

Secteur : **Digital & IA**

Manuel des travaux pratiques

M102 : Acquérir les bases de l'algorithmique

1^{ère} Année

Filière :

Développement
Digital
(Tronc commun)



Remerciements

La DRIF remercie les personnes qui ont contribué à l'élaboration du présent document :

Équipe de conception :

BOUDIAF Saida *Digital learning manager/
Project manager*

MIHOUBI Fattoum, *Cheffe de projet pédagogique/
Ingénieure pédagogique*

CROUZOULON Jonathan, *Directeur pédagogique/
Chef de projet pédagogique*

Équipe de rédaction :

JEMEL Meriam, *Assistant Professor in Computer Science*

Équipe de lecture :

RAHMANI Abdelhak, *Formateur Animateur au CDC Digital & IA*

LAOUIJA Soukaina, *Formatrice Animatrice au CDC Digital & IA*

EL KHATABI GHIZLANE, *Formatrice Animatrice au CDC Digital & IA*

Les utilisateurs de ce document sont invités à communiquer à la DRIF et au CDC Digital & IA toutes les remarques et suggestions afin de les prendre en considération pour l'enrichissement et l'amélioration de ce module.



SOMMAIRE

01 - MODELISER UN PROBLEME

Analyser un problème
Identifier les approches d'analyse d'un problème

02 - FORMULER UN TRAITEMENT

Reconnaître la structure d'un algorithme
Reconnaître les bases
Structurer un algorithme
Structurer les données

03 - PROGRAMMER EN PYTHON

Transformer une suite d'instructions algorithmiques en suite d'instructions Python
Manipuler les données

04 - DÉPLOYER UNE SOLUTION PYTHON

Déboguer le code Python
Déployer une solution Python



PARTIE 1

MODELISER UN PROBLEME

Dans ce module, vous allez :

- Analyser un problème
- Identifier les approches d'analyse d'un problème



06 heures

Activité 1

Analyser un problème

Compétences visées :

- Bonne compréhension du contexte d'un problème
- Repérage des entrées du problème
- Repérage des sorties du problème
- Distinction des sorties calculées et des sorties non-calculées
- Distinction des différents types de traitements

Recommandations clés :

- Faire une bonne lecture de l'énoncé du problème proposé et se focaliser sur le « quoi faire ? » sans se préoccuper du « comment faire ? »



1,5 heures



CONSIGNES

1- Pour le formateur

- Demander de dégager le contexte du problème
- Demander de dégager la liste des entrées/sorties du problème à partir de la description du problème
- Demander de distinguer entre les résultats calculés ou non
- Demander de déterminer les différents types de traitements

2- Pour l'apprenant

- Dégager le contexte d'un problème
- Dégager la liste des entrées/sorties du problème à partir de la description du problème
- Distinguer entre les résultats calculés ou non
- Distinguer entre les différents types de traitements



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Comprendre la contexte du problème ?
 - Décortiquer le problème en des sous-problèmes ?
 - Dégager les entrées/sorties d'un problème ?
 - Déterminer le type de traitement nécessaire pour résoudre un problème



Activité 1 : Analyser un problème



Exercice 1

- Mr. Ali est un fermier qui fabrique du fromage artisanal. Il achète le lait aux agriculteurs de son village à 6Dh le litre. Il faut 10 litres de lait pour fabriquer 1kg de fromage. Il vend son fromage à 60Dh le kg.
- Vous disposez de la quantité de lait achetée par Mr. Ali pour tout un mois, on souhaiterait calculer :
 1. La quantité (en kg) de fromage fabriquée.
 2. Le net à gagner mensuel du fermier sachant que Mr. Ali a vendu tout le fromage fabriqué et qu'il a payé 1000 Dh le m² pour la location d'un stand de 3 m² sur la place du marché du village et des frais de transport de 500 Dh par jour sachant qu'il se rend au marché 8 fois par mois.

Activité 1 : Analyser un problème

A vous de jouer

1. Donner une description du contexte du problème.
2. Discuter les choix proposés.

Il s'agit d'un problème (choix proposé) de : a. Affichage b. Classement c. Calcul	
Le nom du fermier est une information importante : a. Oui b. Non	
Le nombre de litres de lait pour fabriquer 1kg de fromage est une information importante : a. Oui b. Non	
Ce problème peut être décomposé en n sous-problème(s) a. n=1 b. n=2. c.n=3	

1. Donner une liste des entrées du problème.
2. Déterminer le(s) résultat(s) à trouver.
3. Donner les autres sorties du problème (celles qui doivent être calculées pour déterminer le résultat final ou les résultats finaux).

Activité 1 : Analyser un problème



Exercice 2

- On voudrait faciliter la gestion d'un magasin de vente de prêt à porter.
- Les articles du magasin possèdent : une référence (un code), un libellé et un prix unitaire hors taxe. A partir d'une quantité d'un article achetée à un prix hors taxe, on souhaite établir les factures de 3 clients dont on connaît leurs noms. On suppose que chacun d'eux a acheté un seul article.
- La facture fera apparaître le nom du client et le montant toute taxe (TTC), sachant qu'on applique un taux de TVA de 15% et le client bénéficie d'une remise de 2% sur le montant de ces achats.
- On souhaite aussi attribuer à chacun de ces clients une catégorie sachant que le magasin classe ses clients dans 3 catégories selon des critères :
 - ✓ Catégorie 'A' : le client est titulaire d'une carte de fidélité de moins de 2 ans.
 - ✓ Catégorie 'B' : le client est titulaire d'une carte de fidélité de plus de 2 ans.
 - ✓ Catégorie 'C' : le client est titulaire d'une carte de fidélité de plus de 2 ans et il a plus de 5000 points de fidélité

Activité 1 : Analyser un problème



A vous de jouer

1. Donner le contexte du problème
2. Donner une liste des entrées du problème.
3. Déterminer le(s) résultat(s) à trouver.
4. Donner les autres sorties du problème (celles qui doivent être calculées pour déterminer le résultat final ou les résultats finaux).
5. Identifier les différents types de traitements appliqués pour aboutir aux résultats finaux.
6. On considère que le client a acheté N produits, proposer un traitement récursif pour calculer le prix Hors Taxe de tous les produits achetés (TotalPrixHT) par un seul client

Activité 1

Correction

Exercice 1

Demande une description du contexte :

Dans un contexte de production (fabrication de fromage à partir de lait), le problème consiste à calculer la quantité de fromage fabriqué par un fermier ainsi que le net à gagner à partir d'un ensemble de données fournies

Présentation du tableau suivant et discussion des choix proposés :

Il s'agit d'un problème de : a. Affichage b. Classement c. Calcul	c
Le nom du fermier est une information importante. a. oui b. non	b
Le nombre de litres de lait pour fabriquer 1kg de fromage est une information importante. a. oui b. non	a
Ce problème peut être décomposé en sous problème (s). a. 1 b. 2 c. 3	b

Demande d'une liste des entrées du problème :

1. Prix Unitaire d'un litre de lait : 6Dh
2. Quantité de lait pour fabriquer 1kg de fromage : 10 litres
3. Prix de vente d'un kg de fromage : 60Dh
4. Quantité de lait achetée par mois : donnée
5. Mr. Ali a vendu tout le fromage fabriqué !!!

Exercice 1 (suite)

3. Charge de location d'un stand de 3m^2 : 1000 Dh le m^2
4. Les frais de transport (8 fois par mois) : 500 Dh par jour

4. Questionner sur le(s) résultat(s) à déterminer

- a. la quantité (en kg) de fromage fabriquée
- b. le net à gagner mensuel du fermier

5. Discuter sur les autres sorties du problème

Pour calculer le net à gagner mensuel, on aura besoin de calculer :

- le revenu mensuel
- le coût du lait par mois
- les autres charges mensuelles.

Exercice 2

1. liste des entrées du problème :

1. Nom du client1/Client2/client3
2. Code de l'article acheté par client1/Client2/client3
3. Libellé de l'article acheté par client1/Client2/client3
4. Prix hors taxe de l'article par client1/Client2/client3
5. Quantité de l'article acheté par client1/Client2/client3
6. chaque client a acheté un seul article
7. Taux tva=15%
8. Remise=2%
9. Année fidélité du client1/client2/client3
10. Nombre de point de fidélité pour client1/client2/client3

2. Le résultat à déterminer

- a. Montant toute taxe (prixTTC) après la remise pour client1/Client2/client3
- b. Catégorie du client1/Client2/client3

3. Les autres sorties du problème

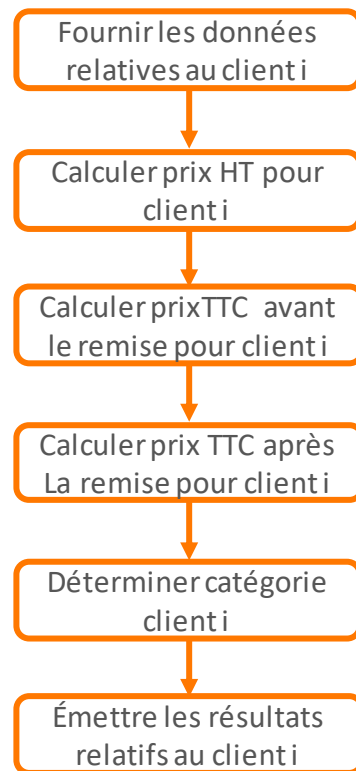
Pour chaque client, afin de calculer le prixTTC après la remise, on aura besoin de calculer :

1. Le prix HT
2. Le prixTTC avant la remise

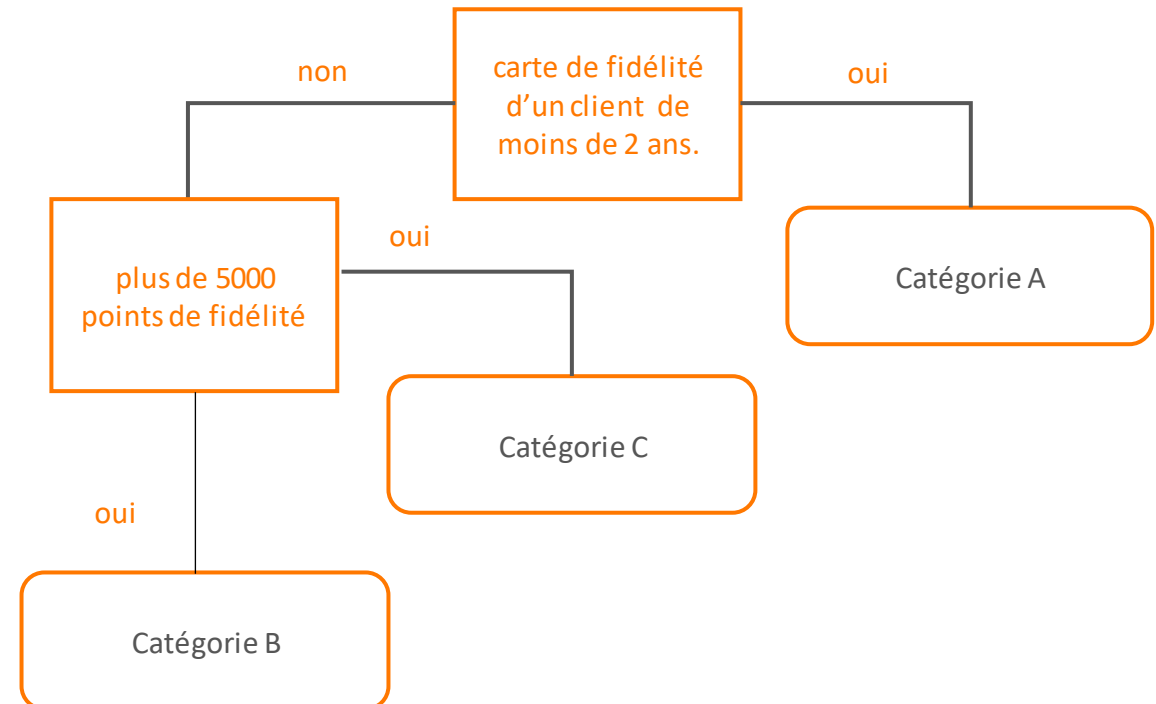
Exercice 2 (suite)

Les différents types de traitements appliqués pour aboutir aux résultats finaux.

1. Traitement séquentiel : Cas d'un seul client i



2. Traitement Conditionnel (Déterminer catégorie Client i)



Exercice 2 (suite)

3. Traitement itératif

Fournir les données relatives au client i

Calculer prix HT pour client i

Calculer prixTTC avant le remise pour client i

Calculer prix TTC après la remise pour client i

Déterminer catégorie client i

Émettre les résultats relatifs au client i

Répéter pour i=1, i=2 et i=3

4. Traitement récursif

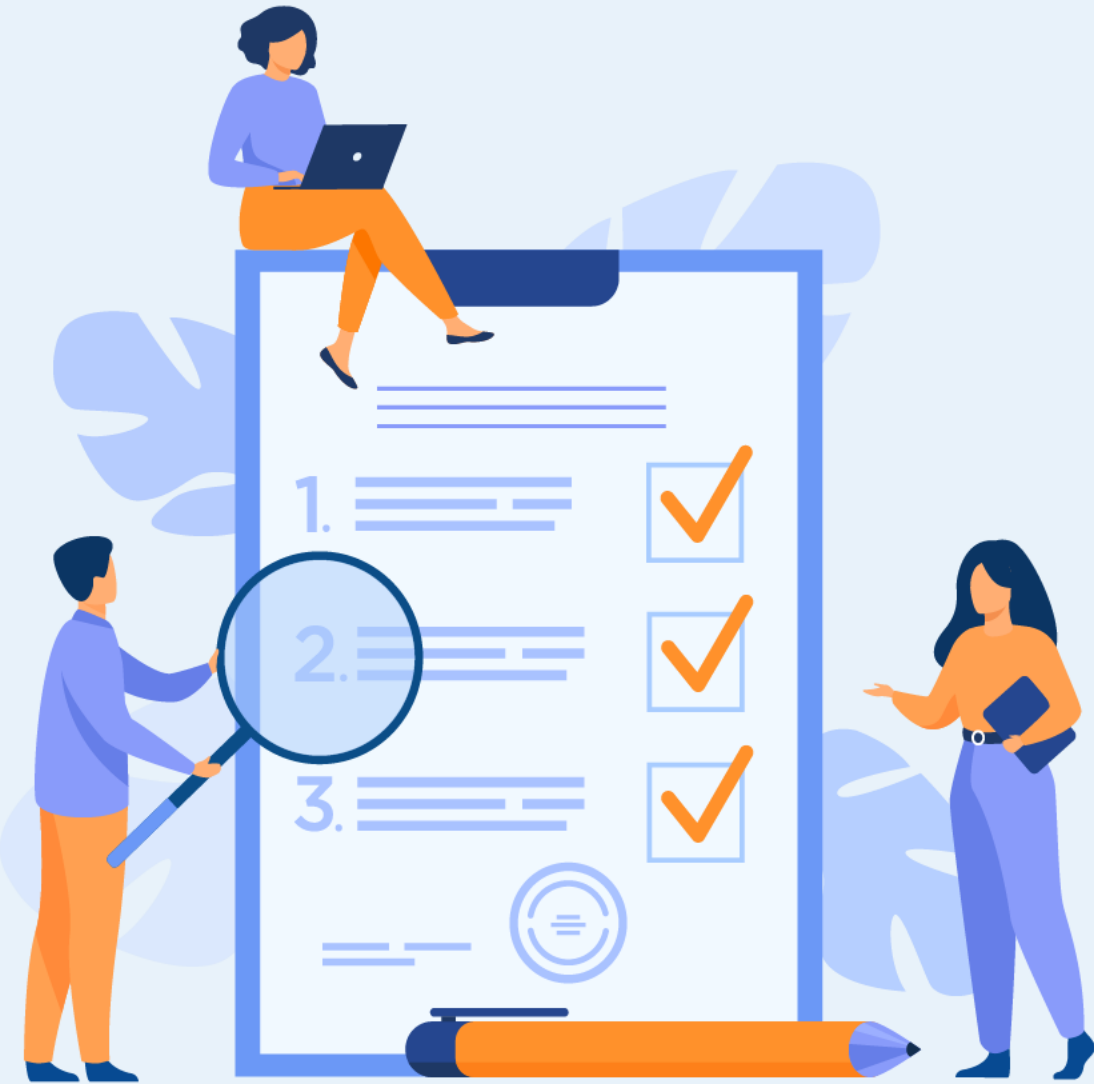
Si on considère N le nombre de produits achetés par un client

- Si $N=1$ alors

$\text{TotalPrixHT}(N) = \text{prixTTC}_{\text{produitN}}$ (prix TTC du produit acheté)

- Si $N>1$ alors

$\text{TotalPrixHT}(N) = \text{prixTTC}_{\text{produitN}} + \text{TotalPrixHT}(N-1)$



Activité 2

Identifier les approches d'analyse d'un problème

Compétences visées :

- Mettre en pratique les approches d'analyse d'un problème

Recommandations clés :

- Bien comprendre le problème et le décomposer en sous-problèmes avant de procéder à une approche pour le traiter



1,5 heures

CONSIGNES

1- Pour le formateur

- Demander de décortiquer le problème en sous-problèmes pour donner une analyse ascendante ou descendante

2- Pour l'apprenant

- Décortiquer le problème en sous-problèmes pour donner une analyse ascendante ou descendante

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Décortiquer le problème en des sous-problèmes?
 - Proposer une analyse ascendante d'un problème ?
 - Proposer une analyse descendante d'un problème ?



Activité 2 :

Approches d'analyse d'un problème



Exercice1

- Hakim est un peintre en bâtiment. Il a remporté un marché pour peindre l'édifice d'un grand immeuble. Le bâtiment est composé de 6 étages. Chaque étage est facturé par Hakim à 6000 Dh. Pour peindre un étage, Hakim achète 10 kg de peinture, pour 200 Dh le Kg. Hakim emploie également 3 jeunes apprentis pour l'aider. Chaque apprenti est payé 10 Dh par heure. Une journée de travail est d'une durée de 6 heures et il faut 4 jours à l'équipe de peintre pour finir un étage.

On souhaite calculer :

- Le prix encaissé par Hakim pour peindre tout l'immeuble.
- le net à gagner mensuel de Hakim.

