Thème 03 - Une histoire du vivant

Chapitre 05 - L'intelligence artificielle

QCM

1. c. / **2.** b. ET c. / **3.** a. ET d.

4 Vrai ou Faux ?

a. Faux. / **b.** Faux. / **c.** Vrai. / **d.** Vrai.

5 Phrases à construire

a. Une **image** numérique est composée de **pixels**, chaque pixel affiche une **couleur** codée sur trois **octets**.

b. Un **exécutable** est un **type** de fichier contenant des **informations** permettant de générer des **instructions**.

c. Pour certaines tâches, utiliser la **programmation** pour **coder** un **algorithme** est inefficace, il convient alors d'**entraîner** un **modèle** d'**IA**.

7 Faux positifs

1. Tableau de contingence rempli :

	Étoile	Rond
Bien classifié	15	27
Mal classifié	4	3

2. On veut sélectionner les étoiles, les faux positifs sont donc les ronds, identifiés à tort comme des étoiles : il y en a 3.

8 Repérer des erreurs dans un programme

1. Il manque la ponctuation : à la fin de la définition de la fonction « convertit_kelvin »

```
defconvertit_kelvin(tempCelsius):
```

Il manque une fermeture de parenthèses dans la ligne 2 *(toujours vérifier qu'il y a autant de parenthèses ouvertes et fermées).*

```
tempCelsius=float(input(«T (en °C) :»))
```

Il y a un signe égal en trop dans la ligne 5 *(le simple signe égal = est une affectation de valeur, alors que le double signe égal == est une comparaison).*

```
tempKelvin=tempCelsius + 273.15
```

Il faut utiliser un point et non une virgule pour séparer la partie entière de la partie décimale à la ligne 7.

```
cste=0.0028977
```

Il faut fermer les guillemets à la ligne 13 *(toujours vérifier qu'il y a un nombre pair de guillemets dans une ligne)*

```
print(«λmax (en nm) :»)
```

2. Ce fichier, dont l'extension est .py, contient le code source du programme. Les instructions contenues dans le code source ne peuvent pas être directement exécutées par le processeur, car Python est un langage « interprété ».

Sous Windows, seuls les fichiers se terminant par .exe peuvent être directement exécutés.

9 Du modèle à son exploitation

Conseils

Pour observer une dépendance entre deux variables, on peut tracer un nuage de points et observer leur distribution.

Méthode graphique : on peut tracer une droite au jugé et déterminer graphiquement l'équation.

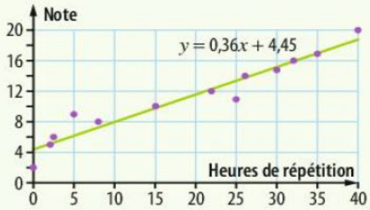
Méthode numérique : à l'aide d'un tableur, on peut déterminer automatiquement la courbe de tendance optimale et son équation (chapitre 12).

On utilise l'équation déterminée précédemment pour faire une prédiction.

Le tableau de contingence permet de connaître la précision d'un modèle, il suffit de compter les prédictions correctes et celles erronées.

Solution

1. On observe que les points sont globalement alignés. Il existe donc une dépendance affine.



2. On obtient l'équation $y = 0{,}36x + 4{,}45$.

3. Avec 10 heures de répétition, on prédit un score de $0{,}36 \times 10 + 4{,}45 = 8{,}05$ qui est en dessous de 10. On peut prédire que le participant échouera à la sélection.

Participant	Heures de répétition	Prédiction	Résultat réel	
Marcus	10	échec	échec	vrai négatif
Pauline	20	succès	succès	vrai positif
Liam	10	échec	succès	faux négatif
Ginko	30	succès	échec	faux positif

4. Ce qui donne le tableau de contingence suivant :

	positif	négatif
Vrai	¼	¼
Faux	¼	¼

5. Avec seulement 50 % de résultats pertinents (vrai positifs et négatifs), le test ne peut pas être considéré comme fiable. Cependant la taille de l'échantillon est trop faible et ne permet pas de conclure avec certitude.

10 Différents symptômes pour une même erreur

1. L'algorithme affiche 17 avec la première liste ; 20 avec la deuxième liste ; 14 pour la troisième liste et un message d'erreur pour la dernière liste.

