**AFCI Formation**

**Guide pour la scénarisation pédagogique des modules digitaux**

1. **Mode d’emploi**

**Objet du présent fichier**

Concevoir un module digital, c'est beaucoup écrire ! Les formations sont d'abord écrites sur des documents extérieurs aux outils de digitalisation. Les jeux de relecture sont facilités, vous gardez la maîtrise sur le contenu que vous proposez et la médiatisation peut se faire sur une base sûre.

Cette trame se compose de plusieurs parties (présentation + X sections) formant la base d'un module digitalisé standard.

**Mode d'emploi**

1. Déterminez le nombre de sections dans votre module

2. Si vous avez + de 3 sections, copiez-collez autant de descriptif de section qu’il vous manque.

3. Adaptez la partie 3 de l'onglet présentation de votre formation pour qu'il puisse accueillir le nombre de sections voulues

4. Commencez à remplir l'onglet présentation :

* Auteur (a savoir…vous !)
* Domaine (Par exemple : Sanitaire et social, numérique,...)
* Certification concernées (les diplômes qui peuvent être concernés par ce module)
* Compétences ciblées
* Référence technique : laissez à Thomas ou à l’ambassadeur le soin de le compléter 😉

5.Continuez par l’en-tête :

* Titre (titre du module, quoi d’autre ?)
* Image/GIF/Vidéo (choisissez des images libres de droit, par exemple sur les sites “FREEPIK” ou “PIXABAY”, pour les vidéos youtube, copiez-collez simplement le lien)
* Introduction/résumé : Présentez en quelques lignes le module aux stagiaires, comme vous pourriez le faire en présentiel, une touche d’humour ou de sympathie est fortement appréciée !
* Objectif : ce sont les objectifs pédagogiques, ce que le stagiaire doit apprendre, ils commencent toujours par : “A la fin de ce module de formation, vous serez capable de…” suivit d’un verbe d’action, au besoin, fiez-vous à la taxonomie de Bloom
* Consignes spécifiques : s’il y en a, par exemple : pendant tout le module, ayez avec vous votre carnet XXX…” Sinon laissez cette case vide
* Durée indicative : combien de temps faut-il, à peu près, pour terminer le module

6. Remplissez le tableau “Plan du module”, cela vous permettra de voir plus clair dans les différents chapitres, au besoin, ajoutez des sections

7. Remplissez les sections

Complétez la présentation de la section, si besoin, référez-vous au point 5

8. Par section, déterminez le nombre d'activités.

Attention, une section peut très bien ne contenir qu'une seule activité

9. Commencez à remplir vos activités en rédigeant le contenu.

10. Si vous avez déjà une idée, vous pouvez choisir dès à présent le type d'activité

11. A chaque section son évaluation, elle n’est pas forcément là pour sanctionner le stagiaire, mais plutôt pour lui permettre de s’assurer qu’il à bien compris. Vous devez proposer à minima un exercice par section, choisissez s’il s’agira d’un texte ou d’un devoir, ne remplissez que la partie concernée.

12. Lorsque toutes les sections sont créées, compléter la dernière partie : Section finale

13. Dans la section finale, rédigez le résumé du module, il reprend les axes essentiels à retenir, ajoutez l’image si vous le souhaitez

14. Créer l’évaluation finale du module

15. Une fois l'ensemble des chapitres sections terminés, retournez sur la présentation et vérifiez l'exactitude des informations du plan

**Félicitations ! Votre module de formation est prêt à être digitalisé !**

1. **Lexique**

Activité :

Consigne :

Consigne pédagogique :

Consigne opératoire/opérationnelle :

Devoir :

Evaluation :

Module :

Objectif :

Section :

Test :