A vous de jouer

- Dégager les sous-problèmes à partir du problème de l'énoncé
- Pour chacun des sous-problèmes, donner une analyse descendante ascendante à travers un schéma

Activité 2 : Correction

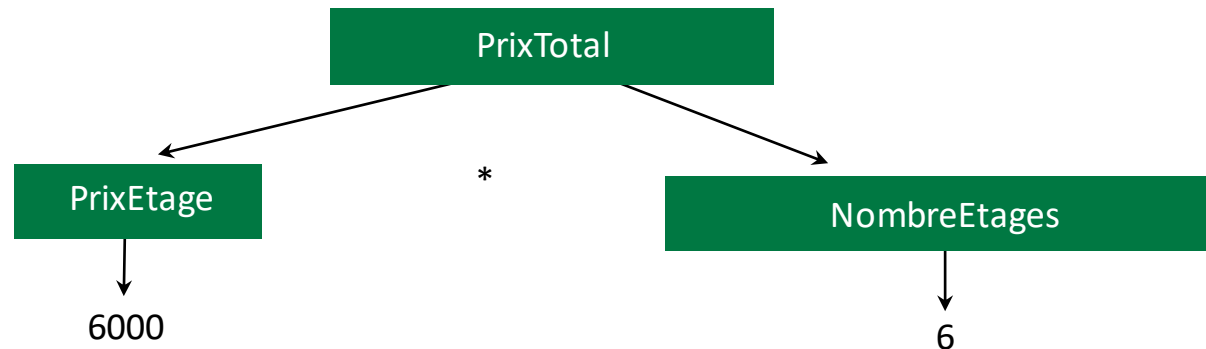
Exercice 1

1. Les sous-problèmes dégagés à partir du problème de l'énoncé.

- Appeler les stagiaires à dégager les sous-problèmes à partir du problème de l'énoncé.
- Deux sous-problèmes :
 1. Le prix encaissé par Hakim pour peindre tout l'immeuble : **PrixTotal**
 2. Déterminer le net à payer mensuel : **NetMois**

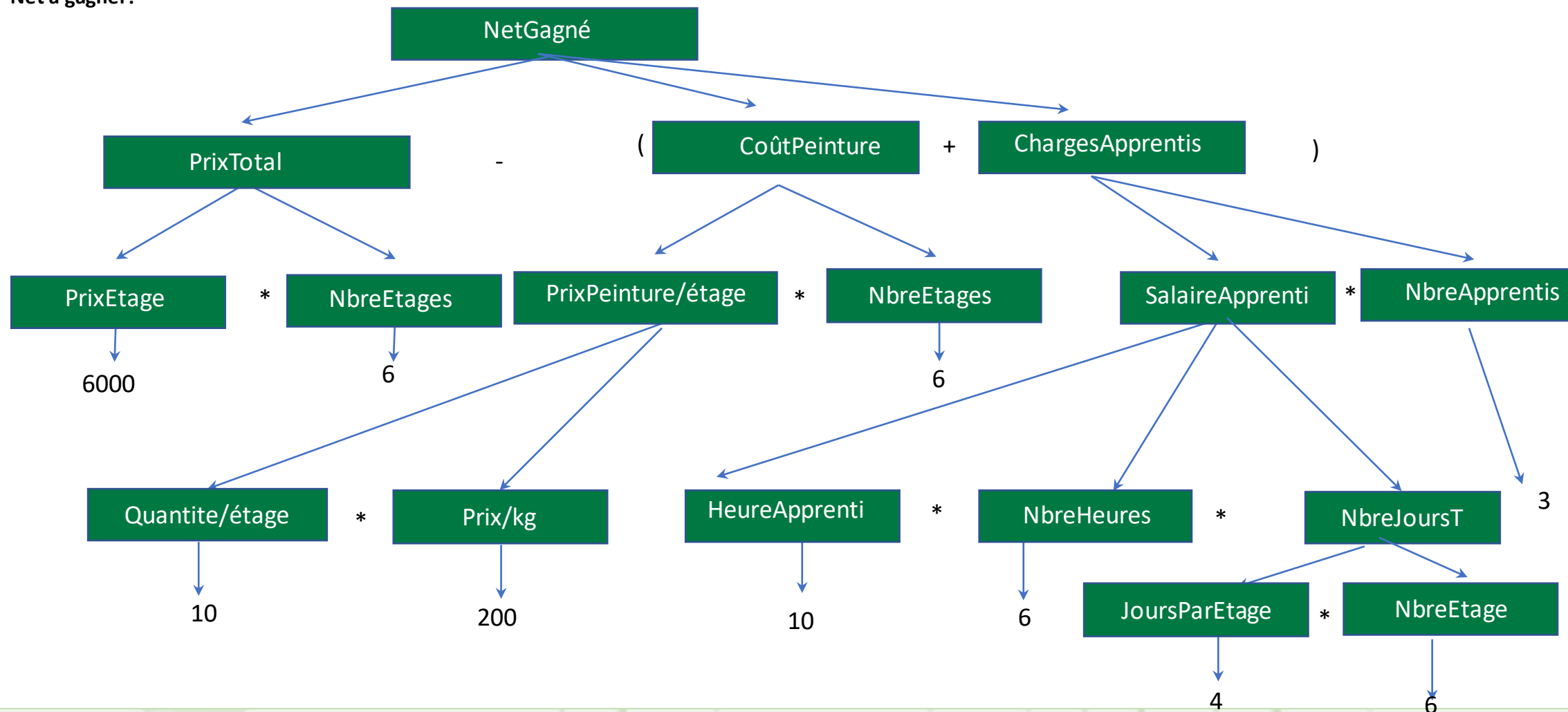
2. Une analyse descendante des sous-problèmes à travers un schéma

PrixTotal= ?



Activité 2 : Correction

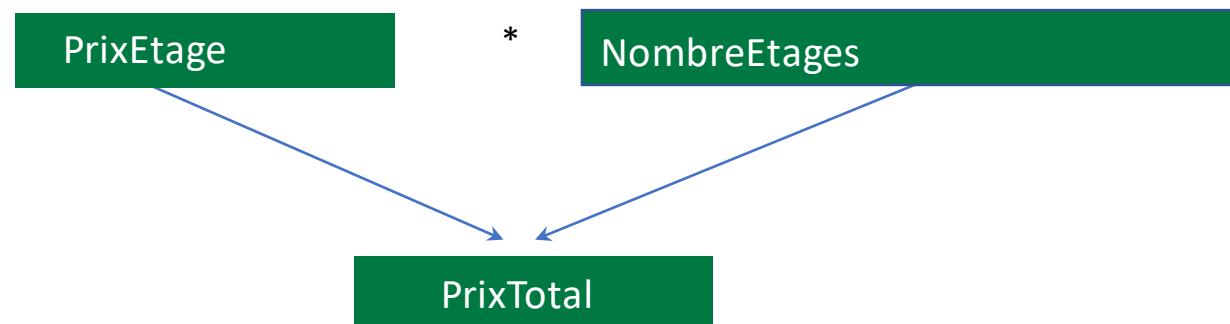
Net à gagner?



Activité 2 : Correction

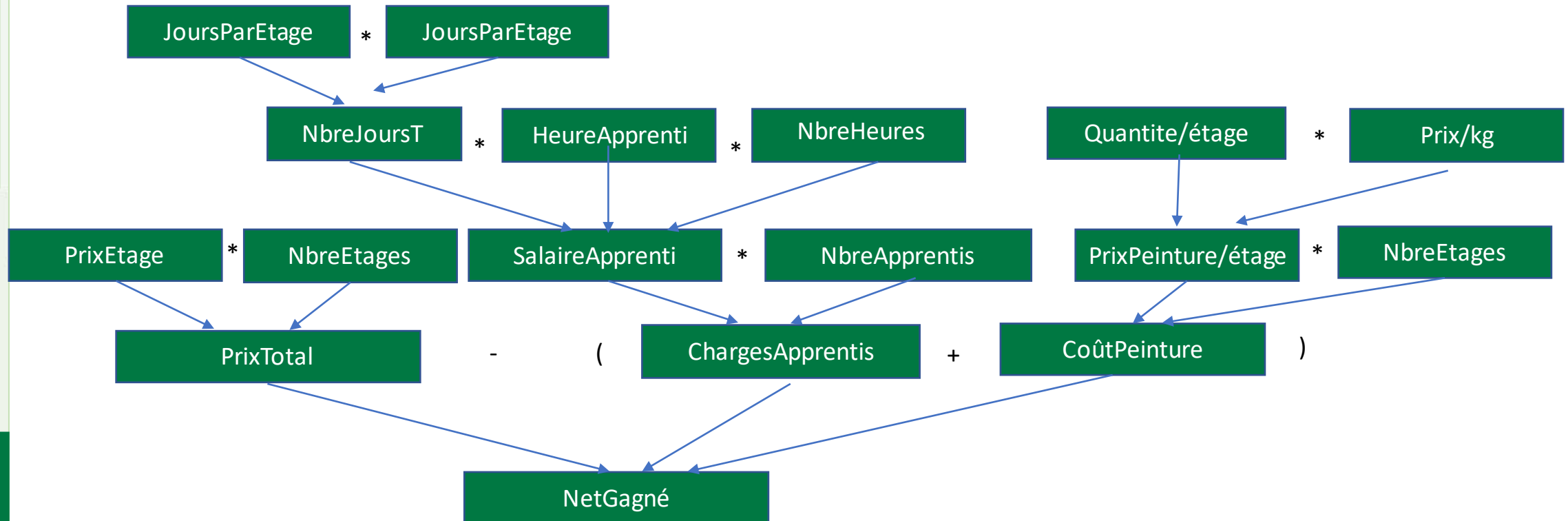
Une analyse ascendante des sous-problèmes à travers un schéma

PrixTotal = ?



Activité 2 : Correction

Net à gagner ?





PARTIE 2

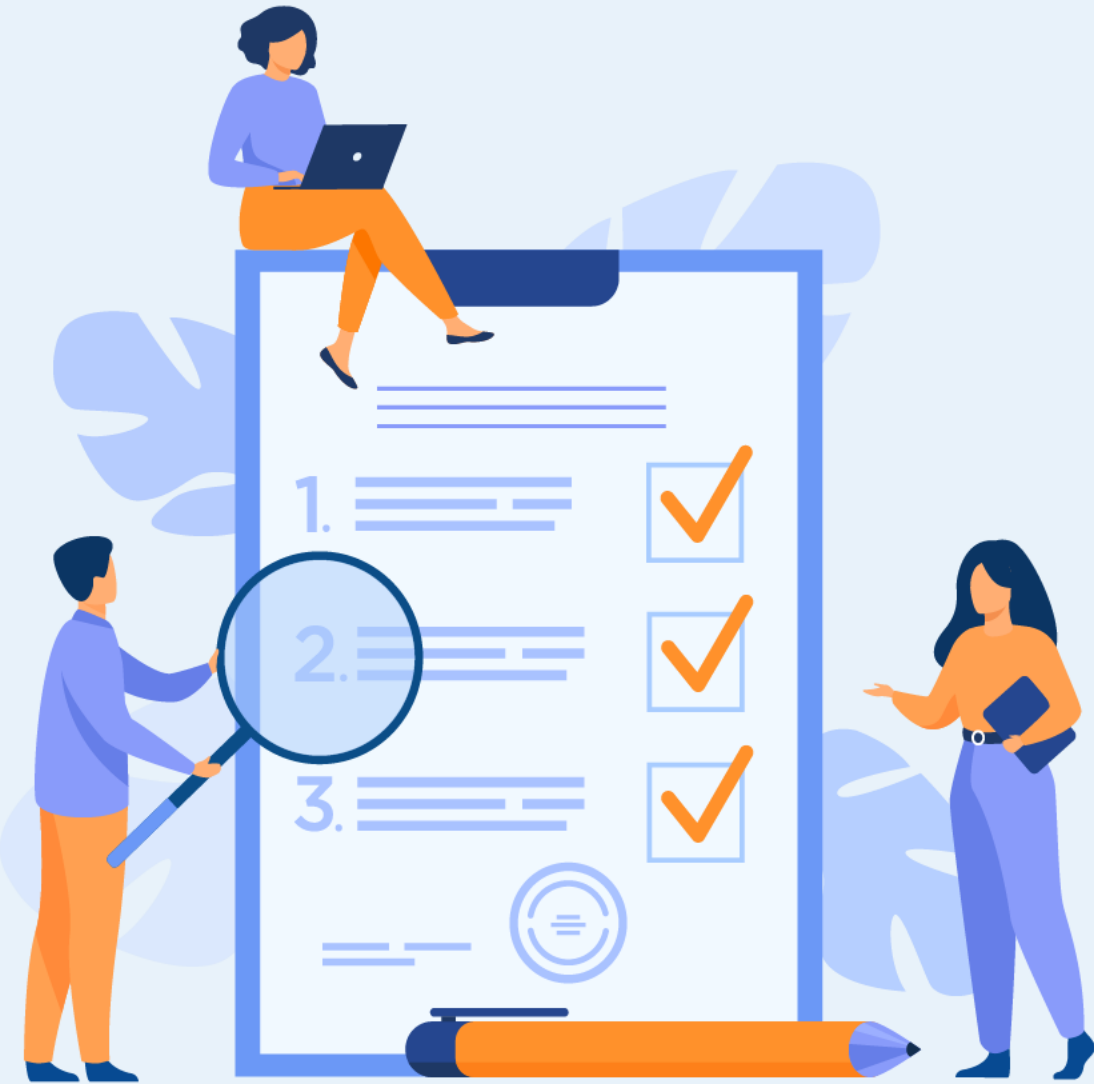
FORMULER UN TRAITEMENT

Dans ce module, vous allez :

- Reconnaître la structure d'un algorithme
- Reconnaître les bases
- Structurer un algorithme en procédures et fonctions
- Reconnaître les structures de données



20 heures



Activité 1

Reconnaitre la structure d'un algorithme

Compétences visées :

- Manipulation de la déclaration des variables, des constantes et de la définition de leurs types
- Manipulation des expressions arithmétiques et des expressions logiques
- Maîtrise de la structuration d'un algorithme

Recommandations clés :

- Bonne compréhension du problème à traiter avant de procéder à l'écriture de l'algorithme.



1 heure

CONSIGNES

1- Pour le formateur

- Rappeler les notions de variables et constantes ainsi que leurs types
- Rappeler l'ordre selon lequel se déroule une opération arithmétique/logique
- Demander d'appliquer l'ordre de priorité pour évaluer les différentes opérations
- Rappeler la structure d'un algorithme

2- Pour l'apprenant

- Définir les notions de variable/constante
- Définir l'ordre selon lequel se déroule une opération arithmétique/logique
- Appliquer l'ordre de priorité pour évaluer les différentes opérations
- Définir la structure d'un algorithme



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Différencier les variables et les constantes ?
 - Définir les types adéquats des variables/constantes ?
 - Définir les notions de priorités des opérateurs ?
 - Evaluer une expression logique et arithmétique ?
 - Définir la structure d'un algorithme ?



Activité 1 :

Reconnaitre la structure d'un algorithme

Exercice 1

1. Quel est l'ordre de priorité des différents opérateurs de l'expression suivante :

$$((3 * a) - x^2) - (((c - d) / (a / b)) / d)$$

2. Sachant que $a = 4$, $b = 5$, $c = -1$ et $d = 0$, évaluer les expressions logiques suivantes :

$$(a < b) \text{ ET } (c \geq d)$$

$$\text{NON } (a < b) \text{ OU } (c \neq d)$$

Exercice 2

Donner toutes les raisons pour lesquelles l'algorithme suivant est incorrect :

1. Incorrect
2. y : Entier
3. z : Réel
4. Début
5. $z \leftarrow x + 2$
8. $y \leftarrow 5y + 3$
9. Fin

Activité 1 :

Reconnaitre la structure d'un algorithme



Exercice 3

On souhaite écrire un algorithme qui permet de calculer la somme, la moyenne, la valeur la plus grande, la valeur la plus petite et le nombre de valeurs positives d'un nombre d'entiers saisis. Le nombre d'entier à saisir est une donnée.

1. Identifier les entrées/sorties du problème.
2. Lister les variables de l'algorithme à proposer ainsi que leurs types.
3. Ecrire l'algorithme correspondant.

Exercice 4

On souhaite écrire un algorithme permettant de calculer la facture d'un client (factureC) sachant qu'il est possible d'acheter 2 articles a1 et a2.

factureC est calculée à partir :

- du prix HT d'un article acheté.
- de la quantité d'un article acheté.
- du taux de TVA fixé à 10%
- d'une remise de 5% accordée.

Activité 1 :

Reconnaitre la structure d'un algorithme



Exercice 4 (suite)

1. Identifier les entrées/sorties du problème.
2. Lister les variables de l'algorithme à proposer ainsi que leurs types.
3. Lister les constantes de l'algorithme à proposer ainsi que leurs déclarations.
4. A partir des variables et des constantes listées, donner l'expression permettant d'évaluer la facture d'un client.
5. Evaluer cette expression sachant que:
 - Le client a acheté 5 exemplaires de a1 ayant un prix HT=5Dh et 3 exemplaires de a2 ayant un prix HT=7Dh .
6. Ecrire l'algorithme correspondant.

Activité 1 : Correction

Exercice 1

1. L'ordre de priorité des différents opérateurs de l'expression suivante :

$$((3 * a) - x ^ 2) - (((c - d) / (a / b)) / d)$$

- Rappeler l'ordre selon lequel se déroule les opération arithmétiques :

Priorité	Opérateurs
1	- signe négatif
2	() parenthèses
3	^ puissance
4	* et / multiplication et division
5	+ et - addition et soustraction

- Demander d'appliquer l'ordre de priorité des opérations arithmétiques

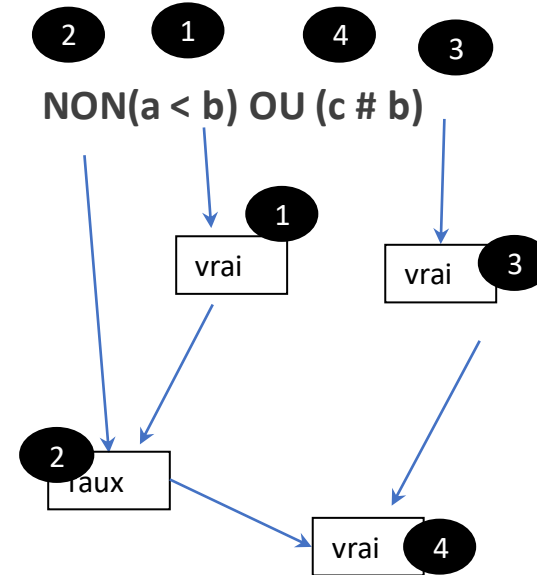
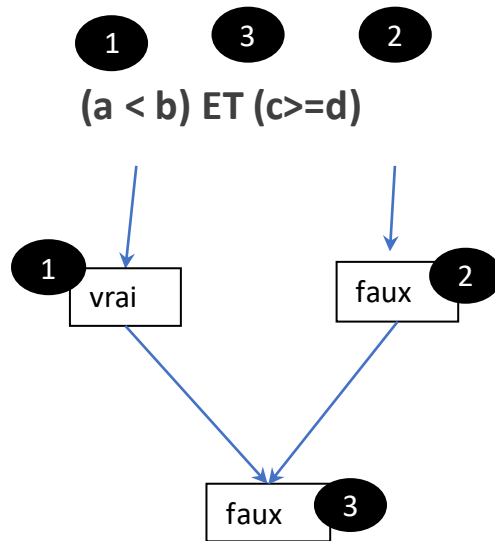
$$((3 * a) - x ^ 2) - (((c - d) / (a / b)) / d)$$

1 3 2 8 4 6 5 7

Activité 1 : Correction

Exercice 1 (suite)

2. Pour $a = 4$, $b = 5$, $c = -1$ et $d = 0$, évaluation des expressions logiques suivantes :



Activité 1 : Correction

Exercice 2

3. Les raisons pour lesquelles l'algorithme suivant est incorrect :

1. Incorrect
2. y : Entier
3. z : Réel
4. Début
5. $z := x + 2$
8. $y := 5y + 3$
9. Fin

- Cet algorithme est incorrect pour plusieurs raisons :
 - Ligne 2 : La déclaration des variables commence par le mot « Var »
 - Ligne 5 : La valeur de x est indéterminée
 - Ligne 8 : Il faut écrire $5*y$ et non $5y$

Exercice 3

1. Entrées du problème :

- N: le nombre des entiers à saisir
- Nb: l'entier à saisir à chaque fois
- Sortie du problème :
 - Somme: somme des entiers
 - Moyenne: la moyenne des entiers
 - Max: la valeur max des entiers
 - Min: la valeur min des entiers
 - Nb_pos: le nombre des entiers positifs

2. Variables du problème :

Exercice 3 (suite)

- **Moyenne**
- **Var N, Somme, Max, Min, Nb_pos, i, Nb: entier**
- **Var Moyenne : réel**
- **Début**
- $Nb_pos \leftarrow 0$
- **Ecrire** ("Saisir le nombre d'entier à entrer")
- **Lire** (N)
- **Ecrire** ("Saisir un nombre")
- **Lire** (Nb)
- $Max \leftarrow Nb$

```
Si (Nb >= 0) Alors  
    Nb_pos ← Nb_pos + 1
```

```
FinSi
```

```
Pour i de 2 à N faire
```

```
    Ecrire ("Saisir un nombre")
```

```
    Lire( Nb)
```

```
    Si (Nb > Max) Alors
```

```
        Max ← Nb
```

```
    FinSi
```

```
    Si (Nb < Min) Alors
```

```
        Min ← Nb
```

```
    FinSi
```

```
Somme ← Somme + Nb
```

```
Si (Nb >= 0) Alors
```

```
    Nb_pos ← Nb_pos + 1
```

```
FinSi
```

```
FinPour
```

```
Moyenne ← Somme \ N
```

```
Ecrire ("La valeur la plus grande = ", Max)
```

```
Ecrire ("La valeur la plus petite = ", Min)
```

```
Ecrire ("La Somme = ", Somme)
```

```
Ecrire ("Moyenne = ", Moyenne);
```

```
Ecrire ("Nombre d'entiers positifs = ", Nb_pos)
```

```
fin
```