Si l'on calcule « à la main » ces sommes : on trouve 17 ; 20 ; 18 et 20.

2. Les résultats diffèrent à cause des longueurs des listes. L'algorithme ne prend en compte que les cinq premiers termes de la liste et les additionne. Néanmoins, pour L2, cela donne le bon résultat car le terme manquant est un 0, ce qui n'est pas le cas pour L3.

3. Algorithme modifié :

```python
def total_animaux(L):
    n=len(L)
    S=0
    for k in range(n):
        S=S+L[k]
    return S
```

13 Le bug du Zune

1. Dans les deux premières lignes, on définit les variables Année et Jour.

On a ensuite une boucle conditionnelle « Tant que Jour > 365 ». L'algorithme entrera dans cette boucle par exemple lorsque la variable Jour vaudra 366 (ce qui est le cas du dernier jour d'une année bissextile, mais aussi du 1er janvier suivant une année non bissextile) ou 367 (ce qui est le cas du 1er janvier suivant une année bissextile).

Dans cette boucle, on distingue deux conditions selon que l'année qui se termine est bissextile ou non et dans chacun des cas, on ramène la variable Jour à 1 et on augmente Année de 1 (pour indiquer qu'on est le 1er janvier de l'année suivante).

2. Avec le 242ème jour d'une année non bissextile ou le 345ème jour d'une année bissextile, l'algorithme n'entrera jamais dans la boucle « Tant que (Jour > 365) », il ne se passera donc rien. Les variables Année et Jour resteront identiques à l'issue de cet extrait de code et ce seront les instructions qui suivent cet extrait (après la boucle) qui seront effectuées.

3. Le 366ème jour d'une année bissextile, l'algorithme entre dans la boucle « Tant que (Jour > 365) ». À l'intérieur de cette boucle, c'est la première condition qui est vérifiée (« Si Année est bissextile»). Mais, dans cette condition, le seul cas envisagé est « Si jour > 366 » (c'est-à-dire le jour 367, autrement dit le 1er janvier de l'année suivante). Or, à ce moment le jour vaut 366. Il ne se passe donc rien et le jour reste égal à 366, la boucle recommence alors puisque la condition « Jour > 365 » est toujours vérifiée et ainsi de suite. La boucle est alors exécutée indéfiniment, d'où le problème.

Une modification pour éviter cette erreur serait de programmer une sortie forcée de la boucle si jamais on entre dans le test de l'année bissextile, mais pas dans celui du Jour > 366 :

```
Année= année en cours
Jour= numéro du jour en cours
Tant que (Jour > 365) :
  Si Année est bissextile :
      Si Jour > 366 :
            On enlève 366 à Jour
            On augmente Année de 1
      Sinon :
            Arrêter la boucle
  Sinon :
      On enlève 365 à Jour
      On augmente Année de 1
```

14 Test de primalité

1. On peut par exemple tester ces programmes sur tous les entiers compris entre 2 et 9. Attention, 2 est un cas très spécial car c'est le seul nombre pair qui est tout de même premier. Quant à 9, c'est le plus petit nombre impair qui n'est pas premier.

2. L'idée de ces trois programmes est à peu près la même. On teste les nombres plus petits que n. Dès que l'un d'entre eux divise n, on peut dire que n n'est pas premier. Cependant, si aucun d'entre eux ne divise n, c'est que n est premier.

a) Le programme A donne un résultat faux avec $n = 9$. En effet, il renvoie True avec tous les nombres impairs. Le problème vient du fait que, dans la boucle, on teste d'abord la divisibilité de n par 2 ; si n est pair, alors le programme renverra False (ce qui est correct), mais SINON (c'est-à-dire pour n'importe quel nombre impair), le programme renverra alors True (ce qui est correct pour 3, 5 et 7, mais pas pour 9).

Remarque : pour $n = 2$, ce programme ne renverra rien du tout car range(2,n) est une liste vide vu que n est exclu.

b) Le programme B donne un résultat faux avec $n = 2$. En effet, il renverra toujours False, quel que soit le nombre n entré supérieur ou égal à 2. Le problème vient du fait qu'on commence la boucle avec $k = 1$. Or tous les entiers sont divisibles par 1 donc la condition « n%k==0 » sera vérifiée dès le début.

c) Le programme C donne également un résultat faux avec $n = 2$. En effet, il renverra toujours False, quel que soit le nombre n entré. Le problème vient du fait que, dans la boucle, k prend toutes les valeurs jusqu'à n INCLUS. Or tout entier est divisible par lui-même donc la condition « n%k==0 » sera vérifiée lorsque k vaudra n.