1. **Trame d’écriture**
2. **Présentation du module :**

**Auteur : Didier ROLLAND**

**Domaine :** Numérique

**Certifications concernées : TP DWWM**

**Compétences ciblées :**

**Références techniques :**

1. **En-tête**

**Titre : L’algorithmie**

**Image/ GIF/ Vidéo d’introduction :**

**Introduction/ Résumé du module :**

**Objectif :** A l’issue de ce module de formation, vous serez capable de

**Consignes spécifiques :**

**Durée indicative :**

1. **Plan du module**

|  |  |
| --- | --- |
|  | **Titre** |
| **Section 1** | Définir la notion d’algorithme et son but |
| **Section 2** | Présentation d’un algorithme avec les différents éléments |
| **Section 3** |  |
| **Section 4** |  |
| **…** |  |

1. **Section 1**
   1. **Présentation de la section**

**Titre :** Définir la notion d’algorithme et son but

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :** Nous allons voir ce

**Objectifs :** Comprendre à quoi va servir un algorithme

**Consignes spécifiques :**

**Durée indicative :** 15 mn

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | Un algorithme est une suite d'opérations, qu'on nomme instructions, à effectuer pour atteindre un objectif (résoudre un problème, cuisiner un plat, faire se déplacer un robot, etc.). On dit que ces instructions doivent être exécutées.  En informatique, ou dans le cadre de la création de systèmes qui ont une composante numérique, on réalise des algorithmes afin de concevoir, ou de décrire, un programme informatique.  L'algorithmique est l'ensemble des principes, règles et techniques qui permettent d'écrire un algorithme.  *Remarque :*  *Bien que l'informatique soit assez récente, l'algorithmique existe depuis longtemps. Elle tire ses origines des méthodes de résolution d'équations mathématiques mises au point par le mathématicien arabe Al-Khwârizmî au IXe siècle. Le mot « algorithme » est d’ailleurs issu du nom d'Al-Khwârizmî.*  Un algorithme n'est pas encore un programme informatique, c'est simplement un mode opératoire, une série d'instructions structurées.  Une fois l'algorithme écrit et validé, il faut donc, dans un second temps :   * Le traduire dans un langage informatique (C++, Python, Javascript, etc.) ; * Éventuellement, l’adapter aux composants matériels du système numérique.   On obtient alors un programme informatique qui implémente l’algorithme.  **Le pseudo-code**  Méthodes pour créer un algorithme, plusieurs méthodes permettent de créer un algorithme.   * On peut écrire l'algorithme dans un langage appelé pseudo-code. Le pseudo-code ressemble à un langage humain simplifié et formaté. * On peut dessiner un algorigramme. Un algorigramme est un algorithme sous forme de diagramme. * On peut écrire l'algorithme directement dans un langage informatique (C, PHP, Java, etc.).   Avantages du pseudo-code et de l’algorigramme  Le pseudo-code et l'algorigramme sont à prioriser lorsqu'on conçoit un programme informatique.  Leurs avantages sont les suivants.   * Ils ne dépendent pas d’un langage de programmation : pseudo-code et algorigramme sont universels, ils n'utilisent aucun principe ou règle propre à un langage de programmation en particulier. * Ils sont plus lisibles : les langages informatiques contiennent des symboles spéciaux ({, $, ", etc.) et des conventions d'écriture parfois contre-intuitives. Cela peut gêner la lisibilité de l'algorithme, surtout si l'algorithme doit par la suite être présenté à des non-spécialistes.   Avantages du pseudo-code par rapport à l’algorigramme, les atouts du pseudo-code, qui font qu'on le choisit souvent plutôt que l’algorigramme, sont les suivants.   * Il conserve une bonne lisibilité, même lorsque l’algorithme s'allonge et que sa structure se complexifie. * Il est plus adapté pour présenter de l’information textuelle (par exemple, pour écrire des phrases qu’un ordinateur exécutant l'algorithme écrirait sur l’écran). * La traduction vers un langage de programmation est plus directe, car les langages de programmation ont globalement la même structure que le pseudo-code. * Il est plus adapté pour la conception de programmes purement informatiques, dédiés aux ordinateurs, mobiles, web, etc. | Leçon |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 2**
  2. **Présentation de la section**