Activité 1 : Correction



Exercice 4

1. Entrées du problème :

- prixHTA1 : prixHT de a1
- QteA1 : Quantité de a1
- prixHTA2 : prixHT de a2
- QteA2: Quantité de a2
- tauxTVa: taux tva
- Remise: remise accordée

Sortie du problème: facture du client

2. Les Variables du problème sont :

prixHTA1: réel, QteA1: entier, prixHTA2: réel, QteA2: entier, factureC: réel

3. Les constantes du problème sont :

Const tva= 0.1: réel

Const remise=0.05: réel

Activité 1 : Correction

Exercice 4 (suite)

4. $\text{factureC} = (\text{prixHTA1} * (1 + \text{tva})) * (1 - \text{remise}) * \text{qteA1} + (\text{prixHTA2} * (1 + \text{tva})) * (1 - \text{remise}) * \text{qteA2}$
5. $\text{factureC} = (5 * (1 + 0.1)) * (1 - 0.05) * 5 + (3 * (1 + 0.1)) * (1 - 0.05) * 2 = 26.125 + 6.27 = 32.395$
6. Algorithme correspondant :

factureClient

Const

tva= 0.1: réel

remise=0.05: réel

Var

prixHTA1, prixHTA2: réel

QteA1, QteA2: entier

factureC: réel

Début

$\text{factureC} \leftarrow (\text{prixHTA1} * (1 + \text{tva})) * (1 - \text{remise}) * \text{qteA1} + (\text{prixHTA2} * (1 + \text{tva})) * (1 - \text{remise}) * \text{qteA2}$

Ecrire ("La facture du client est " + factureC)

Fin

Activité 2

Reconnaitre les bases

Compétences visées :

- Manipulation des instructions d'affectation et les instructions d'E/S
- Maîtrise des structures alternatives (utilisation et syntaxe)
- Maîtrise des structures itératives (utilisation et syntaxe)

Recommandations clés :

- Bonne compréhension du problème à traiter avant de procéder à l'écriture de l'algorithme
- Bonne identification du type de structures à utiliser pour proposer une solution



05 heures



CONSIGNES

1- Pour le formateur

- Rappeler la structure générale d'un algorithme
- Questionner sur les variables de l'algorithme et leurs types
- Questionner sur les instruction E/S
- Questionner sur la formule pour résoudre le problème
- Guider les stagiaires à définir la structure à utiliser pour résoudre le problème (séquentielle, itérative, répétitive)

2- Pour l'apprenant

- Définir la structure générale d'un algorithme
- Déterminer les variables de l'algorithme et leurs types
- Définir les instruction E/S
- Définir la formule pour résoudre le problème
- Définir les structures à utiliser pour résoudre le problème



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe

4 - Critères de réussite

- Le stagiaire est-il capable de:
 - Définir les instructions d'affectation et les instructions E/S ?
 - Définir les structures itératives ?
 - Définir les structures répétitives



Activité 2 : Reconnaitre les bases

Exercice 1

Une école désire mettre au point un programme qui permet à chaque élève d'entrer ses notes et de calculer ainsi sa moyenne. Ecrire un algorithme qui lit donc 4 notes N_1 , N_2 , N_3 et N_4 et calcule et affiche la moyenne obtenue. Les coefficients respectifs C_1 , C_2 , C_3 , C_4 sont définies par l'école ($C_1 : 1$, $C_2 : 1,5$, $C_3 : 2$, $C_4 : 1$)

Exercice 2

Un atelier découpe des pièces métalliques ayant la forme suivante :



Ecrire un algorithme qui lit les valeurs de m et l , calcule la surface de la pièce correspondante et son périmètre, avant de les afficher.

Exercice 3

Écrire un algorithme qui lit un entier $N > 0$ puis saisit une suite de N quadruplets (un quadruplet est une suite de 4 entiers compris entre 0 et 20). Pour chacun des quadruplets, l'algorithme doit afficher le minimum et le maximum des 4 nombres. Enfin, on affichera le plus petit et le plus grand de tous les nombres.

Exercice 4

Écrire un algorithme qui étant donnés deux entiers représentant un mois et une année, affiche le nombre de jours du mois de cette année. On prendra en considération l'année bissextile (divisible par 4 et ne se termine pas par 00, ou elle se termine par 00 et son quotient par 100 est divisible par 4).

Activité 2 :

Reconnaitre les bases

Exercice 5

Écrire un algorithme qui lit un entier positif, calcule et affiche sa factorielle.

On rappelle que :

$$0! = 1$$

$$n! = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1 \quad \text{pour tout } n \neq 0$$

Exercice 6

Écrire un algorithme qui lit 2 entiers positifs a et b calcule et affiche leur plus grand commun diviseur (pgcd).

Vous pouvez utiliser l'algorithme d'Euclide qui est basé sur le principe suivant :

$$\text{pgcd}(a, b) = a \quad \text{si } b = 0$$

$$\text{pgcd}(a, b) = \text{pgcd}(b, a \bmod b) \quad \text{si } b \neq 0$$

Activité 2 :

Reconnaitre les bases



Exercice 7

Un organisme de location de voiture propose la formule de location suivante:

- pour les 100 premiers kilomètres : tarif t1 au km,
- pour les kilomètres de 101 à 1000 : tarif t2 au km,
- au-delà de 1000 kilomètres : tarif t3 au km

Pour cette formule de location, il convient d'ajouter une assurance comptabilisée par jour et dont le montant est donné.

1. Ecrire un algorithme qui lit le nombre total de kilomètres et le nombre de jours de location ainsi que les tarifs t1, t2, t3 puis calcule et affiche le coût de tarification pour un forfait au kilomètre.

Pendant la période de fin d'année, cet organisme propose aussi un forfait journalier de location qui consiste à un kilométrage illimité au prix t4 par jour. Pour cette période, l'assurance est comptabilisée comme les autres périodes de l'année.

2. Apporter les modifications nécessaires sur l'algorithme proposé précédemment pour lire le tarif t4, calculer et afficher le coût de tarification pour un forfait journalier
3. Apporter les modifications nécessaires pour indiquer au client le forfait le plus avantageux.

Activité 2 : Correction

Exercice 1

- **Rappeler la structure générale d'un algorithme**
- **Questionner sur les variables de l'algorithme et leurs types**
 - Var N1, N2, N3, N4, M : réel
 - C1, C2, C3, C4 : entier
- **Questionner sur les instruction E/S**
 - Lire(variable) et écrire (variable)
- **Questionner sur la formule pour le calcul de la moyenne**
 - $M := ((N1 * C1) + (N2 * C2) + (N3 * C3) + (N4 * C4)) / (C1 + C2 + C3 + C4)$

Moyenne

Var N1, N2, N3, N4, M : réel

C1, C2, C3, C4 : entier

Début

Lire(N1)

Lire(N2)

Lire(N3

Lire(N4)

C1 := 1, C2 := 1,5, C3 := 2, C4 := 1

$M := ((N1 * C1) + (N2 * C2) + (N3 * C3) + (N4 * C4)) / (C1 + C2 + C3 + C4)$

écrire("la moyenne est :", M)

Fin

Activité 2 : Correction

Exercice 2

Un atelier découpe des pièces métalliques ayant la forme suivante :



Rappeler la structure générale d'un algorithme

Questionner sur les variables de l'algorithme

Var m, l, s, p, r : réel

Questionner sur les formules pour le calcul de la surface et du périmètre de la Figure :

$$r := m / 2$$

$$p := 2 * l + (2 * r * 3.14)$$

$$s := (l * m) + (3.14 * r * r)$$

Rappel : Surface cercle = $3.14 * r^2$

Périmètre cercle = $2 * r * 3.14$

Mesures

Const Pi=3.14

Var m, l, s, p, r : réel

Debut

Lire (m , l)

$r := m / 2$

$p := 2 * l + (2 * r * \text{Pi})$

$s := (l * m) + (\text{Pi} * r * r)$

écrire ("le périmètre est :",p)

écrire ("la surface est :",s)

fin

Activité 2 : Correction

Exercice 3

- **Questionner sur les variables de l'algorithme**
 - N (entier): le nombre de quadruplets
 - Max (entier): l'entier max dans chaque quadruplet
 - Min(entier): l'entier min dans chaque quadruplet
 - Maximum(entier): max de tous les entiers
 - Minimum(entier): min de tous les entiers
- **Questionner sur les structures qu'il est possible d'utiliser pour résoudre le problème**
 - **Structure répétitive pour :**
 - vérifier si le nombre saisi est positif
 - vérifier si l'entier saisi est entre 0 et 20
 - pour saisir les 4 quadruplets
 - **Structure itérative pour:**
 - Chercher le min/max d'un quadruplet
 - Chercher le min/max de tous les entiers

Activité 2 : Correction

Exercice 3

Quadruplets
Var N, I, j, x, min, max, minimum, maximum : entier

Debut

Répéter

Lire(N)

Jusqu'à N > 0

minimum := 20

maximum := 0

Pour I de 1 à N faire

min := 20

max := 0

Pour j de 1 à 4 faire

Répéter

Lire(x)

Jusqu'à x >= 0 ET x <= 20

Si x < min Alors

min := x

finSi

Si x > max Alors

max := x

finSi

FinPour

Ecrire ("le minimum du", I, "ème quadruplet est", min)
Ecrire ("le maximum du", I, "ème quadruplet est :", max)

Si min < minimum Alors

minimum := min

finSi

Si max > maximum Alors

maximum := max

finSi

FinPour

Ecrire ("le minimum de tous les nombres est :", minimum)

Ecrire ("le maximum de tous les nombres est :", maximum)

Fin

Activité 2 : Correction

Exercice 4

- **Demander de lister les entrées du problème ainsi que leurs types**
 - $m \rightarrow$ mois (entier)
 - $a \rightarrow$ année (entier)
 - $j \rightarrow$ jour (entier)
- **Rappeler l'expression permettant de vérifier si une année a est bissextile ou non**
 - si $(a \bmod 4 = 0)$ ET $(a \bmod 100 \neq 0)$ OU $(a \bmod 400 = 0)$ alors a est bissextile sinon non
- **Questionner sur la possibilité d'utiliser la structure alternative Si_alors_sinon pour écrire l'algorithme**
 - Non car dans ce cas l'emboîtement de si en cascade est fastidieux à écrire. Il est préférable d'utiliser le schéma conditionnel généralisé.
- **Demander de lister les variables sur lesquelles un test sera effectué**
 - La variable a pour déterminer s'il s'agit d'une année bissextile et ainsi déterminer le nombre de jours pour le mois de février
 - La variable m pour déterminer le nombre de jours pour les autres mois
- **Guider les stagiaires dans l'écriture de l'algorithme en se basant sur les résultats obtenus précédemment**

Activité 2 : Correction

Exercice 4 (suite)

```
Nb-jours
Var m , a , j: entier
Debut
    lire( m , a )
    cas m de :
        1 : j := 31
        3 : j := 31
        5 : j := 31
        7 : j := 31
        8 : j := 31
        10 : j := 31
        12 : j := 31
```

```
4 : j := 30
6 : j := 30
9 : j := 30
11 : j := 30
2 : si (a mod 4 = 0) ET ( (a mod 100 ≠ 0) OU (a mod 400 = 0)) alors
    j := 29
    sinon
        j := 28
    finsi
    default : j := 0
Fincas
écrire("Le nombre de jours est :", j)
```

Fin

Activité 2 : Correction

Exercice 5

- **Rappeler la définition de la fonction factorielle**

$$0! = 1$$

$$n! = n \times (n - 1) \times (n - 2) \times \dots \times 2 \times 1 \quad \text{pour tout } n \neq 0$$

- **Définir les structures à utiliser pour résoudre le problème**
 - Structure répétitive répéter..jusqu'à pour vérifier la valeur de n
puisque le nombre d'itérations n'est pas connu

La condition d'arrêt est $n \geq 0$
 - Structure répétitive pour... faire pour calculer le factorielle puisque le
nombre d'itération est connu (n)
- **Ecrire l'algorithme factorielle correspondant**

Factorielle

Var n, i, res : entier

Debut

Répéter

Lire(n)

Jusqu'à $n \geq 0$

res := 1

Pour i de 1 à n Faire

res := res * i

FinPour

Ecrire("La factorielle de ", n, "est: ", res)

Fin

Activité 2 : Correction

Exercice 6

- **Rappeler la définition de la fonction pgcd**

$\text{pgcd}(a, b) = a$ si $b = 0$

$\text{pgcd}(a, b) = \text{pgcd}(b, a \bmod b)$ si $b \neq 0$

- **Définir les structures à utiliser pour résoudre le problème**

- Structure répétitive répéter..jusqu'à pour vérifier la valeur de a et b(supérieure à 0) puisque le nombre d'itérations n'est pas connu

La condition d'arrêt est $a > 0$ ($b > 0$)

- Structure répétitive tantQue... faire pour calculer le pgcd la condition $b \neq 0$ doit être vérifiée pour chaque itération

- **Ecrire l'algorithme factorielle correspondant**

PGCD

Var a , b , c: entier

Debut

Répéter

Lire(a)

Jusqu'à a > 0

Répéter

Lire(b)

Jusqu'à b > 0

TantQue (b ≠ 0) faire

c := a mod b

a := b

b := c

FinTQ

Ecrire ("Le plus grand commun diviseur est : " , a)

Fin

Activité 2 : Correction

Exercice 7

1. Demander de lister les résultats calculés pour la résolution du problème ainsi que leurs types

- tarifForfaitKilo : Réel → tarif calculée pour une location au kilométrage
- Questionner sur les formules permettant de déterminer ce résultat
 - tarifForfaitKilo?
 - Si $km \leq 100 \rightarrow \text{tarif1} := km * t1$
 - Si $km > 100$ et $< 1000 \rightarrow \text{tarif1} := 100 * t1 + (km - 100) * t2$
 - Si $km > 1000 \rightarrow \text{tarif1} := 100 * t1 + 900 * t2 + (km - 1000) * t3$
- Questionner les types de structures à utiliser pour résoudre le problème
 - Structure itératives pour vérifier toutes les formules déjà précisées
 - Structures répétitives pour vérifier les valeurs des données saisies
- Guider les stagiaires dans l'écriture de l'algorithme en se basant sur les résultats obtenus précédemment

Activité 2 : Correction

Exercice 7 (suite)

Algorithmelocation

Var

nbTotalKilo: entier

nbJour: entier

t1,t2,t3: réel

tarifForfaitKilo, ass: réel

Début

Répéter

Ecrire("donner le tarifs t1,t2 et t3")

Lire(t1)

Lire(t2)

Lire(t3)

Jusqu'à t1>0 ET t2>0 et t3>0

Répéter

Ecrire("donner le nombre de kilométrages")

Lire(nbTotalKilo)

Jusqu'à nbTotalKilo>0

Répéter

Ecrire("donner le nombre de jour")

Lire(nbJour)

Jusqu'à nbJour >0

Répéter

Ecrire("Donner le montant de l'assurance")

Lire(ass)

Jusqu'à nbJour >0

Si nbTotalKilo \leq 100 alors

tarifForfaitKilo \leftarrow nbTotalKilo * t1 + ass*nbJour

Sinon

Si nbTotalKilo >1000 alors

tarifForfaitKilo \leftarrow 100*t1 + 900*t2 + (nbTotalKilo-1000)*t3 +
ass*nbJour

sinon

tarifForfaitKilo \leftarrow 100*t1 + (nbTotalKilo -100)*t2 + ass*nbJour

FinSi

FinSi

Ecrire (" tarifForfaitKilo=" +
tarifForfaitKilo)

Fin

Activité 2 : Correction

Exercice 7 (suite)

2. Lister le résultat calculé pour la résolution du problème

TarifJournalier : Réel → tarif calculé pour un forfait journalier

- Questionner sur les formules permettant de déterminer ces résultats
 - TarifJournalier ?
 - $\text{TarifJournalier} = \text{nbJour} * (\text{t4} + \text{ass})$
- Guider les stagiaires à modifier l'algorithme en se basant sur les résultats obtenus précédemment

Algorithme location

Var

.....

TarifJournalier, t4: réel

Début

.....

Ecrire("donner tarif t4")

Lire (t4)

$\text{TarifJournalier} \leftarrow \text{t4} * \text{nbJour} + \text{nbJour} * \text{ass}$

.....

Fin

Activité 2 : Correction

Exercice 7 (suite)

3. Amélioration de l'algorithme pour choisir le forfait le plus avantageux

Algorithme location

.....

Début

.....

Si tarifForfaitKilo < TarifJournalier **alors**

Ecrire ("la forfait au kilomètre est plus avantageux")

Sinon

si tarifForfaitKilo > TarifJournalier **alors**

Ecrire ("la forfait journalier est plus avantageux")

Sinon

Ecrire ("les 2 forfaits sont égaux")

FinSi

FinSi

Fin

Activité 3

Structurer un algorithme

Compétences visées :

- Maîtrise de la programmation modulaire (procédures et fonctions)
- Maîtrise des notions de paramètres formels et de paramètres effectifs
- Maîtrise des différents types de passage des paramètres
- Manipulation correcte des variables globales et des variables locales

Recommandations clés :

- Bonne compréhension du problème général et le lien entre les procédures et les fonctions représentant les solutions des sous-problèmes dégagés



07 heures



CONSIGNES

1- Pour le formateur

- Demander de décortiquer le problème en sous-problèmes
- Demander la définition d'une fonction/procédure pour résoudre chaque sous-problème
- Mettre l'accent sur la différence entre une procédure et une fonction
- Mettre l'accent sur les différents types de passage de paramètres
- Mettre l'accent sur la portée des variables.
- Demander l'écriture de l'algorithme principal faisant appel aux procédures et fonctions définies précédemment.

2- Pour l'apprenant

- Décortiquer le problème en sous-problèmes
- Définir une fonction/procédure pour résoudre chaque sous-problème
- Comprendre la différence entre une variable globale et une variable locale
- Définir les différents types de passages de paramètres
- Ecrire l'algorithme principal faisant appel aux procédures et fonctions définies précédemment.



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Déclarer une procédure et une fonction ?
 - Appeler une procédure et une fonction dans un programme principal ?
 - Distinguer les différents types de passage de paramètres ?
 - Distinguer les différents types de variables (globales/locales) ?



Activité 3 : Structurer un algorithme

Exercice 1

Faire la trace d'exécution de l'algorithme AlgParamètres suivant :

AlgParamètres

Var

i, j : Entier

transmit(a : entier ; **var** b : entier)

Début

a := a + 100

a := b + 100

Ecrire("a = ", a, "b = ", b)

Fin

Début

i := 1

j := 2

transmit(i, j)

Ecrire("i = ", i, "j = ", j)

Fin

Activité 3 : Structurer un algorithme

Exercice 2

Soit l'algorithme licorne suivant contenant la définition de chacune des procédures dormir() et marcher().

Pour chacune des instruction d'écriture dites s'il s'agit d'une instruction correcte ou non. Justifier votre réponse

licorne

Var

taille : entier

charge : entier

dormir()

var tempsSommeil : entier

Début

temps=120

Ecrire(tempsSommeil)

Ecrire(tempsMarche)

Ecrire(charge)

Ecrire(taille)

Fin

marcher()

Var

tempsMarche : entier

Début

tempsMarche=60

Ecrire(ecrire(tempsSommeil)

Ecrire(tempsMarche)

Ecrire(charge)

Ecrire(taille)

Fin

Début

Ecrire(ecrire(tempsSommeil)

Ecrire(tempsMarche)

Ecrire(charge)

Ecrire(taille)

Fin

Activité 3 : Structurer un algorithme

Exercice 3

1. Ecrire une fonction qui permet de lire deux entiers X et Y strictement positifs puis apporter les modifications nécessaires pour transformer cette fonction en procédure
2. Ecrire une fonction qui permet de retourner la somme des diviseurs d'un entiers positif puis apporter les modifications nécessaires pour transformer cette fonction en procédure
3. Deux entiers positifs X et Y sont dits nombres AMIS si $SX = Y$ et $SY = X$, avec :
 - SX est la somme des diviseurs de X excepté lui-même.
 - SY est la somme des diviseurs de Y excepté lui-même.

Ecrire un algorithme permettant de vérifier si deux entiers positifs sont amis ou non (utiliser les fonctions/procédures déjà définies)

Exercice 4

Ecrire une procédure qui permet de dessiner un triangle isocèle d'étoiles de hauteur h en intégrant une procédure qui permet d'afficher une ligne.