Une version corrigée de ce programme est :

```python
def primalite_correct(n):
    for k in range(2,n):
        if n%k==0:
            return False
    return True
```

15 Entraîner une voiture autonome

1. Obstacle → s'arrêter : [1,0,0]→[0,1,0]

Ligne droite → vitesse constante : [0,0,1]→[0,0,1]

2. Tableau complété :

Poids	Valeur
P_11	0
P_12	1
P_13	0
P_21	1
P_22	0
P_23	0
P_31	0
P_32	0
P_33	1

3. Le modèle ne pourra interpréter que les données en entrée de son modèle. Le panneau stop n'en faisant pas partie, il sera ignoré, la situation sera traitée comme un virage normal.

4. Soleil → pique-nique : [1,0]→[1,0]
Pluie → cinéma : [0,1]→[0,1]

Poids	Valeur
P_11	1
P_12	0
P_21	0
P_22	1

Exercice type Épreuve commune de contrôle continu

1a. D'après le tableau, les Crustacés présentent 17 % d'espèces menacées contre 28 % pour les Coraux et récifs. Les Crustacés semblent *a priori* moins touchés par le risque d'extinction. Calculons le nombre d'espèces plus précisément.

Chez les Crustacés, on dénombre donc environ 17 % d'espèces pour les quatre catégories demandées. Pour un total de 2 872 espèces, il y a donc environ 488 espèces menacées (0,17 × 2 872). Par le même raisonnement, Coraux et récifs possèdent environ 237 espèces menacées (0,28 × 845).

Le nombre d'espèces en danger est donc en réalité plus grand chez les Crustacés que chez les Coraux et récifs.

1b. Il est indéniable que, sur la base de cette figure, le groupe des Amphibiens par exemple présente sept fois plus d'espèces en danger que le groupe des Poissons osseux. Cependant, ces chiffres ne doivent pas masquer que :

– les 10 % énoncés ne tiennent pas compte de l'insuffisance des données, car (1) le risque est peut-être plus important pour les Poissons osseux, compte-tenu des 10 % d'espèces dont le statut est indéterminé, ce qui, dans le pire des cas et en intégrant les espèces vulnérables, représenterait jusqu'à presque 20 % des espèces prises en compte ; et (2) le nombre d'espèces considérées est faible en regard du nombre total d'espèces de Poissons osseux déjà décrites (de l'ordre de 30 000) ;

– le risque d'extinction est toujours grave, car la disparition d'une espèce correspond à la perte d'un patrimoine génétique unique (assemblage d'allèles), qui n'existe pas dans une autre espèce ;

– la disparition d'une seule espèce peut entraîner des déséquilibres importants dans l'écosystème du fait de la place et de la fonction qu'elle y occupe.

2a. La technique de comptage utilisée s'appelle la méthode de « capture-marquage-recapture ».

Si l'on suppose qu'il n'y a ni mortalité, ni natalité, ni immigration, ni émigration entre les deux sessions de capture (aucun facteur susceptible de faire varier les effectifs), il est possible d'estimer l'effectif de la population totale du Grand Hamster d'Alsace par la formule :

$N = \dfrac{205 \times 208}{43} = 991{,}6$, soit environ 992 individus.

Voir le Mémo MATHS « Quatrième proportionnelle » de l'activité 1 du chapitre 9.

2b. **Champ de Luzerne.** Les taux de survie et le nombre d'émergence sont les plus faibles dans les champs de luzerne : la mortalité atteint 15 % après quelques heures seulement et atteint 80 % après une trentaine de jours. L'émergence d'une première portée n'y est observée que deux fois.

Champ de blé. Au contraire, dans le blé récolté, la mortalité est de moins de 25 % jusqu'au cinquantième jour, mais elle chute brutalement dès la moisson pour atteindre des taux de survie aussi faibles que dans la luzerne dès le soixantième jour (moins de 20 % de survie). L'émergence d'une première portée y est observée six fois, avec une première fois plus précoce de 10 jours par rapport à la première émergence d'une portée dans la luzerne.