**Titre :** Présentation d’un algorithme avec les différents éléments

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :** Nous allons

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :** 3 h 45 mn

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** | Une **variable** permet de mémoriser et de manipuler des données dans l’algorithme. Une variable possède un **nom** et un **type**. Elle est déclarée en début de programme, précédée par le mot-clé **Variable**.  Les principaux **types de variables** sont les suivants.   * **Nombre**, pour les variables qui contiennent des nombres. * **Booléen**, pour les variables qui contiennent des valeurs binaires, qui peuvent valoir vrai ou faux. * **Chaine**, qui vient de « chaine de caractères » pour les variables qui contiennent des caractères alphanumériques, des mots, des phrases, etc.   *Remarque :*  *Les conventions d’écriture varient d'un auteur à l'autre, d'un manuel à l'autre : il n'y a pas une seule et unique façon d'écrire en pseudo-code.*  Voyons des exemples :  **Variable** age, prix : **nombre**  **Variable** estMajeur : **booléen**  **Variable** nom, prenom : **chaine**   * Le type **nombre** **est parfois scindé en un type** ***entier*** pour les nombres entiers, et en un type ***réel*** pour les nombres réels. * On essaie toujours de donner **des noms explicites aux variables**, pour que quelqu'un qui lit l'algorithme comprenne immédiatement leur rôle.   L'instruction **d’affectation** s'écrit en indiquant le nom de la **variable**, suivi de l'opérateur d'affectation ← et de la nouvelle **valeur** à donner à la variable. Cette valeur doit respecter le type de la variable.  Encore des exemples :  Age ← 28  Prix ← 5,76  estMajeur ← faux  Prénom ← Didier  Il est aussi possible d'affecter **une expression mathématique** à une variable de type nombre. Les expressions peuvent contenir des valeurs numériques, d'autres variables de type nombre, ou un mélange des deux.  Les expressions mathématiques s'écrivent avec les opérateurs mathématiques suivants.   * ‘ **+** ‘ qui est l’opérateur d'addition. * ‘ **–** ‘ qui est l’opérateur de soustraction. * ‘ **\*** ‘ qui est l’opérateur de multiplication. * ‘ **/** ‘ qui est l’opérateur de division. * ‘ **(** ‘ et ‘ **)** ‘ qui permettent de structurer les expressions complexes.   De nouveau des exemples :  prixPromo ← prix \* 0,6  Volume ← (longueur \* largeur) \* hauteur  Le système numérique qui exécute l'algorithme aura surement besoin d'interagir avec l’extérieur pour :   * Dialoguer avec un utilisateur humain ; * Récupérer des informations qui viennent de capteurs ou piloter des actionneurs (moteurs, pompe, lampe, etc.) ; * Communiquer sur un réseau de données.   L'instruction de **lecture** permet au système d'interagir avec l’extérieur, elle se note avec le mot-clé **Lire**.  Cette instruction permet au système numérique :   * De demander à l’utilisateur de lui fournir une information. * De récupérer des données qui proviennent d’un capteur. * De recevoir des données qui proviennent d’un réseau.   La valeur de cette information est affectée à une **variable**, dont le nom suit le mot-clé **Lire**.  *On parle aussi d’instruction* ***d’entrée****, car Lire permet de faire entrer une information dans le système.*  Par exemples :  **Lire** age  **Lire** prix  L'instruction **d'écriture** permet au système d'interagir avec l’extérieur, elle se note avec le mot-clé **Écrire**.  Cette instruction permet au système numérique :   * D’afficher une information à destination de l’utilisateur. * D’envoyer une commande à un actionneur. * D’envoyer des données sur un réseau.   