Par exemple pour h = 3 on obtient le triangle suivant :

```
  *  
 ***  
*****
```

Activité 3 : Structurer un algorithme



Exercice 5

Un rationnel est caractérisé par son :

- Numérateur (entier)
- Dénominateur (entier)

On vous demande d'écrire les procédures et les fonctions suivantes :

1. Une procédure **saisie_rat** permettant la saisie d'un rationnel.
2. Une procédure **somme_rat** permettant de calculer et d'afficher la somme de deux nombres rationnels
3. Une procédure **produit_rat** permettant de calculer et d'afficher le produit de deux nombres rationnels
4. Une procédure **inverse_rat** permettant d'inverser un nombre rationnel
5. Une procédure **comparaison_rat** permettant de comparer deux rationnels
6. Une fonction **est_irréductible** permettant de vérifier si un rationnel est irréductible ou non.

Indication: On considère la fonction **nb_div (entier, entier)** prédéfinie qui retourne le nombre de diviseur commun de deux entiers donnés.

7. Une procédure **affiche_rat** permettant d'afficher un rationnel

Activité 3 : Structurer un algorithme



Exercice 5 (suite)

En utilisant les procédures/fonctions définies ci-dessus, écrire un algorithme principal permettant de :

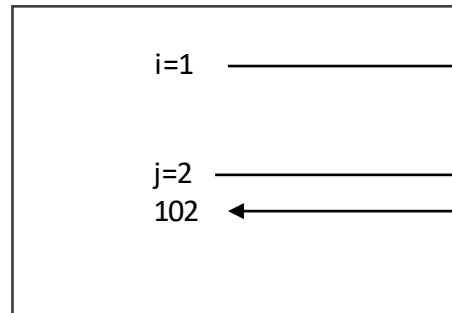
- Saisir deux nombres rationnels R1 et R2,
- Calculer et afficher la somme de deux nombres rationnels R1 et R2.
- Calculer et afficher le produit de deux nombres rationnels R1 et R2.
- inverser et afficher le nombre rationnel R2 avant et après l'inversion.
- Vérifier si R2 est irréductible ou non

Activité 3 : Correction

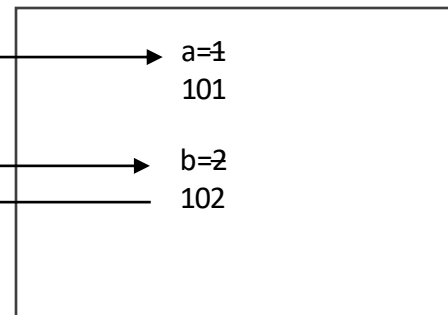
Exercice 1

- **Rappeler les différents types de passage des paramètres**
 - Passage par valeur → paramètre donnée
 - Passage par adresse → paramètre résultat ou d'un paramètre donnée/résultat

Espace mémoire défini pour les variables du programme principal



Espace mémoire défini pour les variables de la procédure



- A la fin de l'exécution de cet algorithme, on aura $a=101$ $b=102$
 $i=1$ $j=102$

Activité 3 : Correction

Exercice 2

- **Rappeler la différence entre variable globale et une variable locale**
- La portée d'une variable globale est totale : tout sous-algorithme du programme principal peut utiliser cette variable
- La portée d'une variable locale est uniquement le sous-algorithme qui la déclare

licorne

Var

taille : entier

charge : entier

dormir()

var tempsSommeil : entier

Début

temps=120

Ecrire(tempsSommeil) → OK

Ecrire(tempsMarche) → NO (variable locale à **marcher()**)

Ecrire(charge) → OK

Ecrire(taille) → OK

Fin

marcher()

Var

tempsMarche : entier

Début

tempsMarche=60

Ecrire(ecrire(tempsSommeil) → NO (variable locale à **dormir()**)

Ecrire(tempsMarche) → OK

Ecrire(charge) → OK

Ecrire(taille) → OK

Fin

Début

Ecrire(ecrire(tempsSommeil) → OK

Ecrire(tempsMarche) → OK

Ecrire(charge) → OK

Ecrire(taille) → Ok

Fin

Activité 3 : Correction

Exercice 3

1. Demander la définition d'une fonction/procédure permettant de retourner un entier positif saisi au clavier

```
lire_ent() : entier  
Var x : entier  
Début  
  Répéter  
    Lire ( x )  
  Jusqu'à x > 0  
  retourner x  
Fin
```

```
lire_ent(varx:entier)  
Début  
  Répéter  
    Lire ( x )  
  Jusqu'à x > 0  
Fin
```

- Mettre l'accent sur la différence entre une procédure et une fonction

Contrairement à une procédure, une fonction a une valeur de retour. Une procédure est un bloc d'instruction qui a la possibilité de modifier la valeur d'un ou de plusieurs de ses arguments. Une fonction n'a pas cette possibilité-là.

Activité 3 : Correction

Exercice 3 (suite)

2. Demander une définition d'une fonction/procédure permettant de retourner la somme des diviseurs d'un entier positif

```
Somme_div(X : entier) : entier  
Var S, I, M : entier  
Debut  
S := 0  
M := X div 2  
Pour I de 1 à M faire  
  Si X mod I = 0 Alors  
    S := S+I  
  FinSi  
Finpour  
retourner S  
Fin
```

```
Somme_div(X : entier, var s: entier)  
Var I, M : entier  
Debut  
M := X div 2  
Pour I de 1 à M faire  
  Si X mod I = 0 Alors  
    S := S+I  
  FinSi  
Finpour  
Fin
```

- Demander l'écriture de l'algorithme principal faisant appel aux procédures et fonctions définies précédemment.
- Demander de proposer 2 versions. La première utilise uniquement les fonctions alors que la deuxième ne fait appel qu'aux procédures.

Activité 3 : Correction

Exercice 3 (suite)

3. Programme principal faisant appel aux procédures et fonctions déjà définies

Version utilisant les fonctions

```
Amis
var x, y, sx, sy : entier
Début
x := lire_ent ( )
y := lire_ent ( )
sx := Somme_div ( x )
sy := Somme_div ( y )
Si sx = y ET sy = x Alors
    Ecrire ("Les 2 entiers sont amis")
Sinon
    Ecrire ("Les 2 entiers ne sont pas amis")
Finsi
Fin
```

Version utilisant les procédures

```
Amis
var x, y, sx, sy : entier
Début
X=y=sx=sy=0
lire_ent(x)
lire_ent(y)
Somme_div(x,sx)
Somme_div(y,sy)
Si sx = y ET sy = x Alors
    Ecrire ("Les 2 entiers sont amis")
Sinon
    Ecrire ("Les 2 entiers ne sont pas amis")
Finsi
Fin
```

Exercice 4

- Proposer de définir la procédure **ligne_etoile**

Ligne_etoile permet d'afficher une ligne d'étoile en précisant le nombre d'espaces à considérer ainsi que le nombre d'étoiles

- Proposer de définir la procédure **triangle_etoiles** permettant de dessiner le triangle isocèle

La procédure fait appel à la procédure **ligne_etoile** et prend en considération le nombre de ligne à dessiner (h la hauteur du triangle)

- **ligne_etoile**
(nb_espace : entier,
nb_etoile : entier)
- **Var j : entier**
- **Début**
- **Pour j de 1 à**
nb_espace faire
- **Ecrire(" ")**
- **FinPour**
- **Pour j de 1 à**
nb_etoile faire

triangle_etoiles (h : entier)

Var j : entier

Début

Pour i de 1 à h faire

Ligne_etoile (h-i, 2*i - 1)

FinPour

Fin

Activité 3 : Correction



Exercice 5

Saisie_rat(var num : entier, var denom : entier)

Début

Ecrire("donner le numérateur du rationnel")

Lire(num)

Ecrire("donner le dénominateur du rationnel")

Lire(denom)

Fin

produit_rat(num1 : entier, denom1 : entier, num2 :
entier, denom2 : entier, var numP : entier, var denomP :
entier)

Début

numP:= num1* num2

denomP:= denom1* denom2

Fin

somme_rat(num1 : entier, denom1 : entier, num2 : entier, denom2 : entier,
var numS : entier, var denomS : entier)

Début

numS:= num1* denom2 + num2* denom1

denomS:= denom1* denom2

Fin

Inverse_rat(var num1 : entier, var denom1 : entier)

Var

X: entier

Début

X:=num1

num1 := denom1

denom1 := num1

Fin

Activité 3 : Correction



Exercice 5 (suite)

comparaison_rat(num1: entier, denom1 : entier, num2: entier, denom2 : entier)

Début

Si (num1* denom2)>(num2* denom1) alors

Ecrire ("R1 est supérieur à R2")

Sinon

Si (num1* denom2)<(num2* denom1) alors

Ecrire ("R2 est supérieur à R1")

Sinon

Ecrire ("R1 =R2")

FinSi

Fin

est-irréductible(num1: entier, denom1 : entier) :booléen

Début

Si nb_div(num1,denom1)=1

retourner vrai

Sinon

retourner faux

Fin

affiche_rat(num1: entier, denom1 : entier)

Début

Ecrire(num1 + "/" + denom1)

Fin

Activité 3 : Correction



Exercice 5 (suite)

```
comparaison_rat( num1: entier, denom1 : entier, num2: entier, denom2 : entier )
```

Début

Si $(\text{num1} * \text{denom2}) > (\text{num2} * \text{denom1})$ alors

Ecrire ("R1 est supérieur à R2")

Sinon

Si $(\text{num1} * \text{denom2}) < (\text{num2} * \text{denom1})$ alors

Ecrire ("R2 est supérieur à R1")

Sinon

Ecrire ("R1 =R2")

FinSi

Fin

Activité 3 : Correction



Exercice 5 (suite)

ManipRationnel

var

Num1, denom1, num2, denom2 : entier

NumS, denomS : entier

NumP, denomP : entier

Début

saisie_rat(num1,denom1)

saisie_rat(num2,denom2)

somme_rat(num1,denom1, num2,denom2, NumS, denomS)

Ecrire ("La somme des deux rationnels est")

affiche_rat(NumS, denomS)

produit_rat(num1,denom1, num2,denom2, NumP, denomP)

Ecrire ("Le produit des deux rationnels est")

affiche_rat(NumP, denomP)

Inverse_rat(num2,denom2)

Ecrire ("l'inverse de R2 est")

affiche_rat(num2,denom2)

si est-irréductible(num2,denom2) alors

Ecrire("R2 est irréductible")

Sinon

Ecrire("R2 est non irréductible")

FinSi

Fin

Activité 4

Structurer les données

Compétences visées :

- Manipuler les tableaux vecteurs
- Manipuler les matrices
- Manipuler les chaînes de caractères

Recommandations clés :

- Utiliser les procédures et les fonctions pour faciliter la résolution des problèmes



07 heures

CONSIGNES

1- Pour le formateur

- Demander de décortiquer le problème en sous-problèmes (si c'est nécessaire)
- Demander de préciser le type de la structure itérative à utiliser pour le parcours du tableau (vecteur, matrice)
- Demander de manipuler les chaînes de caractères comme un vecteur de caractères

2- Pour l'apprenant

- Décortiquer le problème en sous-problèmes (si c'est nécessaire)
- Préciser le type de la structure itérative à utiliser pour le parcours du tableau (vecteur, matrice)
- Manipuler les chaînes de caractères comme un vecteur de caractères



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Définir un tableau vecteur et le manipuler ?
 - Définir une matrice et la manipuler ?
 - Maîtriser les principaux algorithmes de tri d'un tableau ?
 - Définir les chaînes de caractères et les manipuler ?



Activité 4 : Structurer les données

Exercice 1

On souhaite écrire un algorithme pour manipuler un tableau T de taille N / $N=20$. L'algorithme contient un ensemble de procédures et fonctions

1. Ecrire une procédure **remplir_Tab()** permettant de remplir un tableau T passer en paramètre
2. Ecrire une procédure **afficher_Tab()** permettant d'afficher les éléments d'un tableau T passé en paramètre
3. Ecrire une fonction **rechercher_Tab()** qui permet de vérifier si un entier existe dans un tableau T ou non. La fonction retourne vrai si cet élément existe et faux sinon
4. Ecrire une procédure **ajouterElement_Tab()** permettant d'ajouter un élément à un tableau T donné en paramètre
5. Ecrire une procédure **trier_Tab()** permettant de trier un tableau
6. Ecrire une fonction **est-ordonné()** qui étant donné un tableau contenant N entiers ($N \leq 100$) (on suppose le tableau rempli) affiche :
 - Vrai : si le tableau est ordonné dans un ordre croissant
 - Faux : si le tableau n'est pas ordonné
7. Ecrire le corps principal de l'algorithme permettant de tester toutes les procédures et fonctions déjà définies

Activité 4 : Structurer les données



Exercice 2

On souhaite programmer le jeu du pendu. Le but de ce jeu consiste à trouver un mot caché dont on connaît la taille, et ceci en proposant un caractère à chaque essai. Si le joueur propose un caractère qui existe dans le mot on l'affiche où il se trouve dans le mot (pour toutes ses occurrences) et cet essai n'est pas compté. Si le caractère n'existe pas dans le mot, cela compte pour un essai.

L'objectif est de découvrir le mot en un nombre d'essais maximum (5 essais).

Pour programmer ce jeu, l'idée est d'utiliser 2 tableaux de caractères. Un premier tableau **Tcar** pour placer le mot à trouver et un autre tableau **TEssai** contenant l'état du jeu.

TEssai est de même longueur que **Tcar** mais rempli par le caractère '*'. Lorsqu'un caractère est trouvé, le tableau **TEssai** est mis à jour: on place le caractère trouvé à la place du '*' pour chacune de ses occurrences.

Activité 4 : Structurer les données

A vous de jouer

1. Ecrire une fonction **MotValide** qui vérifie la validité d'un mot. Cette fonction retourne vrai si le mot est valide et faux si non sachant qu'un mot est valide si :

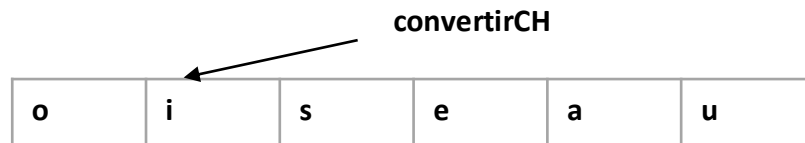
- le nombre de caractères est entre 4 et 25.
- Tous les caractères sont en majuscules.

Indications:

- les codes ascii sont obtenus par l'instruction `ascii("...")`. Exemple `ascii('A')=65`.
- Les codes ascii des majuscules sont entre 65 et 90.

2. Ecrire la procédure **convertirCH** qui convertit un mot en un tableau de caractères. Chaque case du tableau contient un caractère du mot.

Exemple : mot="oiseau"



Activité 4 : Structurer les données

Exercice 2 (suite)

3. Ecrire la procédure **initialiserSolution** qui prend en paramètre un tableau de caractères Tcar et qui demande à l'utilisateur de saisir le mot à trouver et qui remplit Tcar par ce mot (utiliser la fonction MotValide et la procédure convertirCH). Cette procédure prend aussi en paramètre n: la taille du mot à trouver
4. Ecrire la procédure **creerEssai** qui permet remplir le tableau de caractères TEssai par le caractère '*'. Les paramètres de la procédure sont: n et TEssai
Exemple n=6

*	*	*	*	*	*
---	---	---	---	---	---

5. Ecrire la procédure **afficherC** prenant en paramètre Tcar et n et affiche Tcar à l'écran comme une chaîne de caractères
6. Ecrire la fonction **jouer** prenant en paramètre la solution (tableau de caractères Tcar) , le tableau de caractères TEssai, un caractère et n. Cette fonction retourne vrai si le caractère est présent dans la solution et met à jour le tableau TEssai : le caractère doit remplacé à la bonne place le caractère '*' (pour toutes ses occurrences).
Le cas échéant, retourne faux.

Activité 4 : Structurer les données

Exercice 2 (suite)

Exemple :

Tcar :

o	i	s	e	a	u
---	---	---	---	---	---

Tessai :

*	*	*	*	*	*
---	---	---	---	---	---

Caractère='s'

Tessai :

*	*	s	*	*	*
---	---	---	---	---	---

7. Ecrire la fonction **estFini** qui prend en paramètre TEssai et n. Elle retourne vrai si tous les caractères du mot ont été trouvés et faux si non.
8. Ecrire l'algorithme principal réalisant les actions suivantes:
 - Demande du mot à trouver (procédure initialiserSolution)
 - Création du tableau Essai à partir de la taille de la solution (procédure creerEssai)

Activité 4 : Structurer les données



Exercice 2 (suite)

- La boucle principale du jeu :
 - Demander à l'utilisateur de saisir un caractère
 - Mettre à jour le tableau Essai et le nombre d'essais
 - Arrêter le jeu lorsque le nombre d'essais atteint 5 ou lorsque le jeu est fini (le mot recherché est trouvé) dans ce cas le jeu affiche le résultat : gagné ou perdu !

Activité 4 : Structurer les données

Exercice 3

Écrire un algorithme qui lit 2 entiers N et P (> 0 et ≤ 100) puis saisit **N*P** entiers et les stocke dans une matrice ligne par ligne. La matrice est d'une taille de **N** lignes et **P** Colonnes.

Ensuite, l'algorithme affiche le contenu de la matrice ligne par ligne et alternativement de gauche à droite puis de droite à gauche.

