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

BUT Info

BUT Informatique : C1 Réaliser

Session 2026 • Code sujet : **26-BUT-INFO-EP1** • Durée : **2 h 30** • Coefficient : **3,0**
indicatif, non national

Sommaire

Cliquez sur une partie pour y accéder directement

1. Analyse fonctionnelle et règles de gestion (3 questions) **Page 3**

2. Conception et implémentation (4 questions) **Page 5**

3. Vérification, tests et qualité de l'implémentation (3 questions) **Page 8**

4. Ce qu'il faut retenir & Questions fréquentes (FAQ) (Synthèse & Q&A Examen) **Page 11**

Sujet officiel (PDF) : https://butinfo.fr/assets/annales/but-info/C1 - Réaliser/2026/BUT Informatique - C1 - Réaliser - Session 2026 - 1-Sujet.pdf

1. Analyse fonctionnelle et règles de gestion

L'Atelier Numérique des Rives souhaite gérer les prêts de matériel informatique à ses salariés avec une application locale en Python. Cette partie demande d'analyser les opérations, les données et les règles de validation à partir du cahier des charges et des documents techniques.

Question 1.1. À partir du cahier des charges et des documents DT 1 et DT 2, identifiez les principales opérations que l'application doit proposer. Pour chacune, précisez les données reçues et le résultat attendu. Présentez au moins trois opérations distinctes, dont l'enregistrement d'un prêt et la restitution d'un équipement.

Les opérations attendues couvrent la gestion des équipements, des prêts et de leur suivi. Les données reçues sont celles nécessaires à chaque action ; le résultat confirme l'action ou signale son refus.

Opération	Données reçues	Résultat attendu
Enregistrer un équipement	Identifiant, libellé, état	Équipement ajouté à la liste
Créer un prêt	Équipement, salarié, date de prêt, retour prévu	Prêt ajouté ou demande refusée
Restituer un équipement	Identifiant de l'équipement, date de retour effective	Prêt en cours clôturé
Consulter les retards	Prêts enregistrés, date du jour	Liste des prêts en retard non restitués

La restitution renseigne la date effective du prêt ; l'équipement peut alors être considéré disponible.

> **✓ RÉPONSE À RETENIR**
>
> Enregistrer les équipements, créer les prêts, enregistrer les restitutions et consulter les retards.

Question 1.2. Formalisez les règles de validation d'une demande de prêt. Traitez notamment l'existence de l'équipement, sa disponibilité, l'existence du salarié et la cohérence des dates, en distinguant les conditions qui entraînent le refus de la demande.

Refuser la demande dès qu'un contrôle échoue. L'identifiant doit correspondre à un équipement enregistré, dont l'état est `disponible` et auquel aucun prêt en cours n'est associé, c'est-à-dire avec `date_retour_effective` égal à `None`. Le salarié doit être représenté par une chaîne non vide ; le modèle fourni ne prévoit pas de liste permettant de vérifier son existence séparément.

Les deux dates doivent être des dates calendaires valides au format `AAAA-MM-JJ`. Le retour prévu doit être égal ou postérieur à la date du prêt. En cas de refus, aucun prêt n'est ajouté ; si tous les contrôles réussissent, le prêt est créé avec un retour effectif à `None`.

> **✓ RÉPONSE À RETENIR**
>
> Refuser si l'équipement est absent, indisponible ou déjà prêté, si le salarié est vide, si une date est invalide ou si le retour prévu précède le prêt.

Question 1.3. Proposez une représentation textuelle, sous forme de dictionnaires Python, des données d'un équipement et d'un prêt. Indiquez les clés retenues et la valeur utilisée lorsque la date de retour effective d'un prêt en cours est inconnue.

Les clés reprennent les champs minimaux indiqués dans DT 1. L'équipement choisi ci-dessous est emprunté et possède bien un prêt en cours, ce qui rend les deux dictionnaires cohérents. Les dates sont des chaînes au format ISO ; la valeur Python `None` indique que le retour effectif n'a pas encore été enregistré.

```
equipement = {
    "id": "TAB-006",
    "libelle": "Tablette",
    "etat": "emprunté"
}

pret = {
    "equipement_id": "TAB-006",
    "salarie": "S. Martin",
    "date_pret": "2026-06-01",
    "date_retour_prevue": "2026-06-11",
    "date_retour_effective": None
}
```

✓ RÉPONSE À RETENIR

Équipement : `id`, `libelle`, `etat`. Prêt : `equipement_id`, `salarie`, `date_pret`, `date_retour_prevue`, `date_retour_effective`. Retour inconnu : `None`.

