Objectifs

Ce projet synthèse a comme objectif principal de mettre en application les notions de base de la programmation structurée et du développement d’algorithme. Ce projet est aussi une occasion de vous dépasser en tant que programmeur et d’aller au-delà de ce qui est demandé.

Exigences

**Développement/programmation :** Doit se faire de façon individuelle.

**Sujet du projet :** Réalisation d'un système de gestion de paie

**L'application doit :** ▪ Utiliser tous les éléments appris en cours de session : structures séquentielles/conditionnelles/répétitives, collections, modularisation, gestion d’exceptions/d’erreurs.

Fonctionner correctement, sans "bug" et donner les résultats attendus. N'oubliez pas de valider les données lues.

Respecter les normes de base de programmation, entre autres, en découpant de façon pertinente et en documentant le code.

Remise final

**Du 14 au 20 décembre 2020**

**AUCUN RETARD NE SERA ACCEPTÉ**

1 : Le sujet :

Une entreprise de développement de logiciels veut informatiser la gestion de la paie mensuelle de ses employés**.** À cet effet vous devez réaliser une application Python qui gère une liste d'employés et leurs paies ou chaque employé pouvant avoir une paie pour chaque mois. Votre application doit sauvegarder l'historique de toutes les paies à des fins d'analyse et de suivi. Cette application doit assurer les fonctionnalités suivantes :

* D’afficher la liste des employés ;
* De modifier les données de l'employé (en l’occurrence le nom).
* D’ajouter un employé ;
* De supprimer un employé;
* Saisir les informations d'une paie pour un mois donné;
* De calculer la paie pour un mois donné;
* D'afficher la paie de tous les employés ou d'un employé pour un mois donné;
* D’afficher l'historique de paie d'un employé donné.

2 : Les structures de données utilisées

Pour réaliser votre application, vous aurez besoin d'un modèle de données composé:

* D'une liste des employés qui sera représentée par un dictionnaire dont la **clé** est le **numéro** de l'employé et comme **valeur** une liste composée du **nom, prénom, la fonction** et le **taux horaire**. Exemple :

listeEmployes={

'E1' : ['Bounouar', 'Ahmed', 'Directeur',65],

'E2' : ['Gauthier', 'Pierre', 'Programmeur',40],

}

* D'une liste pour stocker les paies des employés qui sera représentée par un dictionnaire dont la clé est le numéro d'employé et comme valeur un autre dictionnaire dont la clé est représentée par le numéro du mois et une valeur représentée par une liste composée du nombre d'heures, le taux horaire, le salaire brut, l'impôt à payer et le salaire net. Exemple:

historiquePaies={

'E1':{1: [60, 65, 3900,780, 3120),

2: [65, 65, 4225, 845, 3380)

},

}

* Un dictionnaire pour stocker le nombre d'heures mensuelles travaillées par chaque employé, dont la clé est le numéro d'employé et comme valeur le nombre d'heures travaillées par mois. Cette liste sert pour calculer la paie mensuelle des employés. Exemple:

listeHeuresEmployes = {

'E1' : 60,

'E2' : 65

}

3 : Consignes de développement de l'application

**Remarque** : Du temps en classe sera consacré au travail sur le projet et vous devrez aussi y mettre du temps en devoir. Lors des périodes de classe, le professeur sera disponible pour répondre aux questions et vous aider en cas de besoin.

Programmez aussi leurs tests logiques prévus dans votre tableau de jeux d’essais et remplissez la dernière colonne.