Par exemple, si la matrice est la suivante :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Le tableau de dimensions 4 * 4 seront affichés dans cet ordre :

1, 2, 3, 4

8, 7, 6, 5

9, 10, 11, 12

16, 15, 14, 13

Activité 4 : Correction

Exercice 1

1. Procédure remplir_Tab()

- Questionner sur le type de structure nécessaire pour remplir le tableau

Structure répétitive pour.. Faire puisque le nombre d'itérations est connu

- Questionner sur le type de paramètre

Paramètre E/S (ajout du mot clé var)



```
remplir_Tab ( Var T : Tab)
Var
i : entier

Debut
Pour i de 1 à N Faire
    Ecrire("entrer un entier")
    Lire(T[i])
Fin pour
Fin
```

2. Procédure afficher_Tab()

- Questionner sur le type de structure nécessaire pour afficher le tableau

Structure répétitive pour.. Faire puisque le nombre d'itérations est connu

- Questionner sur le type de paramètre

Paramètre d'entrée



```
afficher_Tab() (T : Tab)
var
i : entier

Debut
Ecrire(« Les elements du tableau sont : »)
Pour i de 1 à N faire
    Ecrire T[i])
Fin pour
Fin
```

Activité 4 : Correction

Exercice 1 (suite)

3. Procédure rechercher_Tab()

- Questionner sur le type des structure nécessaires pour chercher un élément dans le tableau

Structure répétitive tant que.. pour le parcours du tableau et la recherche de l'élément puisque le nombre d'itération est non connu

Structure conditionnelle pour vérifier si l'élément existe ou non

- Questionner sur le type des paramètres

Paramètres d'entrées



```
rechercher_Tab() (T : Tab, x : entier)
var
i : entier

Debut
i:=1
Tant que ((i<= n) ET (T[i] < x)) Faire
i++
FinTQ

Si i > n alors
Ecrire ("Elément introuvable ")
Sinon
Ecrire (i)
Fin Si
Fin
```

Activité 4 : Correction

Exercice 1 (suite)

4. Procédure ajouterElement_Tab

- **Questionner sur le type de paramètre**
Paramètres d'entrées
- Mettre l'accent sur le fait d'augmenter la taille du tableau N:N+1

```
ajouterElement_Tab (T : Tab, x : entier)
```

```
Début  
T[N]:=x  
N=N+1  
Fin
```

5. Procédure trier_Tab()

- **Questionner sur le type de paramètre**
Paramètres d'entrées
- Mettre l'accent le fait de faire appel à la procédure Tri_selection

```
trier_Tab (T : Tab, x : entier)
```

```
Debut  
Tri_Selection(T)  
afficher_Tab(T)  
Fin
```

```
Tri_Selection(Var T : Tab)
```

```
var
```

```
i, j, x, indmin : Entier
```

```
Début
```

```
Pour i de 1 à (N-1) Faire
```

```
indmin := i
```

```
Pour j de (i+1) à n Faire
```

```
Si (T[j] < T[indmin]) Alors
```

```
indmin := j
```

```
FinSi
```

```
FinPour
```

```
x:= T[i]
```

```
T[i]:= T[indmin]
```

```
T[indmin]:=x
```

```
FinPour
```

```
Fin
```

Activité 4 : Correction

Exercice 1 (suite)

6. Fonction est_ordonnée

- **Demander de préciser le type de la structure itérative à utiliser pour la résolution de ce problème**
 - Quelle est la condition d'arrêt de la structure itérative TantQue ? Puisque le nombre d'itérations n'est pas connu à l'avance...
- **Donner une indication:** Ajouter une variable booléenne Test qui est initialisée à vrai et prend faux si $T[i] > T[i+1]$



```
Est_ordonné(T:Tab) : booléen
Var
  l, x : entier
  test : booléen
Début
  test := vrai
  i := 1
  Tant que i < N ET test Faire
    Si T[ i ] > T[ i+1 ] Alors
      test := faux
    Finsi
    i := i+1
  FinTQ

  Si test = vrai Alors
    retourner vrai
  Sinon
    retourner faux
Fin
```

Activité 4 : Correction

Exercice 1 (suite)

7. Programme principal



```
AlgorithmePrincipal  
Type  
Tab: tableau[1..Max] d'entier  
Const  
Max=100 : entier  
Var  
T : Tab  
N: entier  
Début  
N=20  
remplir_Tab(T)  
afficher_Tab(T)  
rechercher_Tab(T,5)  
ajouterElement_Tab (T,4)  
  
Si (Et_ordonné(T)= faux) Alors  
    trier_Tab(T)  
FinSi  
  
Fin
```


Activité 4 : Correction

Exercice 2

1. fonction **MotValide** qui vérifie la validité d'un mot

- Questionner sur les types des structures nécessaires pour la définition de la fonction

Structure conditionnelle pour vérifier la longueur d'un mot

Structure itérative pour le parcourir du mot et la vérification des lettres majuscules



```
MotValide(mot : chaine) :booléen
Var
val : booléen
i : entier
Début
Si (long(mot)<=25) ET (long(mot)>=4) alors
    val=vrai
    i=1
    TantQue (val=vrai ) faire
        Si (65<=ascii(mot[i])) ET (90>=ascii(mot[i])) alors
            i=i+1
        Sinon
            val=faux
    FinTQ
Retourner val
Sinon
Retourner faux
Fin
```

Activité 4 : Correction



Exercice 2 (suite)

2. procédure **convertirCH** qui convertit un mot en un tableau de caractères. Chaque case du tableau contient un caractère du mot.

convertirCH (mot : chaîne, var Tcar : tableau [1..25] de caractère)

var

i : entier

Début

Pour i de 1 à long(mot)

 Tcar[i] ← mot[i]

FinPour

Fin

Activité 4 : Correction

Exercice 2 (suite)

3. Ecrire la procédure **initialiserSolution**

Mettre l'accent sur le fait d'appeler les procédures précédentes

initialiserSolution (var Tcar : tableau [1..25] de caractère, var n)

var

i : entier

Début

Ecrire ("Donner le mot à chercher")

Lire(mot)

n=long(mot)

Si MotValide(mot) alors

convertirCH(mot,Tcar)

FinSi

Fin

Activité 4 : Correction

Exercice 2 (suite)

4. Procédure **creerEssai** permettant remplir le tableau de caractères TEssai par le caractère '*'.

- Mettre l'accent sur la nécessité de parcourir le tableau TEssai

```
creerEssai (var TEssai : tableau [1..25] de caractère, n: entier)
var
  i : entier
Début
  Pour i de 1 à n faire
    TEssai[i] ← '*'
  FinPour
Fin
```

5. Procédure **afficherC** prenant en paramètre Tcar et n et affiche Tcar à l'écran comme une chaîne de caractères

```
afficherC (var Tcar : tableau [1..25] de caractère, n: entier)
var
  i : entier
Début
  Pour i de 1 à n faire
    Ecrire (Tcar[i])
  FinPour
Fin
```

Activité 4 : Correction

Exercice 2 (suite)

6. Fonction jouer

- Questionner sur les paramètres de cette fonction et leurs types

Tcar: tableau de caractères → paramètre E/S

TEssai: tableau de caractères → paramètre d'entrée

C: un caractère → paramètre d'entrée

n: un entier → paramètre d'entrée

- Questionner sur le type de retour de cette fonction

Un booléen

**jouer (var Tcar : tableau [1..25] de caractère, TEssai : tableau [1..25] de caractère
c : caractère, n: entier) : booléen**

var

b : booléen

Début

b ← faux

Pour i de 1 à n faire

Si Tcar [i]=c alors

TEssai [i]=c

b ← vrai

FinSi

FinPour

Retourner b

Fin

Activité 4 : Correction

Exercice 2 (suite)

7. Ecrire la fonction **estFini**

- Questionner sur les paramètres de cette fonction et leurs types

TEssai: tableau de caractères → paramètre d'entrée

n: un entier → paramètre d'entrée

- Questionner sur le type de retour de cette fonction

Un booléen

- Questionner sur les types des structures nécessaires

Structure conditionnelle pour vérifier s'il existe encore une lettre non dévoilée

Structure répétitive pour parcourir le tableau → on utilise la structure Tant que
puisque on va pas forcément parcourir tout le tableau

estfini (TEssai : tableau [1..25] de caractère, n: entier) : booléen

var

b : booléen

Début

b ← vrai

i=1

TantQue (b= vrai) et (i<=n) faire

Si TEssai [i]='*' **alors**

 b ← faux

sinon

 i=i+1

FinSi

FinTQ

Retourner b

Fin

Activité 4 : Correction

Exercice 2 (suite)

8. Algorithme principal réalisant les actions suivantes:

- Demande du mot à trouver (procédure initialiserSolution)
- Création du tableau Essai à partir de la taille de la solution (procédure creerEssai)

AlgorithmeJeu

Var

Tcar, TEssai : tableau[1..25] de caractère

nbEssai, n : entier

c : caractère

Début

n=0

initialiserSolution(Tcar,n)

creerEssai(Tessai,n)

nbEssai=0

Répéter

Ecrire("Donner un caractère")

Lire(c)

Jouer (Tcar,TEssai,c)

afficheC(Tessai,n)

nbEssai=nbEssai+1

jusqu'à (nbEssai=5) ou estfini(Tessai)

si estfini(Tessai, n) alors

 ecrire("gagnant")

Sinon

 Ecrire("Perdant")

FinSi

Fin

Activité 4 : Correction

Exercice 3

- Demander de proposer une procédure qui permet de saisir une matrice

```
Saisie_matrice( var M: tableau[1..100,1..100] d'entiers, N: entier, P: entier)
Debut
  Pour i de 1 à N faire
    Pour j de 1 à P faire
      lire(M[i , j ] )
    Finpour
  Finpour
Fin
```

- Demander de proposer une procédure qui permet d'afficher une matrice

```
affiche_matrice( var M: tableau[1..100,1..100] d'entiers, N: entier, P: entier)
Debut
  Pour i de 1 à N faire
    Pour j de 1 à P faire
      Ecrire(M[ i , j ] )
    Finpour
  Finpour
Fin
```


Activité 4 : Correction

Exercice 3 (suite)

- Demander de proposer une procédure qui permet de permuter deux entiers

```
Permuter(var n: entier, var fin: entier)
V: entier
Début
    v := deb
    deb := fin
    fin := v
Fin
```

- Demander d'écrire un algorithme permettant un affichage accordéon de la matrice en se basant sur les étapes précédentes

```
Affiche_accordéon
Var
M: tableau[1..100,1..100] d'entiers
N,P : entiers
i , j, deb, fin, v : entier
Début
Lire(N)
Lire(P)
saisie_matrice( M, N, P)
affiche_matrice( M, N, P)
```

```
deb := 1
fin := P
k:=1
Pour i de 1 à N Faire
    Pour j de deb à fin (pas= k) Faire
        écrire( M[ i , j ] )
    Finpour
    Permuter(deb,fin)
    k:=k+1
Finpour
Fin
```



PARTIE 3

PROGRAMMER EN PYTHON

Dans ce module, vous allez :

- Transformer une suite d'instructions algorithmiques en suite d'instructions Python
- Manipuler les données



24 heures



Activité 1

Transformer une suite d'instructions algorithmiques en suite d'instructions Python

Compétences visées :

- Manipuler les bases du langage de programmation Python (variable d'entrée, variable de sortie les instructions d'affectations, les instructions d'entrées/sorties, les structures itératives les structures conditionnelles etc.)
- Appliquer les bonnes pratiques de la programmation Python ainsi que l'optimisation d'un code python

Recommandations clés :

- Faire une bonne lecture des exercices
- Réfléchir à une solution en pseudo-code et la traduire en Python
- Garantir une optimisation du code en appliquant les bonnes pratiques, les commentaires, etc.



09 heures

CONSIGNES

1- Pour le formateur

- Demander de proposer un algorithme permettant de trouver une solution au problème
- Rappeler la syntaxe générale du langage Python
- Demander de traduire l'algorithme proposé en langage Python
- Demander d'assurer une optimisation du code et une application des bonnes pratiques de codage

2- Pour l'apprenant

- Proposer un algorithme permettant de trouver une solution au problème
- Se rappeler de la syntaxe générale du langage Python
- Traduire l'algorithme proposé en langage Python
- Garantir une optimisation du code Python développé et une application des bonnes pratiques du codage



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe
- Utilisation d'un ordinateur
- Utilisation d'un environnement intégré de développement IDE installé sur ordinateur (IDLE Python 3.9, PyCharm...)

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Assurer une traduction structurée des algorithmes en Python?
 - Ecrire des blocs d'instructions en Python ?
 - Assurer une optimisation du code en appliquant les bonnes pratiques du langage Python et ?



Activité 1 :

Transformer une suite d'instructions algorithmiques en suite d'instructions Python

Exercice 1 :

Nous souhaitons développer une application en Python permettant de:

- Saisir un nombre fini d'entiers compris entre 1 et 10.
- Afficher la somme des inverses des entiers saisis par l'utilisateur.
- Afficher le plus petit, le plus grand entier saisi ainsi que la moyenne de tous les entiers saisis par l'utilisateur.
- Afficher le nombre des entiers premiers parmi la suite des entiers saisis. Un nombre premier n'est divisible que par 1 ou par lui-même.
- L'application propose un menu de choix offrant toutes les fonctionnalités décrites précédemment.

SaisieEntiers

Var

n, r : Entier

Début

n:=0

Répéter

TantQue (n<1) ou (n>10)

Ecrire("entrer un entier positif:")

lire(n)

FinTQ

Ecrire("voulez vous ajouter un autre entier? ")

Ecrire("Tapez 1 si oui ")

lire(r)

Jusqu'à (r<>1)

Fin

Activité 1 :

Transformer une suite d'instructions algorithmiques en suite d'instructions Python

Exercice 1 :

A vous de jouer

1. Proposer une traduction de l'algorithme **saisieEntiers** suivant en langage Python.
2. Proposer un algorithme pour chacune des fonctionnalités décrites et traduire chacun des algorithmes proposé en langage Python.
3. Développer l'application proposant le menu de choix en Python.

NB. Veuillez à appliquer les bonnes pratiques de développement en Python.

Activité 1 :

Transformer une suite d'instructions algorithmiques en suite d'instructions Python

Exercice 2 :

Nous souhaitons développer le jeu suivant en Python :

- A tour de rôle, 2 joueurs choisissent 3 entiers qui ne peuvent prendre que 3 valeurs : 0, 1 ou 2.
- Si la différence entre la somme des nombres choisis par chacun est :
 - Pair: le joueur qui a proposé la plus grande somme d'entiers gagne.
 - Impair: le joueur qui a proposé la plus petite somme d'entiers gagne.

SaisieEntiers

Var

nbEntier: entier

Début

nbEntier=0

Ecrire("Premier Joueur à vous de jouer")

Tantque(nbEntier<3)

Repéter

Ecrire("entrer 0-1-2:")

jusqu'à (n<>0) ou (n<>1) ou (n<>2)

nbEntier=nbEntier+1

FinTQ

nbEntier=0

Ecrire("Deuxième Joueur à vous de jouer")

Tantque(nbEntier<3)

Repéter

Ecrire("entrer 0-1-2:")

jusqu'à (n<>0) ou (n<>1) ou (n<>2)

nbEntier=nbEntier+1

FinTQ

Fin

Activité 1 :

Transformer une suite d'instructions algorithmiques en suite d'instructions Python



Exercice 2 :

A vous de jouer

1. Traduire l'algorithme suivant permettant la saisie des entiers choisis par les 2 joueurs.
2. Ecrire un programme Python pour ce jeu.
3. On souhaite améliorer le jeu comme suit: si la différence entre la somme des nombres choisis par chacun vaut :
 - 1: le joueur qui a proposé la plus grande somme d'entiers gagne un point.
 - 2: le joueur qui a proposé la plus petite somme d'entiers gagne un point.
 - aucun point n'est marqué

Le jeu se termine quand l'un des joueurs totalise 10 points

Ecrire un programme Python permettant de simuler la nouvelle version du jeu

Activité 1 : Correction

Exercice 1 :

1 - Traduction de l'algorithme de saisie d'un nombre fini d'entiers compris entre 0 et 10

- Questionner sur le type de la structure nécessaire en Python pour traduire l'algorithme

La structure While (structure répétitive)

SaisieEntier

Var

n, r : Entier

Début

Répéter

n:=0

TantQue (n<1) ou (n>10)

Ecrire("entrer un entier positif:")

lire(n)

FinTQ

Ecrire("voulez vous ajouter un autre entier? ")

Ecrire("Tapez 1 si oui ")

lire(r)

Jusqu'à (r<>1)

Fin

r = '1'

while r == '1': *#structure while pour vérifier si l'utilisateur a encore des entiers à saisir*

n=0

while (int(n) <1) or (int(n) > 10): *#structure while permettant de vérifier si n est entre 1 et 10*

print ("entrer un entier positif:")

n = input()

print ("voulez vous ajouter un autre entier?")

print ("Tapez 1 si oui")

r = input() *#récupérer le choix de l'utilisateur*

Activité 1 : Correction



Exercice 1 (suite) :

2- Algorithme d'affichage de la somme des inverses des entiers et sa traduction en Python

- Questionner sur le type de la structure nécessaire en Python pour traduire l'algorithme

La structure While (structure répétitive)

- Mettre l'accent sur la nécessité de respecter l'indentation
- Mettre l'accent sur la nécessité de commenter le code

SommeInverse

Var

n, r : Entier

Début

n:=0

sh=0

Répéter

n:=0

TantQue (n<1) ou (n>10)

Ecrire("entrer un entier positif:")

lire(n)

FinTQ

sh=sh+1/n

Ecrire("voulez vous ajouter un autre entier?")

Ecrire("Tapez 1 si oui")

lire(r)

Jusqu'à (r<>1)

Ecrire("la somme des inverses=",sh)

Fin

r = '1'

sh = 0

while r == '1':

n = 0

while (int(n) < 1) or (int(n) > 10):

print("entrer un entier positif:")

n=input()

sh = sh + 1 / int(n) *#calculer la somme des inverses*

print ("voulez vous ajouter un autre entier?")

print ("Tapez 1 si oui")

r = input()

print('la somme des inverses=',sh)

Exercice 1 (suite) :

- Algorithme d'affichage de min, max et moyenne des entiers saisis et sa traduction en Python.

MinMaxMoy

Var

n, r, min, max : Entier
moy: réel

Début

nb=0
somme:= 0
max:=0
min:=20

Répéter

n:=0

TantQue (n<1) ou (n>10)

Ecrire("entrer un entier positif:")
lire(n)

FinTQ

nbnb+1

somme:=somme+n

Si (n > max) Alors

max:= n

FinSi

Si (n < min) Alors

min:= n

FinSi

Ecrire("voulez vous ajouter un autre entier? ")
Ecrire("Tapez 1 si oui ")
lire(r)

Jusqu'à (r<>1)

Ecrire("max=" ,max)
Ecrire("Min=" ,min)
Ecrire("Moyenne=" ,somme/nb)

Fin

Activité 1 : Correction



Exercice 1 (suite) :

- Traduction de l'algorithme **MinMaxMoy** en Python.
 - Rappeler la syntaxe de la structure itérative en Python
 - Mettre l'accent sur la nécessité de respecter l'indentation
 - Mettre l'accent sur la nécessité de commenter le code

```
r = '1'
nb = 0 #initialisation du nombre d'entiers
somme = 0
max = 0 #initialisation de max à 0
min = 20 #initialisation de min à 20

while r=='1':
    n=0
    while (int(n)<0) or (int(n)>10):
        print("entrer un entier positif:")
        n=input()

    nb=nb+1 #incrémenter le nombre d'entiers saisi
```

```
#while r=='1':
.....
.....
if ( int (n) > max ) : #si un entier saisi est >max alors mise à jours de max
    max = int(n)

if ( int(n) < min ) : #si un entier saisi est <min alors mise à jours de min
    min=int(n)

somme = somme + int(n) #calcul de la somme des entiers
print("voulez vous ajouter un autre entier?")
print("Tapez 1 si oui")
r = input() #saisie du choix de l'utilisateur

print ('Max=',max)
print ('Min=',min)
print ('Moyenne=',somme/nb)
```

Activité 1 : Correction

Exercice 1 (suite) :

- Algorithme des entiers premiers et sa traduction en Python

EntierPremier

Var

n, r : Entier

Début

n:=0

nbpremier=0

Répéter

TantQue (n<1) ou (n>10)

Ecrire("entrer un entier positif:")

lire(n)

FinTQ

nb_div=0

i=2

TantQue (i <= n/2) Faire

Si (n Mod i = 0) Alors

nb_div := nb_div + 1

Finsi

i:= i + 1

FinTQ

Si (nb_div = 0) Alors

nbpremier=nbpremier+1

FinSi

Ecrire("voulez vous ajouter un autre entier?")