2. Conception et implémentation

L'application utilise des listes de dictionnaires en mémoire et doit respecter les signatures de fonctions imposées. Cette partie porte sur la recherche d'équipements, la validation des dates, la création d'un prêt et la sélection des prêts en retard sans modifier la liste initiale.

Question 2.1. Écrivez la fonction chercher_equipement(equipements, identifiant). Elle doit parcourir les équipements et renvoyer le dictionnaire dont la clé id correspond à identifiant, ou None si aucun équipement ne correspond. Précisez son comportement lorsque la liste est vide.

Un parcours séquentiel suffit : dès que l'identifiant correspond, la fonction renvoie le dictionnaire. Si la liste est vide ou si aucun identifiant ne correspond, elle renvoie `None`.

```
def chercher_equipement(equipements, identifiant):
    for equipement in equipements:
        if equipement["id"] == identifiant:
            return equipement
    return None
```

✓ RÉPONSE À RETENIR

Équipement trouvé : son dictionnaire ; sinon, notamment pour une liste vide : `None`.

Question 2.2. 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 du calendrier grégorien, et False sinon. Indiquez le traitement d'une chaîne vide et d'une date impossible telle que 2026-02-30.

Vérifier le type, la longueur et les séparateurs impose le format exact. La conversion valide ensuite l'existence de la date ; `ValueError` est interceptée sans interrompre le programme.

```
from datetime import date

def est_date_valide(date_texte):
    if (not isinstance(date_texte, str)
            or len(date_texte) != 10
            or date_texte[4] != "-" or date_texte[7] != "-"
            or not date_texte.isascii()
            or not (date_texte[:4] + date_texte[5:7]
                    + date_texte[8:]).isdigit()):
        return False
    try:
        date.fromisoformat(date_texte)
        return True
    except ValueError:
        return False
```

✓ RÉPONSE À RETENIR

Une chaîne vide et `2026-02-30` renvoient `False` ; une date ISO existante renvoie `True`.

Question 2.3. Écrivez creer_pret(equipements, prets, identifiant, salarie, date_pret, date_retour_prevue). Vérifiez l'existence et la disponibilité de l'équipement, que salarie est une chaîne non vide, la validité des deux dates et leur ordre chronologique. En cas de succès, ajoutez à prets un dictionnaire contenant les clés equipement_id, salarie, date_pret, date_retour_prevue et date_retour_effective initialisée à None. Renvoyez un message explicite de refus ou de confirmation.

Chaque refus intervient avant toute modification de `prets`. La disponibilité exige un état « disponible » et l'absence de prêt associé encore en cours.

```
from datetime import date

def creer_pret(equipements, prets, identifiant, salarie,
               date_pret, date_retour_prevue):
    equipement = chercher_equipement(equipements, identifiant)
    if equipement is None:
        return "Refus : équipement inconnu."
    if equipement["etat"] != "disponible" or any(
        p["equipement_id"] == identifiant
        and p["date_retour_effective"] is None for p in prets
    ):
        return "Refus : équipement indisponible."
    if not isinstance(salarie, str) or not salarie.strip():
        return "Refus : salarié invalide."
    if not est_date_valide(date_pret) or not est_date_valide(date_retour_prevue):
        return "Refus : date invalide."
    if date.fromisoformat(date_retour_prevue) < date.fromisoformat(date_pret):
        return "Refus : retour prévu antérieur au prêt."
    prets.append({"equipement_id": identifiant, "salarie": salarie,
                  "date_pret": date_pret,
                  "date_retour_prevue": date_retour_prevue,
                  "date_retour_effective": None})
    return "Prêt créé avec succès."
```

DR 2 complété : contrôles de création

Nomenclature des contrôles de création d'un prêt

Ordre	Contrôle	Condition à vérifier	Action si le contrôle échoue
1	Existence de l'équipement	Équipement trouvé	Sinon : refus
2	Disponibilité de l'équipement	Disponible, aucun prêt actif	Sinon : refus
3	Identifiant du salarié	Chaîne non vide	Sinon : refus
4	Validité des dates	Deux dates ISO valides	Sinon : refus
5	Ordre chronologique des dates	Retour prévu ≥ date du prêt	Sinon : refus
6	Ajout du prêt	Validations réussies	Ajouter le prêt, confirmer

Compléter les cases marquées par des points de suspension

✓ RÉPONSE À RETENIR

Création acceptée si tous les contrôles réussissent ; le prêt ajouté a `date_retour_effective = None`.