4 : Procédure de développement de l'application

1. Créer un projet nommé "**projetFinal**";
2. Ajouter à votre projet un package nommé "**projetFinalPackage**";
3. Dans le fichier "\_\_init\_\_.py" de ce package, ajouter les éléments suivants:
   1. Un dictionnaire vide pour stocker la liste des employés nommée "**listeEmployes**". Ajouter à ce dictionnaire quelques éléments pour les tests;
   2. Un dictionnaire vide pour stocker l'historique des paies des employés nommée "**historiquePaies**";
   3. La fonction nommée "**isExists**". Cette fonction reçoit le numéro d'employé en paramètre et retourne "True" si l'employé se trouve dans la liste des employés et "False" dans le cas contraire.
   4. La fonction nommée "**ajouterEmploye**". Cette fonction reçoit en paramètre le numéro, le nom, le prénom, la fonction et le taux horaire de l'employé à ajouter au dictionnaire;
   5. La fonction nommée "**afficherEmployes**". Cette fonction affiche la liste des employés. Vous devez vous assurer que la liste n'est pas vide sinon lever une exception avec le message "Aucun employé à afficher";
   6. La fonction nommée "**supprimerEmploye**". Cette fonction reçoit en paramètre le numéro de l'employé à supprimer;
   7. La fonction nommée "**modifierEmploye**". Cette fonction reçoit en paramètre une liste qui contient le numéro de l'employé à modifier et les nouvelles valeurs du nom, prénom, fonction et le taux horaire;
   8. La fonction nommée "**calculerPaie**". Cette fonction calcule la paie de tous les employés et stocke les résultats dans le dictionnaire nommée "historiquePaies". Elle reçoit en paramètre la liste "listeHeures" et le numéro du mois concerné. Pour calculer le salaire net on suppose que les impôts représentent **20%** du salaire brut;
   9. La fonction nommée "**afficherPaies**". Cette fonction affiche la paie d'un employé ou de tous les employés pour un mois donné. Elle reçoit en paramètre le numéro de l'employé concerné et le numéro du mois. Si le numéro de l'employé est "00", elle affiche la paie de tous les employés;
   10. La fonction nommée "**afficherHistoriquePaies**". Cette fonction affiche historique des paies d'un employé. Elle reçoit en paramètre le numéro de l'employé concerné.

**Remarque** :

Une fois ces fonctions programmées, testées et validées, vous êtes prêts à développer le programme principal qui devrait inclure un menu et faire appel aux fonctions développées. C’est aussi ici que vous devez programmer les interactions avec l’utilisateur et la gestion d’erreurs.

**5 : Le développement du programme principal:**

1. Ajouter à votre projet un fichier qui représente votre programme principal nommé "**gestEntreprise**";
2. Ajouter à ce fichier les éléments suivants:
   1. Importer le package nommé "**projetFinalPackage**"
   2. Un dictionnaire vide pour stocker le nombre d'heures travaillées pour chaque employé, dont la clé est le numéro d'employé et le nombre d'heures travaillées par mois comme valeur. Cette liste sert pour calculer la paie mensuelle des employés. Exemple:

listeHeures = {

'E1' : 60,

'E2' : 65

}

* 1. Une fonction nommée "**menu**" qui affiche un menu à l'usager et retourne le choix de l'usager. Le menu doit avoir la forme :

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* menu de l'application \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1 : Ajouter un employé

2: Afficher la liste des employés

3 : Supprimer un employé

4 : Modifier un employé

5 : Préparer une paie

6 : Calculer une paie

7 : Afficher paies

8 : Afficher historique paies

0 : Quitter

Veuillez choisir une option :

**Remarque** : cette fonction force l'usager à donner un choix valide sinon elle affiche un message d'erreur. C’est-à-dire vous devez valider le choix de 'usager.