Ecrire("Tapez 1 si oui ")

lire(r)

Jusqu'à (r<>1)

Ecrire("le nombre des entiers premiers=",nbpremier)

Fin

Activité 1 : Correction



Exercice 1 (suite) :

- Traduction de l'algorithme **EntierPremier** en Python.
 - Rappeler la syntaxe de la structure itérative en Python
 - Mettre l'accent sur la nécessité de respecter l'indentation
 - Mettre l'accent sur la nécessité de commenter le code

```
r = '1'
nbpremier = 0

while r == '1':
    n = 0
    while ( int(n) < 0) or ( int(n) > 10):
        print ("entrer un entier positif:")
        n = input()

    nb_div = 0
    i = 2
    while ( i <= int(n) / 2):
        if ( int(n) % i == 0 ): #recherche des diviseur de n
            nb_div += 1
        i += 1

    if ( nb_div == 0): #vérifier si n et 1 sont les seuls diviseurs de n
        nbpremier = nbpremier + 1 #incrémenter le nombre des premiers

    print("voulez vous ajouter un autre entier?")
    print("Tapez 1 si oui")
    r=input()

print("le nombre des entiers premiers=",nbpremier)
```

Activité 1 : Correction



Exercice 1 (suite) :

3. L'application avec un menu de choix offrant toutes les fonctionnalités décrites précédemment.

```
r = '1'
n = 0
sh = 0
nb = 0
somme = 0
max = 0
min = 20
nbpremier = 0
```

```
while r=='1':
```

```
    while (int(n)<1) or (int(n)>10):
        print("entrer un entier positif:")
        n=input()
```

```
#while r=='1':
#.....
#.....
#-----calcul de la somme des inverses-----
    sh = sh + 1 / int(n)

#-----calcul de max, min et moyenne-----
    nb = nb + 1

    if ( int(n) > max ):
        max = int(n)

    if ( int(n) < min):
        min = int(n)
    somme = somme + int(n)
```

```
#while r=='1':
#.....
#.....
#-----calcul nombre des entiers premier-----
    nb_div = 0
    i = 2

    while ( i <= int (n) / 2):
        if ( int(n) % i == 0 ) :
            nb_div += 1
            i+=1

    if ( nb_div == 0 ) :
        nbpremier = nbpremier + 1
#-----
    print ("voulez vous ajouter un autre entier?")
    print ("Tapez 1 si oui")
    r = input()
```


Exercice 1 (suite) :

```
repchoix = '1'

while (repchoix == '1'): #structure while pour vérifier si l'utilisateur souhaite encore appliquer des fonctionnalités
    print ("veuillez saisir votre choix:")
    print("1-Afficher somme des inverses")
    print("2-Afficher Min, Max et moyenne")
    print("3-Afficher le nombre des entiers premiers")

    choix = input()
    if(choix=='1'): #choix=1(somme des diviseurs)
        print('la somme des inverses=',sh)
    elif (choix=='2'):#choix=2(MinMax,moy)
        print('Max=',max)
        print('Min=',min)
        print('Moyenne=',somme/nb)
    elif (choix=='3'):#choix=3 (nombres premiers)
        print("le nombre des nombre premiers=",nbpremier)
    else:
        print("choix incorrecte")
        print("voulez vous continuez?")
        print("tapez 1 si oui")
        repchoix=input()
```

Exercice2 :

1-Traduction de l'algorithme SaisieEntiers en Python

```
nbEntier=0
print("premier Joueur à vous de jouer")

while nbEntier<3: #vérifier si le nombre des entiers saisis est <3
    n=5
    while (int(n)!=0) and (int(n)!=1) and(int(n)!=2):
        print("entrer 0-1-2:")
        n=input()
    nbEntier=nbEntier+1 #incrémenter le nombre d'entiers saisis
    #-----Deuxième Joueur-----
    nbEntier=0
    print("deuxième Joueur à vous de jouer")
    while nbEntier<3:
        m=5
        while (int(m)!=0) and (int(m)!=1) and(int(m)!=2):
            print("entrer 0-1-2:")
            m=input()
        nbEntier=nbEntier+1
```

Activité 1 : Correction



Exercice2

2. Développement de la première version du jeu

```
sommeJ1 = 0 #initialisation des sommes des entiers de J1
sommeJ2 = 0 # initialisation des sommes des entiers de J2
#-----Premier Joueur-----
nbEntier = 0
print ("premier Joueur à vous de jouer")
while nbEntier < 3:
    n = 5
    while ( int (n) != 0) and ( int(n) != 1) and ( int (n) != 2 ):
        print("entrer 0-1-2:")
        n=input()

    sommeJ1 = sommeJ1 + int(n) #sommer les entiers saisis de J1
    nbEntier= nbEntier + 1 #incrémenter le nombre des entiers de J1
```

```
#-----Deuxième Joueur-----
nbEntier=0
print("deuxième Joueur à vous de jouer")
while nbEntier<3:
    m=5
    while ( int (n) != 0) and ( int(n) != 1) and ( int (n) != 2 ):
        print("entrer 0-1-2:")
        n=input()

    sommeJ2 = sommeJ2 + int(m) #sommer les entiers saisis de J2
    nbEntier= nbEntier + 1 #incrémenter le nombre des entiers de J2

#Annoncer le gagnant
if ( ( sommeJ1 - sommeJ2 ) % 2 ) == 0 :
    print("J1 a gagné le jeu")
else:
    print("J2 a gagné le jeu")
```

Activité 1 : Correction



Exercice2 (suite)

3-Développement en Python de la version améliorée du jeu

```
tot_J1 = 0 #initialiser le score de J1
tot_J2 = 0 #initialiser le score de J2

while (tot_J1 < 5 ) and ( tot_J2 < 5 ) :
    sommeJ1 = 0
    sommeJ2 = 0

    nbEntier = 0
    print("premier Joueur à vous de jouer")
    while nbEntier < 3:
        n=5
        while ( int(n) != 0 ) and ( int(n) != 1) and ( int(n) != 2 ):
            print("entrer 0-1-2:")
            n = input()

        sommeJ1=sommeJ1+ int(n)
        nbEntier=nbEntier+1
```

```
#while (tot_J1 < 5 ) and ( tot_J2 < 5 ) :
#.....
#.....
nbEntier = 0
print("premier Joueur à vous de jouer")
while nbEntier < 3:
    m = 5
    while ( int ( m ) != 0) and ( int (m) != 1) and (int (m) != 2):
        print("entrer 0-1-2:")
        m = input()

    sommeJ2=sommeJ2+ int(m)
    nbEntier=nbEntier+1
if( sommeJ1-sommeJ2 = 1 ):
    tot_J1 = tot_J1 + 1 #incrémenter le score de J1
    print("J1 a gagné le tour")

elif ( sommeJ1 - sommeJ2 = 2 ):
    tot_J2 = tot_J2 + 1 #incrémenter le score de J2
    print("J2 a gagné le tour« )
```

Activité 2

Manipuler les données

Compétences visées :

- Manipuler les fonctions et les fonctions lambdas
- Manipuler les structures de données en Python
- Manipuler les types de fichiers de données
- Manipuler quelques bibliothèques standards de Python

Recommandations clés :

- Faire une bonne lecture de l'énoncé de l'exercices
- Faire une distinction entre les différentes structures de données de Python et réviser la manipulation de chacune de ces structures de données



15 heures



CONSIGNES

1- Pour le formateur

- Demander d'identifier les paramètres de chacune des fonctions à développer
- Rappeler la définition d'une fonction lambda
- Rappeler les structures de données dictionnaire , liste , tuple et ensembles et les instructions de leurs manipulations
- Questionner sur les instructions d'écriture dans un fichier CSV (rappel du cours) et sur la bibliothèque qu'il est nécessaire d'importer
- Questionner sur les bibliothèques standards qu'il est nécessaire d'importer

2- Pour l'apprenant

- Identifier les paramètres de chacune des fonctions à développer
- Se rappeler de la définition d'une fonction lambda
- Se rappeler des structures de données dictionnaire , liste , tuple et ensembles et les instructions de leurs manipulations
- Définir les instructions d'écriture dans un fichier CSV (rappel du cours) et la bibliothèque qu'il est nécessaire d'importer
- Définir les bibliothèques standards qu'il est nécessaire d'importer



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travailler seul ou en groupe
- Utilisation d'un ordinateur
- Utilisation d'un environnement intégré de développement IDE installé sur ordinateur (IDLE Python 3.9, PyCharm...)

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Manipuler les fonctions?
 - Manipuler les fonctions lambda?
 - Utiliser adéquatement les structures de données Python?
 - Manipuler les fichiers en Python ?
 - Utiliser les bibliothèques standards



Activité 2 : Manipuler les données



Exercice 1

On souhaite créer une application Python permettant la gestion des comptes bancaires dans une banque :

- Un compte bancaire est défini par un identifiant unique du titulaire, le numéro de compte et son solde
- Deux opérations de base sont possibles sur son compte: déposer et retirer de l'argent
- Aucune facilité de caisse n'est autorisée (i.e. le solde d'un compte ne peut pas être négatif)
- Les acteurs de cette application sont: l'agent de la banque et un client. Chaque client dispose d'un code secret pour accéder à son compte. On suppose que chaque client a un seul compte

Nb :

- le numéro du client est incrémental
- Le numéro de compte est un entier formé par le numéro de son propriétaire et un nombre aléatoire compris entre 0 et 100

Exemple : Si le numéro du client est 5 alors son numéro de compte est par exemple: **556** (où **5** est le numéro du client et **56** est un nombre aléatoire compris entre 0 et 100)

A travers un menu de choix, cette application offre à l'agent les fonctionnalités suivantes :

1. Ajouter un Compte
2. Supprimer un Compte

Activité 2 :

Manipuler les données



Exercice 1 (suite)

- A travers un menu de choix cette application offre à un client les fonctionnalités suivantes :
 1. Modifier son mot de passe
 2. Afficher son solde
 3. Déposer une somme d'argent
 4. Retirer une somme d'argent

Indication : Utiliser la structure de données dictionnaire de Python pour la gestion des comptes et des clients

Exemple :

- Le dictionnaire **Compte** associe à chaque numéro de Compte son solde
Exemple: `Compte={56:200, 20:360}` signifie que le solde du compte numéro 56 est 200Dh et le solde du compte numéro 20 est 360Dh
- Le dictionnaire **Client** qui associe à chaque numéro de client son code secret
Exemple : `Client={1:"566", 2:"836"}` signifie que le code secret du client numéro 1 est égal à 566 et le code secret du client numéro 2 est égal à 836
- Le dictionnaire **ClientCompte** associe à chaque numéro de client son numéro de compte
Exemple : `ClientCompte {56:1,20:2}` signifie que le numéro de compte du client 56 est 1 et le numéro de compte du client 20 est 2

Activité 2 : Manipuler les données



Exercice 1(suite)

- On considère Client, Compte et ClientCompte les dictionnaires utilisés pour la gestion des comptes. Définir les fonctions suivantes relatives à l'agent de la banque :
 - ajouterClient**(numCl, MPC, numC, SoldeC) tels que: numCl est le numéro du nouveau client, MPC est son code secret, numC est le numéro de son compte et SoldeC est le solde initial de son compte
 - SupprimerClient (numC)** tel que numC est le numéro du compte à supprimer
- Définir les fonctions suivantes relatives à un client de la banque (identifier les paramètres de ces fonctions)
 - Modifier MPClient** : permettant de modifier le mot de passe d'un client
 - Déposer** : permettant de déposer une somme d'argent dans un compte
 - Retirer** : permettant de retirer une somme d'argent d'un compte
- Définir la fonction lambda `genererNumCompte` permettant de générer le numéro de compte d'un client à partir du numéro de ce client
- Ecrire une fonction `EcrireFichierCSV` permettant d'enregistrer les numéros des clients et leurs codes secrets dans un fichier CSV
- Ecrire une fonction `manipSTS` permettant de créer une liste, un tuple et un set de numéro de comptes à partir du dictionnaire *ClientCompte*
- Ecrire un programme principal permettant de tester les fonctions définies précédemment. Proposer un menu de choix à l'agent et aux clients de la banque

Activité 2 : Correction



Exercice 1

1. Rappeler la structure de données dictionnaire et les instructions de sa manipulation . Un **dictionnaire** est une collection non **ordonnée**, **modifiable** et **indexée**. En Python, les dictionnaires sont écrits avec des **accolades**, et ils ont des clés et des valeurs.
- Demander de proposer une définition des fonctions **ajouterClient** et **supprimerClient** :

```
def ajouterClient(numCl, MPC, numC, SoldeC):
```

```
    Client[numCl] = MPC
```

```
    Compte[numC] = SoldeC
```

```
    ClientCompte[numCl] = numC
```

```
def SupprimerClient(numCl):
```

```
    Client.pop(numCl) #utiliser la fonction pop pour supprimer le client ayant le numéro numCl
```

```
    x=ClientCompte[numCl] #récupérer le numéro de compte du client à partir du dictionnaire ClientCompte
```

```
    Compte.pop(x) #utiliser la fonction pop pour supprimer le compte du client en question
```

```
    ClientCompte.pop(numCl)
```

Activité 2 : Correction



Exercice 1 (suite)

2. La fonction ***modifierMPClient (numCl, ancienMP, nouveauMP)*** ayant les paramètres suivants:

- **numCl**: numéro du client souhaitant changer son mot de passe
- **ancienMP**: son ancien mot de passe.
- **NouveauMP**: son nouveau mot de passe.

La fonction ***deposer(NumCl, soldeD)*** ayant les paramètres suivants :

- **NumCl** le numéro de client souhaitant déposer de l'agent
- **soldeD**: le solde à déposer

La fonction ***retirer (NumCl, soldeR)*** les paramètres suivants :

- **NumCl** le numéro de client souhaitant déposer de l'agent
- **soldeR** : le solde à retirer
- Rappeler le fait qu'il est impératif de vérifier si le numéro du client existe et si le montant retiré est autorisé

Activité 2 : Correction

Exercice 1 (suite)

#cette fonction permet de vérifier si un client ayant un #numéro numCl existe

```
def rechercheClient(numCl):  
    if int(numCl) in ClientCompte: #vérifier si numCl est une clé dans ClientCompte  
        return True  
    else:  
        return False
```

#cette fonction permet de vérifier si le MP du client est #correcte

```
def VerifMPClient(numCl,MP):  
    if Client[int(numCl)] == MP:  
        print("MP Correcte")  
        return True  
    else:  
        print (Client[int(numCl)])  
        print("MP incorrecte!")  
        return False
```

```
def modifierMPClient(numCl, ancienMP, nouveauMP):  
    if Client[numCl] == ancienMP: #traiter le cas où l'ancienMP est correcte  
        Client[numCl] = nouveauMP  
    else: #traiter le cas où l'ancienMP est incorrecte  
        print("MP incorrecte")
```

```
def deposer(NumCl, soldeD):  
    x=ClientCompte[NumCl] #récupérer le numéro de compte du client  
    Compte[x] = Compte[x] + soldeD #ajouter le solde au compte  
    print(" Nouveau solde= " + Compte[x])
```

```
def retirer(NumCl, soldeR):  
    x=ClientCompte[NumCl]  
    if (Compte[x] - soldeR) < 0: #vérifier si le retrait est permis  
        print("solde Insuffisant!")  
    else:  
        Compte[x] = Compte[x] - soldeR #retirer le solde  
        print (" retrait effectué avec succès! ")  
        print(" Nouveau solde=" + Compte[x])
```

Activité 2 : Correction



Exercice 1 (suite)

3. Rappeler la définition d'une fonction lambda

Une fonction Lambda est comme n'importe quelle fonction Python normale, sauf qu'elle n'a pas de nom lors de sa définition et qu'elle est contenue dans une ligne

- **Questionner sur les bibliothèques qu'il est nécessaire d'importer pour générer le numéro de compte:** Les bibliothèques math et random

```
import math
import random
LambdaNumCompte = lambda numcl: str(numcl) + str(math.floor(random.uniform(0 , 100)))

#math.floor envoie le plus grand entier inférieur ou égal à un nombre donné, random.uniffform(0,100) retourne un nombre aléatoire entre 0 et 100
```

4. La fonction **EcrireFichierCSV** qui permet d'enregistrer les numéros des clients et leurs codes secrets dans un fichier CSV

```
def EcrireFichierCSV ():
    fichier = open("client.csv", "w")
    ecrivainCSV = csv.writer(fichier, delimiter=";")
    ecrivainCSV.writerow(["Numéro Client", "Code Secret"]) # On écrit la ligne d'en-tête avec le titre des colonnes
    for x,y in Client.items():
        ecrivainCSV.writerow([str(x), str(y)])
    fichier.close()
```

Activité 2 : Correction



Exercice 1 (suite)

```
def manipSTS()
    listC= [] #initialiser la liste
    setC=set({}) #initialiser le set
    for x, y in ClientCompte.items(): #parcourir les élément de ClientCompte
        listC.append(y) #ajouter le numéro de compte à la liste
        setC.add(y) #ajouter le numéro de compte au set

    print("*Liste*")
    print(listC)
    print("*Set*")
    print(setC)
    print("**Tuple")
    tupleC=tuple(listC) #transformer la liste en tuple
    print(tupleC)
```

Activité 2 : Correction



Exercice 1 (suite)

6. Le programme principal

```
if __name__ == '__main__':  
    #initialisation des dictionnaires  
    Compte = {1: 20, 2: 19, 3: 100}  
    Client = {1: '123', 2: '566', 3: '999'}  
    ClientCompte = {1: 1, 2: 2, 3: 3}  
  
    choixAC = input("Si vous êtes un agent bancaire tapez 1 si vous êtes un client tapez 2 \n")  
    resultat = False  
    resultatMP = False  
    r = False  
    numCl = 3  
    if choixAC == "1": #vérifier s'il s'agit d'un agent  
        mp = input("Donner votre mot de passe:")  
        if (mp == "0000"):  
            continuer=True  
            while (continuer):  
                print("*****Menu*****")  
                print("1-Ajouter un compte Client")  
                print("2-Supprimer un compte client")  
                print("3-Générer un fichier Client")
```

```
# if (mp == "0000"):  
#.....  
Choix = False  
while (Choix == False):  
    numchoix = input("*****Tapez le numéro de votre choix*****\n")  
    if numchoix == "1":  
        Choix = True  
        numCl = numCl + 1  
        numC = LambdaNumCompte(numCl)  
        solde = input("Donner le solde:")  
        ajouterClient(numCl, "6666", int(numC), int(solde))  
        print("client ajouté avec succès!")  
    elif numchoix == "2":  
        Choix = True  
        numclientSup = input("Donner le numéro du client à supprimer")  
        SupprimerClient(int(numclientSup))  
        print("client supprimé avec succès!")  
    elif numchoix == "3":  
        Choix = True  
        print("Fichier Client généré avec succès")  
        EcrireFichierCSV()  
    else:  
        print("Tapez 1 ou 2")  
else: # c'est le cas ou le mot de passe de l'agent est incorrecte  
    print("Mp incorrect")
```


Activité 2 : Correction



Exercice 1 (suite)

```
#if choixAC == "1": #vérifier s'il s'agit d'un agent
#.....
#.....
elif choixAC == "2": #vérifier s'il s'agit d'un client
    print(Client)
    ChoixNumC = False
    while (ChoixNumC == False):
        s = input("Tapez votre numero\n")
        if (rechercheClient(int(s)) == True):
            ChoixNumC=True #numClient est correcte
            ChoixMp = False
            while (ChoixMp == False): #mot de passe incorrecte
                mp = input("Donner votre mot de passe \n")
                if (VerifMPClient(s, mp) == True): #mot de passe correcte
                    ChoixMp = True
                    ChoixCorrecte = False
                    while (ChoixCorrecte == False): #choix incorrecte
                        print("*****Tapez un numéro au choix*****\n")
                        print("1-Afficher solde")
                        print("2-Déposer un montant")
                        print("3-Retirer un montant")
                        print("4-Modifier mot de passe")
                        numchoix = input("*****Tapez le numéro de votre choix*****\n")
```

Activité 2 : Correction



Exercice 1 (suite)

```
if numchoix == "1":
    ChoixCorrecte = True
    x = ClientCompte[int(s)]
    print("Votre solde=", Compte[x], "D")
elif numchoix == "2":
    ChoixCorrecte = True
    soldeD = input("Donner le montant à déposer:")
    deposer(int(s), float(soldeD))
    print(Compte)
    print("Solde déposé avec succès!")
elif numchoix == "3":
    ChoixCorrecte = True
    soldeD = input("Donner le montant à retirer:")
    retirer(int(s), float(soldeD))
    print(Compte)
    print("Solde retiré avec succès!")
elif numchoix == "4":
    print(Client)
    ChoixCorrecte = True
    nouveauMP = input("Donner votre nouveau MP")
    modifierMPClient(int(s), mp, nouveauMP)
    print("mot de passe modifié avec succès!")
else:
    print("Tapez 1, 2, 3 ou 4")
```

```
#if choixAC == "1": #vérifier s'il s'agit d'un agent
#.....
#.....
#elif choixAC == "2": #vérifier s'il s'agit d'un client
#-----
# if (rechercheClient(int(s)) == True):
#.....
#.....

# if (VerifMPClient(s, mp) == True):
#.....

    else:
        print("Veuillez vérifier votre mot de passe") #MP incorrecte
    else:
        print("Veuillez vérifier votre numéro") #numéro de client inexistant

else: #c'est le cas où le client où l'agent n'a pas saisi 1 ou entrer à son menu
    print("Tapez 1 ou 2")
```



PARTIE 4

Déployer une solution Python

Dans ce module, vous allez :

- Déboguer le code Python.
- Déployer une solution Python.



03 heures



Activité 1

Déboguer le code Python

Compétences visées :

- Gérer convenablement les erreurs en Python
- Pratiquer le débogage en Python
- Pratiquer les outils de suivi et de visualisation de l'exécution d'un code Python

Recommandations clés :

- Lire attentivement les erreurs déclenchées lors de l'exécution d'un programme
- Suivre les étapes de débogage d'un code en Python



1,5 heures

CONSIGNES

1- Pour le formateur

- Rappeler les différents types d'erreurs.
- Rappeler les méthodes de gestion des erreurs en Python.
- Questionner sur la méthode pour exécuter un code dans le débogueur.

2- Pour l'apprenant

- Identifier les différents types d'erreurs
- Se rappeler des méthodes de gestion des erreurs.
- Définir la méthode pour exécuter un code dans le débogueur.



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre des instructions verbales ou écrites du formateur
- Travail seul ou en groupe
- Utilisation d'un ordinateur
- Utilisation d'un environnement intégré de développement IDE installé sur ordinateur (IDLE Python 3.9, PyCharm...)