Question 2.4. **Écrivez lister_retards(prets, date_du_jour) pour renvoyer 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 doivent être exclus et la liste initiale ne doit pas être modifiée. Donnez et justifiez la complexité en temps en fonction de n, le nombre de prêts.**

Une compréhension de liste construit une nouvelle liste sans modifier `prets`. La comparaison de chaînes ISO est chronologique pour des dates valides ; la borne est strictement antérieure.

```
def lister_retards(prets, date_du_jour):
    return [pret for pret in prets
            if pret["date_retour_effective"] is None
            and pret["date_retour_prevue"] < date_du_jour]
```

La fonction examine chaque prêt une fois, avec un travail constant par élément : sa complexité en temps est `O(n)`.

> **✓ RÉPONSE À RETENIR**
>
> Retenir les prêts non restitués dont le retour prévu précède aujourd'hui ; complexité : `O(n)`.

3. Vérification, tests et qualité de l’implémentation

Les essais s’appuient sur les équipements et prêts fournis, notamment un prêt en retard non restitué et un prêt déjà restitué. Il faut construire des tests indépendants couvrant les cas nominaux et limites, analyser une condition de sélection incomplète et exprimer une complexité asymptotique.

Question 3.1. Complétez le document réponse DR 1 avec six cas de test pour est_date_valide, creer_pret et lister_retards. Couvrez 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, précisez les données d’entrée et le résultat attendu vérifiable, ainsi que l’état des listes avant et après un appel qui les modifie.

Chaque cas est indépendant. Pour les créations, l’état initial est celui de DT 2 : trois équipements et deux prêts. Les cas de retards utilisent chacun un prêt isolé, avec la date du jour fixée au 2026-06-12. La liste de prêts fournie à `lister_retards` reste inchangée ; seul le résultat est une nouvelle liste. Les dates et états après création permettent de vérifier l’acceptation ou le refus.

N°	Fonction	Données d’entrée et état initial	Résultat attendu
1	est_date_valide	date = « 2026-06-12 »	True
2	est_date_valide	date = « 2026-02-30 »	False, sans exception non traitée
3	creer_pret	Équipements et prêts DT 2 ; PC-014, « A. Diallo », date_pret = 2026-06-12, date_retour_prevue = 2026-06-12. PC-014 est disponible et sans prêt en cours ; 2 prêts au départ.	Succès ; ajout du prêt avec date_retour_effective = None. La liste passe de 2 à 3 prêts.
4	creer_pret	Équipements et prêts DT 2 ; TAB-006, « S. Martin », date_pret = 2026-06-12, date_retour_prevue = 2026-06-12. Le prêt TAB-006 est en cours ; 2 prêts au départ.	Refus ; aucun ajout. La liste reste à 2 prêts.
5	lister_retards	prets = [{equipement_id: PC-009, date_retour_prevue: 2026-06-10, date_retour_effective: 2026-06-09}] ; date_du_jour = 2026-06-12.	Résultat : liste vide. La liste initiale reste inchangée.
6	lister_retards	prets = [{equipement_id: TAB-006, date_retour_prevue: 2026-06-11, date_retour_effective: None}] ; date_du_jour = 2026-06-12.	Résultat : nouvelle liste contenant uniquement TAB-006. La liste initiale reste inchangée.

DR 1 complété : six cas de test

N°	Fonction testée	Données d'entrée et état initial	Résultat attendu
1	est_date_valide	2026-06-12	True
2	est_date_valide	2026-02-30	False
3	creer_pret	PC-014, A. Diallo 12-06-2026 / 12-06-2026	Ajout : 3 prêts
4	creer_pret	TAB-006, prêt actif 12-06-2026 / 12-06-2026	Refus : reste 2
5	lister_retards	PC-009, retour 10-06 Restitué le 09-06	Résultat : []
6	lister_retards	TAB-006, retour 11-06 Retour effectif : None	Résultat : TAB-006

✓ RÉPONSE À RETENIR

Dates valides : True ; date impossible : False ; PC-014 accepté ; TAB-006 refusé ; PC-009 exclu ; TAB-006 retenu comme retard.

Question 3.2. Expliquez pourquoi la condition proposée, qui sélectionne un prêt uniquement lorsque sa date de retour prévue est antérieure à la date du jour, ne respecte pas entièrement le cahier des charges. Donnez la condition logique corrigée pour ne retenir que les prêts en retard et non restitués.