* 1. Dans une boucle infinie, appeler la fonction "menu" et recevoir le choix de l'usager;
  2. Traiter le choix de l'usager;

Si choix == 0 :

* + - * On quitte le programme;

Sinon si choix == 1 :

* + - * Lire les informations du nouvel employé
      * Assurez-vous que l'employé n'existe pas déjà
      * Appeler la fonction "**ajouterEmploye**"

Sinon si choix == 2:

* + - * Appeler la fonction "**afficherEmployes**"

Sinon si choix == 3 :

* + - * Lire le numéro de l'employé à supprimer
      * Assurez-vous que l'employé existe déjà
      * Appeler la fonction "**supprimerEmploye**"

Sinon si choix == 4 :

* + - * Lire le numéro de l'employé à modifier et les nouvelles informations
      * Assurez-vous que l'employé existe déjà
      * Appeler la fonction "**modifierEmploye**"

Sinon si choix == 5 :

* + - * Vider le dictionnaire liste "listeHeures"
      * Dans une boucle:
        + Lire le numéro de l'employé et le nombre d'heures travaillées du mois
        + Quitter si le numéro d'employé égale à "00"
        + Assurez-vous que l'employé existe déjà
        + Ajouter un élément à la liste "listeHeures"

Sinon si choix == 6 :

* + - * Lire le numéro du mois concerné
      * Appeler la fonction "**calculerPaie**" en lui envoyant en paramètres la liste "listeHeures" et le mois concerné

Sinon si choix == 7 :

* + - * Lire le numéro de l'employé concerné et le numéro du mois
      * Assurez-vous que l'employé existe déjà
      * Appeler la fonction "**afficherPaies**" en lui envoyant le numéro d'employé et le numéro du mois. Envoyer "00" comme numéro d'employé pour afficher la paie du mois de tous les employés

Sinon si choix == 8 :

* + - * Lire le numéro de l'employé concerné
      * Assurez-vous que l'employé existe déjà
      * Appeler la fonction "**afficherHistoriquePaies**".

**Remarque** :

À la fin du développement du programme principal, vous devez effectuer le plus de tests possibles pour vous assurer que le programme ne se plante jamais et a toujours le comportement attendu, peu importe ce que l’utilisateur entre comme données.

6 : Présentation du projet

Une fois le développement terminé, vous devrez faire la présentation de votre projet, en classe, durant la semaine d’examen : **du 14 au** **20 décembre au local 2.708.**

**Contenu de la présentation :**

* + **Démonstration du bon fonctionnement du programme**
  + Explications détaillées des difficultés rencontrées et décrire la solution trouvée
  + Mentionner et expliquer les trouvailles faites et des ajouts développés dans votre application

La présentation de votre projet, doit se faire de façon **professionnelle** et devra durer maximum **15 minutes**.

Échéancier

|  |  |  |
| --- | --- | --- |
| **Document** | **Description** | **Date d’échéance** |
| 1. Développement des procédures et fonctions | Code des **procédures et fonctions** et leurs **tests**,(aucunes demandes d’entrées de l’utilisateur).  **Tableau de jeux d’essais** mis-à-jour avec les tests des fonctions. | Jeudi 2 décembre |
| 2. Projet PyCharm complet | Dossier compressé qui contient le dossier **projet au complet.**  **Tableau de jeux d’essais** mis-à-jour avec les essais du programme principal. | Jeudi 9 décembre |
| 3. Présentation du projet | **Présentation du projet au professeur.**  **L'organisation des présentations sera envoyée par le professeur.** | Semaine d’évaluation  Du 14 au 20 décembre |

7 : Critères d’évaluation

**Analyser le problème 10%**

* Tableau de démarche :
  + Entrées (3)
  + Sorties (3)
  + Traitements (4)

**Traduire l’algorithme dans le langage de programmation 50%**

* Fonctionnement du programme, comme prévu :
  + Procédures et fonctions (20)
  + Programme principal (10)
* Efficacité, optimisation du code (10)
* Respect des normes (5)
* Documentation, commentaires (5)

**Déboguer le code 20%**

* Validation adéquate des données (8)
* Gestion adéquate des exceptions (8)
* Présentation claire des solutions aux problèmes rencontrés (4)

**Appliquer le plan de tests fonctionnels 20%**

* Tableau de jeux d’essai
  + Données d’entrées (8)
  + Résultats attendus (8)
  + Résultats obtenus (4)

**Le projet synthèse a un poids de 60% sur la note finale**