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Différencier entre les différents types d'erreurs (compilation/exécution)
 - Gérer les exceptions en Python ?
 - Suivre et visualiser l'exécution d'un code Python ?



Activité 1 :

Déboguer le code Python

Exercice 1 :

On considère le code Python suivant permettant de calculer le PPCM de deux entiers

Et d'écrire le résultat dans un fichier texte

1. Identifier les différentes erreurs syntaxiques présentes dans ce code et corriger les.
2. Exécuter le code après la correction des erreurs syntaxiques et vérifier le résultat obtenu. Calculer par exemple le PGCD de 6 et 8 vous devez trouver 24.
3. Identifier le type de l'erreur détectée et corriger la.
4. Déclencher des exceptions dans les différents cas suivants:
 - La valeur de b=0.
 - Les valeurs de a et b sont incorrectes (par exemple une chaîne de caractères au lieu d'un entier).
 - L'utilisateur choisit un mode incorrecte pour l'ouverture du fichier.
 - L'utilisateur choisit le mode « x » pour l'ouverture du fichier sachant que le fichier est existant.

```
print( Entrer la valeur de a :")
a = input()
print ("Entrer la valeur de b: )
b = input()

i = 1
while (((i * int(a) ) % int(b) ) != 0):
    i =i + 1

ppcm= i * a
ligne = "PPCM de " + a + " et " + b + " = " + ppcm
print(ligne)

print("Donner le mode d'ouverture du fichier : ")
print("a/w ou x")
mode = input()
f=open("PPCM1.txt",mode)
f.write(ligne)
print("\n")
f.close()
```

Activité 1 : Déboguer le code Python



Exercice 1 (suite) :

5. Lancer le débogueur de l'environnement IDLE pour suivre l'exécution du programme pour a=5, b=12 et mode d'ouverture du fichier r='a' (utiliser l'option Pas-à-pas détaillé)
6. Exécuter le débogueur intégré Python debugger (pdb) pour suivre l'exécution du programme
 - Définissez un breakpoint au niveau de la boucle While afin de vérifier le contenu des variables du problème

Activité 1 : Correction



Exercice 1 :

1. Il existe 2 erreurs syntaxiques:
 - Erreur1 → `print(" Entrer la valeur de a : ")`
 - Erreur2 → EOL while scanning string literal print au niveau de l'instruction ("Entrer la valeur de b:")
 - Erreur3 → `ligne = "PPCM de " + a + " et " + b + " =" + ppcm`
2. Résultat de PPCM de 6 et 8 est 6666
3. Une erreur sémantique ou logique. Correction de l'erreur → `ppcm=i*int(a)` : **il faut convertir la sortie de input en un entier**

Activité 1 : Correction

Exercice 1 (suite) :

```
print( "Entrer la valeur de a :")
a=input()
print("Entrer la valeur de b:" )
b=input()

try: #gestion des exceptions
i = 1
while (((i*int(a))% int(b))!=0):
    i =i + 1

ppcm=i*int(a)
ligne = "PPCM de " + a + " et " + b + " =" + str(ppcm)
print(ligne)
```

```
print("Donner le mode d'ouverture du fichier : ")
print("a/w ou x")
mode=input()
f =open("PPCM.txt",mode)
f.write(ligne)
print("\n")
f.close()

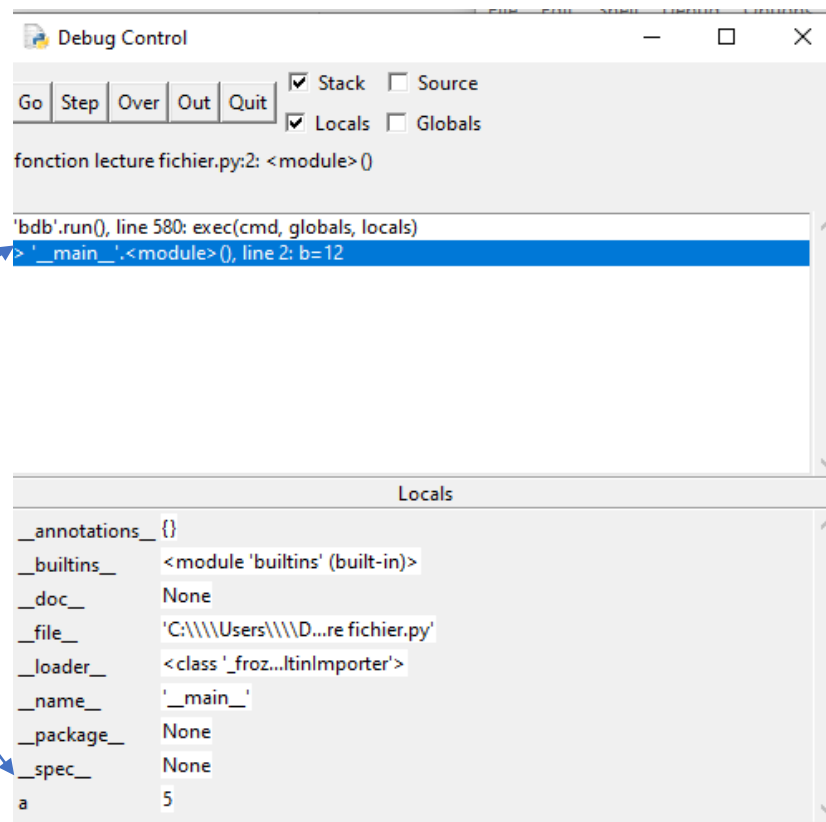
except ZeroDivisionError: #capture de l'exception ZeroDivisionError
#cas : b=0
    print('Erreur! b vaut 0')

except ValueError: : #capture de l'exception ValueError
#cas: valeurs de a et/ou b incorrecte
#cas: saisie de mode d'ouverture de fichier incorrecte
    print('Vérifier les valeurs de a et b')
    print('Vérifier le mode d ouverture du fichier')

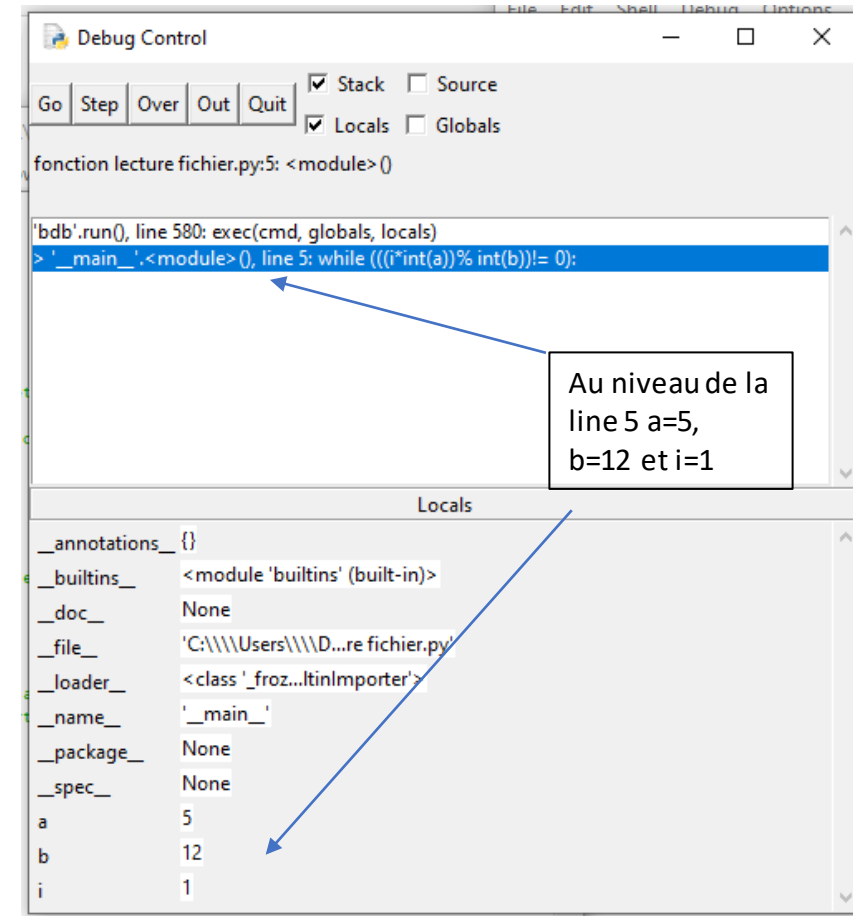
except FileExistsError: #capture de l'exception FileExistsError
#cas: fichier PPCM existe déjà et le mode choisi est 'x'
    print('Le fichier existe déjà')
```

Activité 1 : Correction

5. Lancement de débogueur de l'environnement IDLE pour suivre l'exécution du programme (avec a=5, b=12)

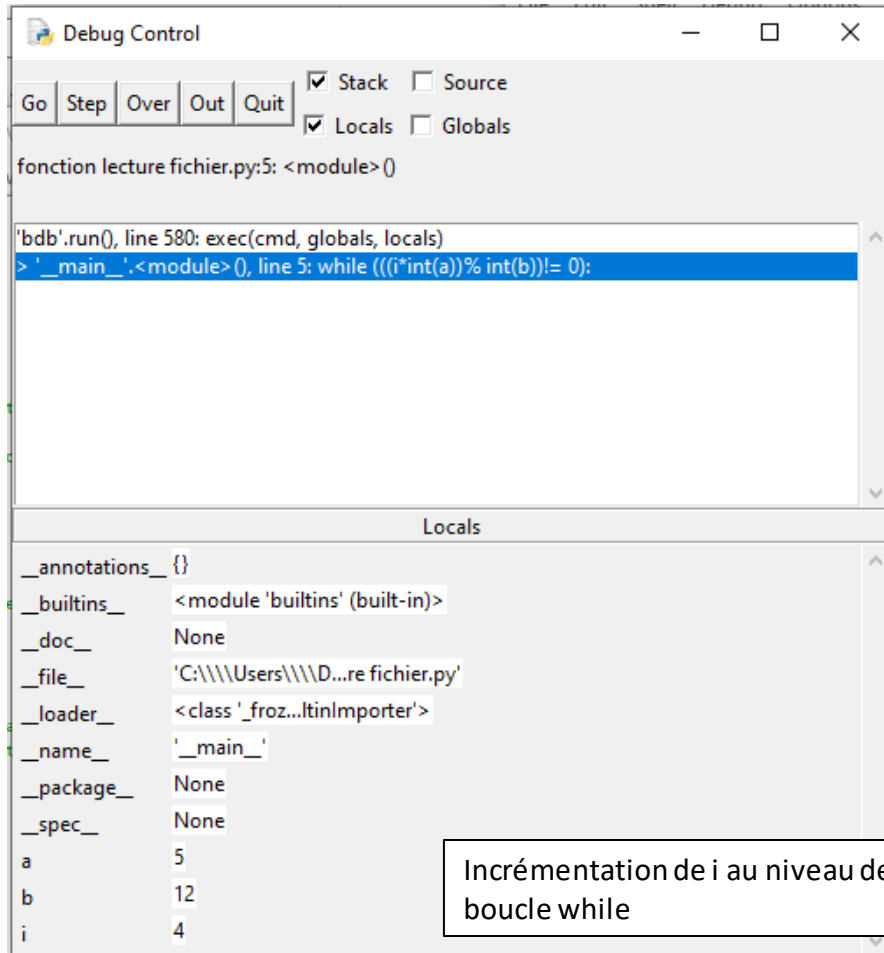


Au niveau de la
line 2 a=5



Au niveau de la
line 5 a=5,
b=12 et i=1

Activité 1 : Correction



Debug Control

Go Step Over Out Quit ☒ Stack ☐ Source
☒ Locals ☐ Globals

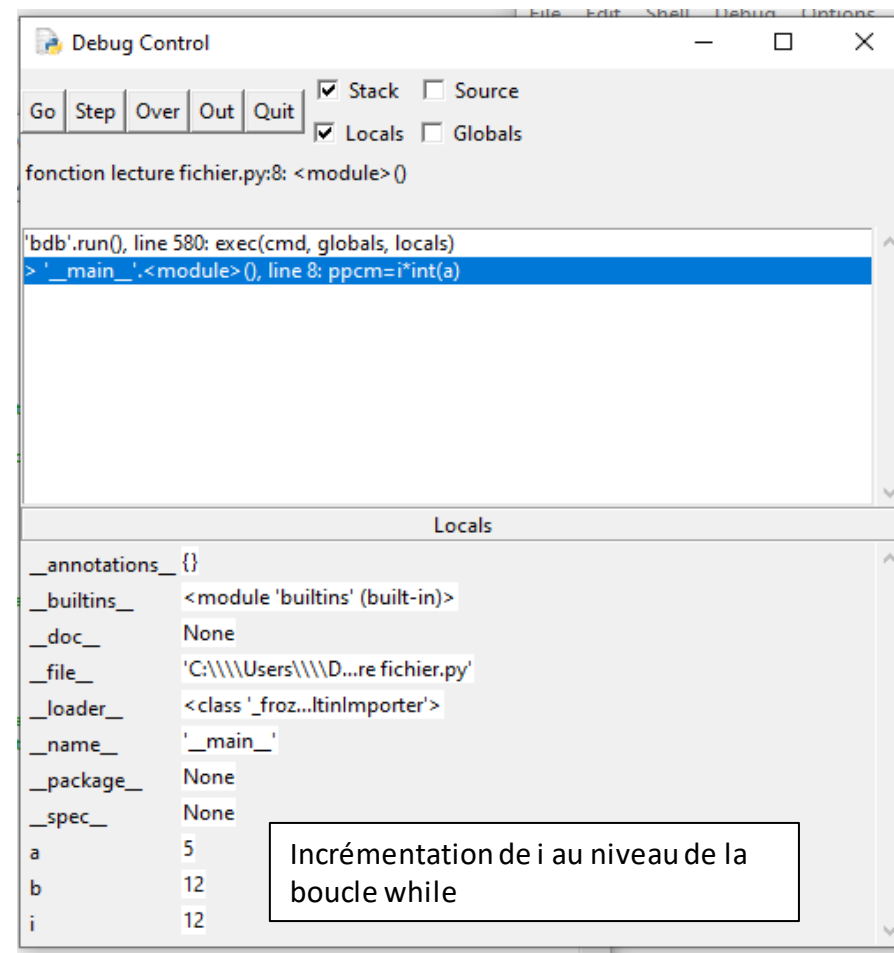
fonction lecture fichier.py:5: <module> ()

'bdb'.run(), line 580: exec(cmd, globals, locals)
> '__main__'.<module> (), line 5: while (((i*int(a))%int(b))!= 0):

Locals

__annotations__	{}
__builtins__	<module 'builtins' (built-in)>
__doc__	None
__file__	'C:\\\\Users\\D...re fichier.py'
__loader__	<class '_froz...ltinImporter'>
__name__	'__main__'
__package__	None
__spec__	None
a	5
b	12
i	4

Incrémentation de i au niveau de la
boucle while



Debug Control

Go Step Over Out Quit ☒ Stack ☐ Source
☒ Locals ☐ Globals

fonction lecture fichier.py:8: <module> ()

'bdb'.run(), line 580: exec(cmd, globals, locals)
> '__main__'.<module> (), line 8: ppcm=i*int(a)

Locals

__annotations__	{}
__builtins__	<module 'builtins' (built-in)>
__doc__	None
__file__	'C:\\\\Users\\D...re fichier.py'
__loader__	<class '_froz...ltinImporter'>
__name__	'__main__'
__package__	None
__spec__	None
a	5
b	12
i	12

Incrémentation de i au niveau de la
boucle while

Activité 1 : Correction

Debug Control

Go Step Over Out Quit

☒ Stack ☐ Source
☒ Locals ☐ Globals

fonction lecture fichier.py:9: <module> ()

'bdb'.run(), line 580: exec(cmd, globals, locals)
> '_main_'.<module> (), line 9: ligne = "PPCM de " + str(a) + " et " + str(b) + "=" + str(ppcm)

Locals

__annotations__	{}
__builtins__	<module 'builtins' (built-in)>
__doc__	None
__file__	'C:\\\\Users\\D\\re fichier.py'
__loader__	<class '_froz...ltinImporter'>
__name__	'_main_'
__package__	None
__spec__	None
a	5
b	12
i	12
ppcm	60

Pour i=12 PPCM=60

Debug Control

Go Step Over Out Quit

☒ Stack ☐ Source
☒ Locals ☐ Globals

fonction lecture fichier.py:10: <module> ()

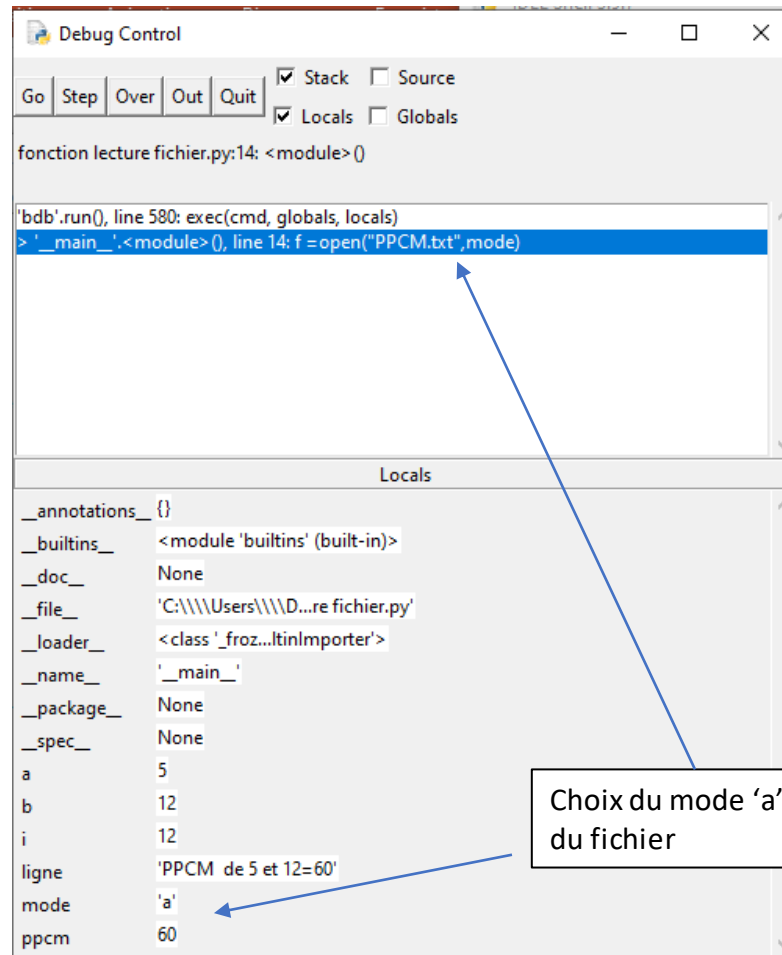
'bdb'.run(), line 580: exec(cmd, globals, locals)
> '_main_'.<module> (), line 10: print(ligne)

Locals

__annotations__	{}
__builtins__	<module 'builtins' (built-in)>
__doc__	None
__file__	'C:\\\\Users\\D\\re fichier.py'
__loader__	<class '_froz...ltinImporter'>
__name__	'_main_'
__package__	None
__spec__	None
a	5
b	12
i	12
ligne	'PPCM de 5 et 12=60'
ppcm	60

Affichage de la chaine ligne

Activité 1 : Correction



Choix du mode 'a' pour l'ouverture
du fichier

Activité 1 : Correction

```
a=5
b=12
try:
    i = 1
    while (((i*int(a))%int(b))!=0):
        import pdb;
        pdb.set_trace()
        i=i + 1

ppcm=i*int(a)
ligne = "PPCM de " + str(a) + " et " + str(b) + "=" + str(ppcm)
print(ligne)
print("Donner le mode d'ouverture du fichier: ")
print("a/w ou x")
mode='a'
f=open("PPCM.txt",mode)
f.write(ligne)
f.write("\n")
```

```
> c:\users\dell\desktop\tp algo\ppcm.py(8)<module>()
-> i =i + 1
(Pdb) i
1
(Pdb) c
> c:\users\dell\desktop\tp algo\ppcm.py(7)<module>()
-> pdb.set_trace()
(Pdb) i
2
(Pdb) i
2
(Pdb) c
> c:\users\dell\desktop\tp algo\ppcm.py(8)<module>()
-> i =i + 1
(Pdb) i
3
(Pdb) |
```