La condition proposée compare uniquement la date de retour prévue au jour courant. Elle peut donc sélectionner un prêt déjà restitué : PC-009 a une date de retour prévue au 2026-06-10, antérieure au 2026-06-12, mais son retour effectif est renseigné au 2026-06-09. Le cahier des charges exige les deux critères : retour prévu antérieur au jour courant et retour effectif égal à `None`.

Condition corrigée : `pret["date_retour_prevue"] < date_du_jour and pret["date_retour_effective"] is None`.

✓ RÉPONSE À RETENIR

Retenir uniquement un prêt dont le retour prévu est antérieur au jour courant et dont le retour effectif vaut None.

Question 3.3. Pour une liste de n prêts parcourue une seule fois, avec un nombre constant d'opérations pour chaque élément, donnez la complexité en temps en notation asymptotique.

La fonction parcourt la liste et examine chacun des *n* prêts une seule fois. Chaque examen effectue un nombre constant d'opérations, quel que soit le nombre total de prêts. Le coût total est donc proportionnel à *n* et augmente linéairement avec la taille de la liste. En notation asymptotique, la complexité en temps est `O(n)`.

✓ RÉPONSE À RETENIR

Complexité temporelle : O(n), car chaque prêt est parcouru une fois avec un coût constant par élément.

Ce qu'il faut retenir de ce sujet (Session 2026)

Bilan de l'épreuve : Cette évaluation locale de BUT Informatique, session 2026, code 26-BUT-INFO-EP1, dure 2 h 30 et indique un coefficient 3,0 non national. Autour d'une application Python de gestion des prêts de matériel, elle évalue l'analyse des règles métier, l'implémentation de fonctions, la validation des dates, les tests, la qualité des données et la complexité algorithmique.

Notions-clés, formules & règles à maîtriser

- Un prêt n'est accepté que si l'équipement existe et n'a aucun prêt en cours, c'est-à-dire avec date_retour_effective égale à None.
- Une date doit être valide au format ISO AAAA-MM-JJ; la date de retour prévue peut être égale, mais pas antérieure, à la date du prêt.
- Un prêt est en retard si sa date de retour prévue est strictement antérieure à la date du jour et si date_retour_effective vaut None.
- Un parcours de n éléments avec un nombre constant d'opérations par élément a une complexité en temps de O(n).

Points de vigilance & attentes du jury

- Respecter exactement les signatures des fonctions et les noms de clés des dictionnaires demandés.
- Intercepter les erreurs de conversion de date et renvoyer un résultat contrôlé, sans exception non traitée.
- Ne jamais ajouter de prêt si une validation échoue et ne pas modifier la liste initiale lorsqu'une nouvelle liste est demandée.
- Tester séparément les cas nominaux et limites, et rétablir l'état initial des listes entre les tests qui les modifient.

Questions fréquentes sur cette épreuve (Q&A Examen)

Q : Comment vérifier qu'une chaîne correspond à une date réelle au format demandé ?

Vérifiez le format ISO AAAA-MM-JJ et tentez de convertir la chaîne avec datetime.date.fromisoformat. Interceptez l'exception de conversion pour renvoyer False, notamment pour une chaîne vide ou une date impossible comme 2026-02-30.

Q : Comment savoir si un équipement peut faire l'objet d'un nouveau prêt ?

L'équipement doit exister dans la liste et ne pas être associé à un prêt en cours. Un prêt est en cours lorsque sa date_retour_effective vaut None; si cette condition est vérifiée pour un prêt associé, la création doit être refusée.

Q : Quels éléments faut-il prévoir dans les six tests demandés ?

Les six cas doivent couvrir les dates valide et impossible, la création pour un équipement disponible et le refus pour un équipement déjà emprunté, puis le prêt restitué et le prêt en retard non restitué. Précisez les entrées et résultats vérifiables.

Q : Qu'attend le sujet pour sélectionner les retards et justifier la complexité ?

Le critère de retard associe un retour prévu strictement antérieur à la date du jour à une date de retour effective égale à None. Le parcours examine les n prêts une fois avec un travail constant par prêt, ce qui correspond à O(n).

Sujet officiel associé (PDF) : https://butinfo.fr/assets/annales/but-info/C1 - Réaliser/2026/BUT Informatique - C1 - Réaliser - Session 2026 - 1-Sujet.pdf

Ressources BUT Informatique : Toutes les annales & corrigés • Fiches de révision complètes • Simulateur de notes d'examen

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.