Dans le blé sur pied, le taux de survie se stabilise en 15 jours autour de 55 % de taux de survie et ne diminue qu'au bout d'une centaine de jours à environ 40 %. Ce temps long sans mortalité supplémentaire permet huit émergences de première portée et cinq de deuxième portée.

D'après ces observations, la survie sur un temps compatible avec la reproduction, l'émergence de première portée et d'une seconde génération dans l'année sont donc nettement favorisées dans le blé sur pied. La prédation (par des Renards, des Mustelidés ou des Oiseaux) étant la principale cause de mortalité des Hamsters, on peut supposer que le blé sur pied constitue le meilleur refuge pour cette espèce, par comparaison avec le blé moissonné qui expose probablement beaucoup les individus et la luzerne qui représente à tout moment un habitat où ils seraient vulnérables (végétation plus rase, plus éparse, semis plus tardif, croissance végétative plus tardive par rapport aux dates de reproduction des Hamsters).

Remarque : la source est ONCFS et non ONCES.

3a. Il faut admettre que les nombres d'espèces d'oiseaux et de mammifères sont également triplées avec les mesures de sauvegarde prises.

En considérant que N est le nombre d'espèces sauvegardées, il faut transcrire en langage mathématique « tripler le nombre d'espèces protégées correspond à 3742 espèces supplémentaires », ce qui donne : $3N = N + 3\,742$

On a alors $2N = 3\,742$ et donc $N = \frac{3\,742}{2} = 1\,871$ espèces.

Il y avait donc 1 871 espèces sauvegardées avant l'augmentation de 5 % des zones.

3b. Si pour chaque hausse de 5 %, on triple le nombre d'espèces, alors en faisant deux hausses successives, on triple deux fois, ce qui revient à multiplier par 9.

Il y aura donc $3\,742 \times 3 = 11\,226$ espèces.

4. En 57 millions d'années, l'espèce ancestrale a évolué et est devenue l'espèce actuelle de l'Ibis géant, grâce à des innovations génétiques entraînant des variations de l'ADN (nouveaux allèles). La sélection naturelle (sélection utilitaire et sélection sexuelle) s'est exercée sur des variants individuels, ainsi peut-être que le hasard ou la dérive génétique dans le cas de petits effectifs. Les adaptations qui ont permis à cette espèce de perdurer dans le temps et se reproduire, ainsi que le hasard, ont façonné l'évolution de cette espèce jusqu'à aboutir à ce qu'elle est aujourd'hui.

5. Les propositions de protection des espèces peuvent être très différentes selon que l'on considère des espaces naturels à l'échelle mondiale ou des espèces rares en France.

À l'échelle mondiale, la création d'aires protégées, même en faible proportion (5 %), mais judicieusement spatialisés, donnerait des résultats potentiels efficaces : la variété des espèces protégées serait multipliée par 3. Pour en définir l'emplacement géographique d'une manière optimisée, il est nécessaire de ne pas s'intéresser uniquement à la richesse en espèces, mais de prendre également en considération la diversité fonctionnelle et la diversité évolutive.

Si les simulations montrent clairement que l'augmentation des surfaces protégées donnerait des résultats sur un grand nombre d'espèces, il reste cependant à choisir si l'on souhaite protéger l'ensemble des espèces d'un espace ou plus spécifiquement des espèces rares (d'oiseaux et mammifères) quelle que soit la richesse de leur milieu. Ce choix influence fortement la géographie des espaces à protéger.

Des difficultés existent à l'issue de ces propositions. Les espaces identifiés sont souvent économiquement pauvres et leur donner un statut de protection fort les rend peu compatibles avec certains projets d'aménagement ou d'exploitation, notamment forestière. Pour faire respecter leur statut de protection, les moyens au sein de ces pays pauvres seraient probablement insuffisants (nombre de gardes par exemple).

En France, le problème du Grand Hamster semble être très différent. La pérennité de cette espèce au sein de territoires très aménagés dépend de cultures agricoles. Les deux problématiques se rejoignent sur le fait que la protection des espaces, naturels ou anthropiques permet la protection d'espèces retenues selon un choix sociétal. Un autre point commun est que c'est la bonne connaissance des écosystèmes et de la biologie des animaux à protéger qui permettent de définir une protection efficace.