Après le mot **Écrire**, on indique ce qui sera transmis. Cela peut être la **valeur d'une variable**, **une chaine** ou **un mélange des deux**.  *On parle aussi d’instruction de* ***sortie****, car Écrire permet de faire sortir une information du système.*  Exemples :  Ecrire « quel est votre nom ? »  Ecrire « votre prénom est : » , prénom , « . »  *Remarques :*  *Dans le cas où le système est un ordinateur, ce qui sera le cas dans les prochains exemples, l’information est affichée sur l’écran.*   * ***La valeur écrite doit être de type chaine****.* * *Plusieurs chaines peuvent être écrites à la suite, en les séparant par des virgules.* * *Pour ne pas se tromper sur le rôle de l’instruction lecture et le rôle de l’instruction écriture, il faut se rappeler que l’algorithme est destiné à être exécuté par un système numérique. C’est donc le point de vue du système qu’il faut adopter quand on écrit les instructions. Quand le système exécute l’instruction* ***Lire****, il récupère ainsi une information. Quand le système exécute l'instruction* ***Écrire****, il écrit quelque chose à l’écran.*   Voici ci-dessous un algorithme complet qui utilise les instructions de lecture et d’écriture. Le système pour lequel cet algorithme est dédié est un ordinateur. Il demande à l'utilisateur son prénom et son nom, puis le salue à l'écran.  **Variable** prénom, nom : **chaines de caractères**  **Début**  **Écrire** “Quel est votre prénom ?”  **Lire** prénom  **Écrire** “Quel est votre nom ?”  **Lire** nom  **Écrire** “Bonjour ” , prénom ,“ ” , nom  **Fin**  Cet exemple permet de voir la structure complète d’un algorithme.  On note les éléments suivants.   * On utilise les mots-clés **Début** et **Fin**, pour encadrer l'algorithme à proprement parler. Les variables sont déclarées juste avant, à part. * Les instructions de l’algorithme sont **décalées vers la droite** par rapport aux mots **Début** et **Fin**. C’est ce qu’on appelle **l'indentation**. L’indentation permet de rendre l'algorithme plus lisible en mettant en évidence les groupes d’instructions de même niveau.   Une **condition** est une expression qui peut prendre l'une des deux valeurs suivantes **vrai** ou **faux**. On dit également que c'est une valeur de type "**logique**" ou "**booléen**".  Les principaux **opérateurs** de comparaison que vous rencontrerez sont les suivants :   * égal à ( = en pseudo code) * différent de ( != en pseudo code) * strictement supérieur ( > en pseudo code) * strictement inférieur ( < en pseudo code) * supérieur ou égal ( > = en pseudo code) * inférieur ou égal ( < = en pseudo code)   Ces comparaisons n'ont un sens que si les variables que l'on **compare** sont de **même type**.  Comment écrire cette condition :  Si *condition* alors  *Instructions*  Fin si  Dans ce cas vous pouvez écrire : si a > 10 alors on multiplie a par 2  Si a>10 alors  a ← a \* 2  fin si  On peut écrire des **conditions** plus **complexes** en reliant des comparaisons à l'aide des **opérateurs logiques ET, OU et NON**.   * **Condition 1 ET condition 2** sera vraie si les deux conditions sont **toutes les deux vraies**.   Par exemple, la condition : "âge supérieur à 5 ET âge inférieur à 10" sera vraie si la variable âge est strictement compris entre 5 et 10.   * **Condition 1 OU condition 2** sera vraie si **l'une au moins des deux conditions est vraie**.   