Prêt	equipement_id	str	Identifiant d'un équipement connu.
Prêt	salarie	str	Nom ou identifiant non vide du salarié.
Prêt	date_pret et date_retour_prevue	str : AAAA-MM-JJ	Dates calendaires valides.
Prêt	date_retour_effective	str ou None	None signifie que le prêt est en cours.

Les champs indiqués sont les champs minimaux requis.

DT 2 : DONNÉES DE RÉFÉRENCE ET EXEMPLES DE PRÊTS

Le jeu de données ci-dessous constitue l'intégralité des données nécessaires aux cas de test. À la date du **2026-06-12**, le prêt de TAB-006 est en retard et n'a pas été restitué. Le prêt de PC-009 a été restitué et ne doit pas être compté comme un retard. PC-014 et VID-002 sont disponibles et n'ont pas de prêt en cours.

Exemples de représentation Python :

equipements = [{"id": "PC-014", "libelle": "Ordinateur portable", "etat": "disponible"}, {"id": "TAB-006", "libelle": "Tablette", "etat": "emprunté"}, {"id": "VID-002", "libelle": "Vidéoprojecteur", "etat": "disponible"}]

prets = [{"equipement_id": "TAB-006", "salarie": "S. Martin", "date_pret": "2026-06-01", "date_retour_prevue": "2026-06-11", "date_retour_effective": None}, {"equipement_id": "PC-009", "salarie": "A. Diallo", "date_pret": "2026-06-02", "date_retour_prevue": "2026-06-10", "date_retour_effective": "2026-06-09"}]

Données de test complètes

Type	Identifiant	Salarié ou libellé	Date de prêt	Retour prévu	Retour effectif
Équipement	PC-014	Ordinateur portable, disponible	Sans objet	Sans objet	Sans objet
Équipement	TAB-006	Tablette, emprunté	Sans objet	Sans objet	Sans objet
Équipement	VID-002	Vidéoprojecteur, disponible	Sans objet	Sans objet	Sans objet
Prêt	TAB-006	S. Martin	2026-06-01	2026-06-11	None
Prêt	PC-009	A. Diallo	2026-06-02	2026-06-10	2026-06-09

Liste des équipements et prêts disponibles pour l'évaluation.

Règles d'interprétation des résultats

Situation	Résultat attendu
Identifiant absent de la liste	Recherche renvoie None.
Équipement déjà associé à un prêt avec date_retour_effective égale à None	Création du nouveau prêt refusée.
Date invalide ou date de retour prévue antérieure à la date du prêt	Demande de prêt refusée sans ajout dans la liste.
Retour effectif renseigné	Le prêt n'est pas considéré comme en cours ni comme en retard.
Retour prévu strictement antérieur à la date du jour et retour effectif égal à None	Le prêt figure dans la liste des retards.

Ces règles s'appliquent à toutes les réponses.

DT 3 : CONSIGNES DE QUALITÉ, DE TEST ET DE COMPLEXITÉ

Les fonctions doivent produire un comportement prévisible pour les entrées nominales et les entrées invalides. Une fonction de validation renvoie une valeur booléenne et ne doit pas interrompre le programme par une exception non traitée. Une fonction de création ne doit ajouter aucun prêt si une seule des validations échoue.

Un test est décrit par des données d'entrée et un résultat attendu vérifiable. Les tests doivent être indépendants : l'état initial des listes est rétabli avant chaque cas qui modifie les données. Le parcours d'une liste de n éléments, avec un nombre constant d'opérations par élément, a une complexité en temps de l'ordre de **O(n)**. Aucune mesure de temps d'exécution n'est demandée.

Les bonnes pratiques attendues sont la clarté des noms, le respect des signatures, l'absence de modification involontaire des données et la séparation entre validation des entrées et mise à jour des prêts.

Grille minimale de couverture des tests

Catégorie	Cas à couvrir	Vérification à effectuer
Validation de date	Date valide et date impossible	Valeur booléenne correcte, aucune exception non traitée.
Création de prêt	Équipement disponible et équipement déjà emprunté	Ajout uniquement dans le cas accepté.
Retards	Prêt non restitué en retard et prêt restitué	Seul le prêt non restitué en retard est sélectionné.

Les candidats doivent proposer des entrées et des résultats attendus précis.

NE RIEN ÉCRIRE DANS CETTE PARTIE

DR 1 : Document réponse à rendre avec la copie : Questions 3.1 et 3.2 **À RENDRE AVEC LA COPIE**

Complétez le tableau avec six cas de test distincts. Indiquez des données concrètes et un résultat attendu vérifiable. Pour les cas qui modifient une liste, précisez l'état de la liste avant et après l'appel. Répondez à la question 3.2 dans l'espace prévu sur votre copie.

Plan de tests de l'application

N°	Fonction testée	Données d'entrée et état initial	Résultat attendu
1	est_date_valide	.....	.....
2	est_date_valide	.....	.....
3	creer_pret	.....	.....
4	creer_pret	.....	.....
5	lister_retards	.....	.....
6	lister_retards	.....	.....

Compléter les cases marquées par des points de suspension.

Cadre réservé aux justifications, calculs complémentaires ou schémas du candidat :

.....

.....

.....

.....

.....

.....

.....

NE RIEN ÉCRIRE DANS CETTE PARTIE

DR 2 : Document réponse à rendre avec la copie : Question 2.3 **À RENDRE AVEC LA COPIE**

Vous pouvez utiliser cette trame pour présenter la validation d'une demande de prêt. Complétez chaque ligne par la condition vérifiée et par la suite donnée à la demande en cas d'échec. Vous pouvez joindre une réponse sous forme de code Python sur votre copie.

Nomenclature des contrôles de création d'un prêt

Ordre	Contrôle	Condition à vérifier	Action si le contrôle échoue
1	Existence de l'équipement	.....	.....
2	Disponibilité de l'équipement	.....	.....
3	Identifiant du salarié	.....	.....
4	Validité des dates	.....	.....
5	Ordre chronologique des dates	.....	.....
6	Ajout du prêt	.....	.....

Compléter les cases marquées par des points de suspension.

Cadre réservé aux justifications, calculs complémentaires ou schémas du candidat :

.....

.....

.....

.....

.....

.....

.....

Copyright © 2027 FormaV®. Tous droits réservés.

FormaV® est une marque déposée auprès de l'INPI sous le numéro 5227897.

Ce document a été élaboré par FormaV® avec le plus grand soin afin d'accompagner chaque apprenant vers la réussite de ses examens. Son contenu (textes, graphiques, méthodologies, tableaux, exercices, concepts, mises en forme) constitue une œuvre protégée par le droit d'auteur.

Toute copie, partage, reproduction, diffusion ou mise à disposition, même partielle, gratuite ou payante, est strictement interdite sans accord préalable et écrit de FormaV®, conformément aux articles L.111-1 et suivants du Code de la propriété intellectuelle. Dans une logique anti-plagiat, FormaV® se réserve le droit de vérifier toute utilisation illicite, y compris sur les plateformes en ligne ou sites tiers.

En utilisant ce document, vous vous engagez à respecter ces règles et à préserver l'intégrité du travail fourni. La consultation de ce document est strictement personnelle.

Merci de respecter le travail accompli afin de permettre la création continue de ressources pédagogiques fiables et accessibles.