Affichage de la valeur de i au niveau de chaque breakpoint



Activité 2

Déployer une solution Python

Compétences visées :

- Déployer une solution Python
- Créer un fichier d'exécution en Python
- Générer une documentation en Python

Recommandations clés :

- Suivre convenablement toutes les étapes une par une pour assurer le déploiement d'une solution Python, la création de fichier d'exécution et la génération de documentation



1,5 heures

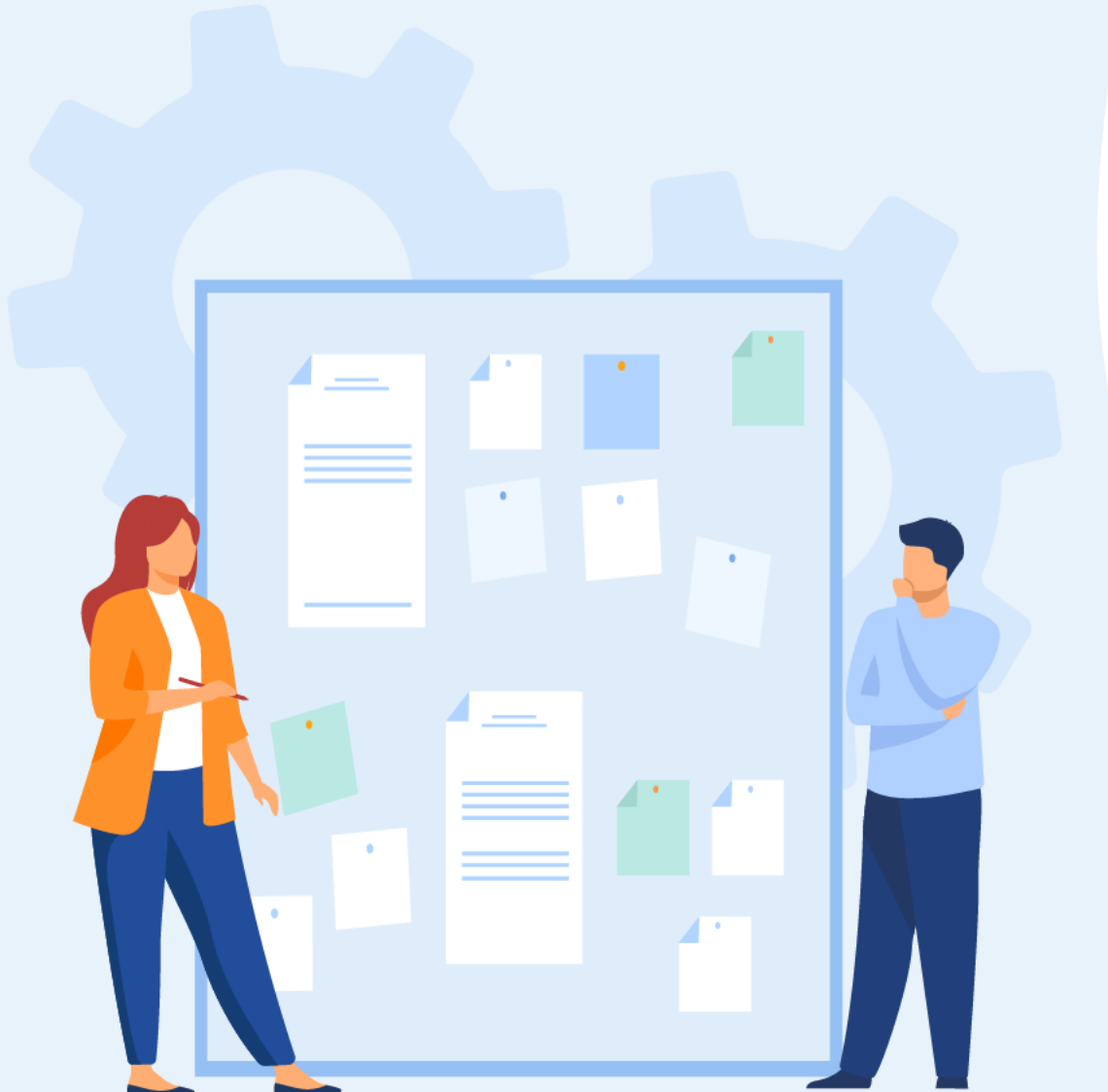
CONSIGNES

1- Pour le formateur

- Demander de vérifier que la dernière version de pip est installée
- Demander de créer la structure de fichiers nécessaire
- Demander de générer des packages de distribution
- Demander de créer un compte sur TestPyPI et de créer un jeton
- Demander d'installer Twine et de l'exécuter
- Demander d'exécuter les commandes nécessaires pour créer un exécutable
- Demander d'exécuter les commandes nécessaires pour créer une documentation

2- Pour l'apprenant

- Vérifier que la dernière version de pip est installée
- Créer la structure de fichiers nécessaire
- Générer des packages de distribution
- Créer un compte sur TestPyPI et créer un jeton
- Installer Twine et de l'exécuter
- Exécuter la commande nécessaire pour créer un exécutable
- Exécuter la commande nécessaire pour créer une documentation



CONSIGNES

3 - Conditions de réalisation

- Utilisation des supports pédagogiques fournis par le formateur
- Suivre les instructions verbales ou écrites du formateur
- Travail seul ou en groupe
- Utilisation d'un ordinateur
- Utilisation d'un environnement intégré de développement IDE installé sur ordinateur (IDLE Python 3.9, PyCharm...)
- Acquisition d'une connexion internet

4 - Critères de réussite

- Le stagiaire est-il capable de :
 - Déployer une solution Python ?
 - Créer un fichier exécutable ?
 - Générer une documentation ?



Activité 2 : Déployer une solution Python

Exercice 1 :

On désire réaliser une application qui évalue la force d'un mot de passe sachant qu'un mot de passe est une chaîne de caractères qui ne comporte ni un espace ni une lettre accentuée. La force d'un mot de passe est évaluée selon les critères suivants :

- Si score <20: le mot de passe est 'très faible'
- Si $20 \leq \text{score} < 40$: le mot de passe est 'faible'
- Si $40 \leq \text{score} < 60$: le mot de passe est 'Fort'
- Sinon: le mot de passe est 'très fort'

Pour calculer le score d'un mot de passe il faut calculer les bonus et les malus

Les bonus sont attribués comme suit :

- Nombre total de caractères *4
- (Nombre total de caractères – nombre de lettres majuscules) *2
- (Nombre total de caractères – nombre de lettres minuscules) *3
- Nombre de caractères non alphabétiques *5

Les malus sont attribués comme suit :

- La longueur de la plus longue séquence de lettre minuscules *2
- La longueur de la plus longue séquence de lettre majuscules *2

Activité 2 : Déployer une solution Python

Exercice 1 (suite) :

Exemple :

- Pour le mot de passe 'P@clF_annee2017', le score se calcule comme suit
- Nombre total de caractères=15
- Nombre de caractères majuscules=3
- Nombre de caractères minuscules=6
- Le nombre de caractères non alphabétiques=6
- Le somme de bonus = $15*4+(15-3)*2+(15-6)*3+6*5=141$
- 'Promo' est la plus longue séquence de caractères minuscules de taille=5
- 'SI' est la plus longue séquence de caractères majuscule de taille=2
- La somme des malus= $5*2+2*2=14$
- Le score final = $141-14=127 > 80$ donc ce mot de passe est considéré ' très fort '

Activité 2 :

Déployer une solution Python



Exercice 1 (suite) :

- Le nombre de caractères non alphabétiques=6
- Le somme de bonus = $15*4+(15-3)*2+(15-6)*3+6*5=141$
- 'Promo' est la plus longue séquence de caractères minuscules de taille=5
- 'SI' est la plus longue séquence de caractères majuscule de taille=2
- La somme des malus= $5*2+2*2=14$
- Le score final = $141-14=127 > 80$ donc ce mot de passe est considéré ' très fort'

Activité 2 : Déployer une solution Python



Exercice 1 (suite) :

Travail demandé :

1. Codage de l'application

1. Ecrire une fonction `NbCMin(pass)` qui retourne le nombre de caractères minuscules
2. Ecrire une fonction `NbCMaj(pass)` qui retourne le nombre de caractères majuscules
3. Ecrire une fonction `NbCApha(pass)` qui retourne le nombre de caractères non alphabétiques
4. Ecrire une fonction `LongMaj(pass)` qui retourne la plus longue séquence de lettres majuscules
5. Ecrire une fonction `LongMaj(pass)` qui retourne la plus longue séquence de lettres minuscules
6. Ecrire une fonction `score(pass)` qui calcule le score d'un mot de passe

2. Déploiement et téléchargement de l'application

1. Créer un packaging de l'application développée
2. Télécharger l'application sur TestPyPI

Activité 2 : Déployer une solution Python



Exercice 1 (suite) :

3. Création d'un fichier exécutable de l'application

1. Générer un fichier exécutable de l'application

4. Génération de documentation

1. Commenter le code développé en spécifiant pour chacune des fonctions: son objectif, ses paramètres et leurs types ainsi que le type de retour de la fonction en question
2. Créer un fichier htm contenant toute la documentation de votre application

Activité 2 : Correction



Exercice 1 :

1. Codage de l'application

```
def nbCMin(MP):  
    nb=0  
    for i in MP:  
        if 'a'<=i<='z': #vérifier les caractères Min  
            nb+=1 #incrémenter le nb de caractères Maj  
    return nb
```

```
def nbCMaj(MP):  
    nb=0  
    for i in MP:  
        if 'A'<=i<='Z': #vérifier les caractères Maj  
            nb+=1 #incrémenter le nb de caractères #Maj  
    return nb
```

```
def NbcAlpha(MP):  
    return len(MP)-nbCMaj(MP)-nbCMin(MP)
```

```
def longMaj(MP):  
    d=0  
    s=0  
    i=0  
    while i< len (MP): #parcourir le mot de passe  
        if 'A'<MP[i]<'Z': #vérifier les caractères Maj  
            s+=1 #incrémenter la taille de la séquence  
        else:  
            if s>d:  
                d=s  
                s=0  
            i+=1  
    return d
```


Activité 2 : Correction

Exercice 1 (suite):

```
def longMin(MP):  
    d=0  
    s=0  
    i=0  
    while i < len(MP): #parcourir le mot de passe  
        if 'a'<MP[i]<'z': #vérifier les caractères Min  
            s+=1  
        else:  
            if s>d:  
                d=s  
            s=0  
        i+=1  
    return d
```

```
def score(MP):  
    #Calcul bonus  
    bonus=len(MP)*4 + (len(MP)-nbCMaj(MP))*2+(len(MP)-nbCMin(MP))*3+ NbcAlpha(MP)*5  
  
    #Calcul pénalité  
    penalites=longMin(MP)*2+longMaj(MP)*2  
    val=bonus-penalites  
    if val<20:  
        print('très faible')  
    elif val<40:  
        print('faible')  
    elif val<80:  
        print('fort')  
    else:  
        print('très fort')  
  
if __name__ == '__main__':  
    MP='P@cSI_promo2017'  
    score(MP)
```

Exercice 1 (suite) :

2. Déploiement et téléchargement de l'application

1. Création d'un packaging de l'application développée

- Vérification de la dernière version de pip installée

```
py -m pip install --upgrade pip
```

- Création de la structure suivante du projet :

```
packaging_ScoreMP /  
|---- LICENSE  
|---- pyproject.toml  
|---- README.md  
|---- setup.py  
|---- src/  
|      |---- package_ScoreMP /  
|      |      |---- __init__.py  
|      |      |---- MPscore.py
```

Activité 2 : Correction

Exercice 1 (suite) :

- Création du fichier pyproject.toml

```
[build-system]
requires = [
    "setuptools>=42",
    "wheel"
]
build-backend = "setuptools.build_meta"
```

- Création du fichier setup.py

```
import setuptools

with open("README.md", "r", encoding="utf-8") as fh:
    long_description = fh.read()

setuptools.setup(
    name=« scoreMP-pkg-USER-NAME",
    version="0.0.1",
    author="Example Author",
    author_email="author@example.com",
    description="A small example package",
    long_description=long_description,
    long_description_content_type="text/markdown",
    url="https://github.com/pypa/sampleproject",
    project_urls={
        "Bug Tracker": "https://github.com/pypa/sampleproject/issues",
    },
```

Activité 2 : Correction



Exercice 1 (suite):

- Création du fichier README.m

```
# Projet scoreMP
```

```
Il s'agit d'un projet d'évaluation de la force d'un mot de pass
```

- Création d'une LICENCE
- Création des packages de distribution en exécutant cette commande sur l'invite de commande. La commande doit être exécutée à partir du même répertoire où se trouve **pyproject.toml** en utilisant la commande :

```
py -m build
```

Activité 2 : Correction

Exercice 1 (suite) :

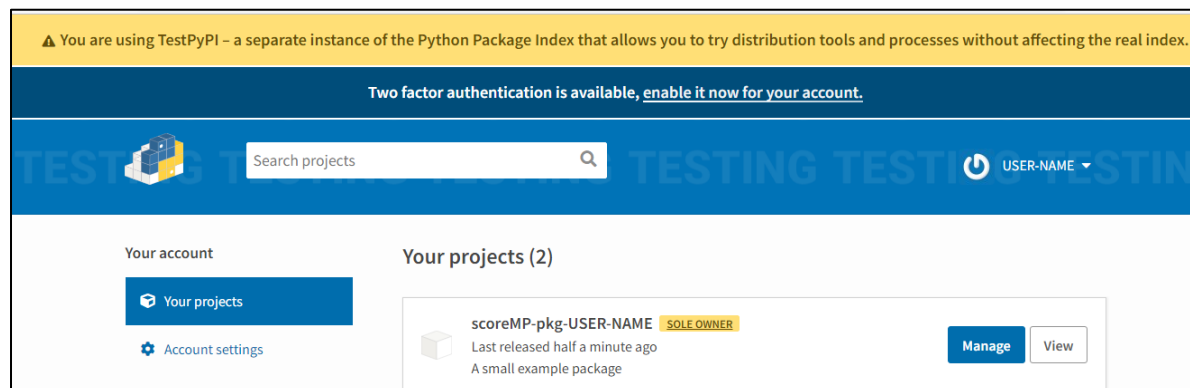
2. Téléchargement de l'application sur TestPyPI

- Création d'un compte sur TestPyPI (<https://test.pypi.org/account/register/>)
- Création un jeton sur <https://test.pypi.org/manage/account/#api-tokens>
- Installation de Twine en utilisant la commande :
- Exécution de Twine pour télécharger toutes les archives sous dist en utilisant la commande:

```
py -m pip install --upgrade twine
```

```
py -m twine upload --repository testpypi dist/*
```

<https://test.pypi.org/project/scoreMP-pkg-USER-NAME/0.0.1/>



Exercice 1 (suite):

3. Création d' un exécutable

- Installation de pywin32 en utilisant la commande:
- Installation de pyinstaller en utilisant la commande:
- Exécution de la commande suivante pour la génération de l'exécutable

```
pip install pywin32
```

```
pip3 install pyinstaller
```

```
pyinstaller MPscore.py
```

- La commande ci-dessus doit être exécutée dans le répertoire où se trouve MPscore.py

Le fichier exécutable est créé dans le répertoire:
\\src\\package_ScoreMP\\build\\MPscore

localpycos	2021-12-15 5:22	Dossier de fichiers	
Analysis-00.toc	2021-12-15 5:22	Fichier TOC	49 Ko
base_library	2021-12-15 5:22	Archive WinRAR ZIP	765 Ko
COLLECT-00.toc	2021-12-15 5:22	Fichier TOC	10 Ko
EXE-00.toc	2021-12-15 5:22	Fichier TOC	5 Ko
MPscore	2021-12-15 5:22	Application	1,960 Ko
MPscore.exe.manifest	2021-12-15 5:22	Fichier MANIFEST	2 Ko
PKG-00.pkg	2021-12-15 5:22	Fichier PKG	1,675 Ko
PKG-00.toc	2021-12-15 5:22	Fichier TOC	3 Ko
PYZ-00	2021-12-15 5:22	Python Zip Applic...	1,659 Ko
PYZ-00.toc	2021-12-15 5:22	Fichier TOC	37 Ko
warn-MPscore	2021-12-15 5:22	Document texte	4 Ko
xref-MPscore	2021-12-15 5:22	Microsoft Edge H...	404 Ko

Exercice 1 (suite) :

4. Création de la documentation

- Installation de Sphinx en utilisant la commande: `py -m pip install -U sphinx`
- Création d'un répertoire docs dans le projet pour contenir la documentation
- Rajout des commentaires dans le fichier ScoreMP.py contenant la description des différentes fonctions (exemple):

```
def nbCMin(MP):  
    """Cette fonction permet de compter le nombre de caractères minuscules  
    dans le mot de passe  
  
    :paramètre chaine de caractères: le mot de passe  
  
    :retour: le nombre de caractères minuscules dans le mot de passe  
    :rtype: entier  
    """  
    nb=0  
    for i in MP:  
        if 'a'<=i<='z':  
            nb+=1  
    return nb
```

```
def nbCMaj(MP):  
    """Cette fonction permet de compter le nombre de caractères  
    majuscules dans le mot de passe  
  
    :paramètre chaine de caractères: le mot de passe  
  
    :retour: le nombre de caractères majuscules dans le mot de passe  
    :rtype: entier  
    """  
    nb=0  
    for i in MP:  
        if 'A'<=i<='Z':  
            nb+=1  
    return nb
```

Activité 2 : Correction

Exercice 1 (suite) :

Spécifier le chemin du fichier
scoreMP.py

```
import os
import sys
# sys.path.insert(0, os.path.abspath('.'))
sys.path.append('C:/Users/DELL/package_ScoreMP/src/package_ScoreMP')
```

```
# -- Project information -----
project = 'scoreMP'
copyright = '2021, meriam'
author = 'meriam'
```

```
# The full version, including alpha/beta/rc tags
release = '1.0'
# -- General configuration -----
```

```
# Add any Sphinx extension module names here, as strings. They can be
# extensions coming with Sphinx (named 'sphinx.ext.*') or your custom
# ones.
```

```
extensions = [
'sphinx.ext.autodoc'
]
```

Ajout de cette ligne pour assurer
une génération automatique de la
doc

Exercice 1 (suite) :

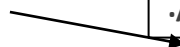
- Création du fichier MP.rt suivant dans le dossier source contenant un appel de toutes les fonctions du projet

```
.. autofunction:: MPscore.nbCMin  
.. autofunction:: MPscore.nbCMaj  
.. autofunction:: MPscore.NbcAlpha  
.. autofunction:: MPscore.longMaj  
.. autofunction:: MPscore.longMin  
.. autofunction:: MPscore.score
```

- Modification du fichier index.rt

```
Welcome to scoreMP's documentation!  
=====  
  
.. toctree::  
    :maxdepth: 2  
    :caption: Contents:  
  
./MP.rst
```

Ajout d'un lien vers MP.rt crée



Exercice 1 (suite):

- Exécution de la commande suivante :

```
.\make html
```

- Génération du fichier de documentation suivant dans docs\build\html

C:/Users/DELL/packaging_ScoreMP/docs/build/html/MP.html#MPscore.nbCMin

scoreMP

Navigation

Quick search

MPscore.nbCMin(MP)

Cette fonction permet de compter le nombre de caractères minuscules dans le mot de passe

Paramètre chaîne de caractères: le mot de passe

Retour: le nombre de caractères minuscules dans le mot de passe
Return type: entier

MPscore.nbCMaj(MP)

Cette fonction permet de compter le nombre de caractères majuscules dans le mot de passe

Paramètre chaîne de caractères: le mot de passe

Retour: le nombre de caractères majuscules dans le mot de passe
Return type: entier

MPscore.NbcAlpha(MP)

Cette fonction permet de compter le nombre de caractères non alphabétiques dans le mot de passe

Paramètre chaîne de caractères: le mot de passe

Retour: le nombre de caractères non alphabétiques dans le mot de passe
Return type: entier

MPscore.longMaj(MP)

Cette fonction permet de compter la longueur de la plus longue séquence de lettres majuscules dans le mot de passe