Par exemple, la condition "prénom=Jean OU nom=Dupont" sera vraie pour :   * Jean Dupont (conditions 1 et 2 vraies) * Jean Durand (condition 1 vraie) * Pierre Dupont (condition 2 vraie)   Mais fausse pour   * Pierre Durand (conditions 1 et 2 fausses) * NON (condition 1) sera vraie si et seulement si condition 1 est fausse. * Par exemple : "NON (x < 3)" sera vraie si x > = 3   Exemple : x est un entier, saisir une valeur de x, si x > 10 alors x sera égal a 10, afficher x  **Variable** x : **entier**  **Début**  **Afficher** « entrez un nombre : »  **Lire** x  **Si** x>10 **alors**  x ← 10  **fin** **si**  **afficher** x  **fin**  Allons plus loin en utilisant si et sinon. Cela veut dire que l’on peut écrire si x>10 alors x=10 sinon x=0, voyons l’algorithme :  **Variable** x : **entier**  **Début**  **Afficher** « entrez un nombre : »  **Lire** x  **Si** x>10 **alors**  x ← 10  **sinon**  x ←0  **fin** **si**  **afficher** x  **fin**  Un algorithme un plus étoffé :  **Variables** âge, prix : **entier**  **début**  **afficher** "entrez votre âge :"  **lire** âge  **si** âge < 16 **alors**  prix ← 10  **afficher** "vous bénéficiez du tarif réduit"  **sinon**  prix ← 15  **afficher** "vous ne bénéficiez pas du tarif réduit"  **fin** **si**  **afficher** "vous devez payer : ", prix, " €"  **fin**  Et ça y est, on y est, on est arrivés, les voilà, c’est les **boucles**. Si vous voulez épater vos collègues, vous pouvez également parler de **structures répétitives**, voire carrément de **structures itératives**. Bon, vous faites ce que vous voulez, ici on est entre nous, on parlera de boucles.  Les **boucles**, c'est généralement le point douloureux de l'apprenti programmeur. C'est là que ça coince, car autant il est assez facile de comprendre comment fonctionnent les boucles, autant il est souvent long d'acquérir les réflexes qui permettent de les élaborer judicieusement pour traiter un problème donné.  Prenons le cas d’une saisie au clavier (une lecture), où par exemple, le programme pose une question à laquelle l’utilisateur doit répondre par O (Oui) ou N (Non). Mais tôt ou tard, l’utilisateur, facétieux ou maladroit, risque de taper autre chose que la réponse attendue. Dès lors, le programme peut planter soit par une erreur d’exécution (parce que le type de réponse ne correspond pas au type de la variable attendu) soit par une erreur fonctionnelle (il se déroule normalement jusqu’au bout, mais en produisant des résultats fantaisistes).  Alors, dans tout programme un tant soit peu sérieux, on met en place ce qu’on appelle un **contrôle de saisie**, afin de vérifier que les données entrées au clavier correspondent bien à celles attendues par l’algorithme.  Allez, voyons ça en algorithme :  **Variable** Rep : **chaine**  **Début**  **Ecrire** "Voulez-vous un café ? (O/N)"  **Lire** Rep  **Si** Rep <> "O" **ET** Rep <> "N" **Alors**  **Ecrire** "Saisie erronnée. Recommencez"  **Lire** Rep  **Fin Si**  **Fin**  Ce genre de test sera bien si vous avez un, deux voir 3 test a faire sinon le code va vite s’allonger alors prenon un autre test plus adéquat comme le « tantque »  **Tantque** *booléen*  *Instructions*  **Fin** **tantque**  **Variable** Rep : **chaine**  **Début**  **Ecrire** "Voulez-vous un café ? (O/N)"  **Lire** Rep  **tantque** Rep <> "O" **ET** Rep <> "N"  **Lire** Rep  **Fin tantque**  **Fin**  On se trouve maintenant dans un code correct mais autre problème qui va se poser, si l’on initialise pas notre variable « rep », cette variable pourra éventuellement contenir une des deux réponse donc ne pas être testée correctement et aussi non n’avons pas de message pour l’utilisateur si la sais est valide ou pas, changeons ça.  **Variable** Rep : **chaine**  **Début**  **Rep** ← ‘’ x ‘’ *(valeur par défaut)*  **Ecrire** "Voulez-vous un café ? (O/N)"  **Lire** Rep  **tantque** Rep <> "O" **ET** Rep <> "N"  **Ecrire** "Vous devez répondre par O ou N. recommencez"  **Lire** Rep  **Fin tantque**  **Ecrire** "Saisie acceptée."  **Fin**  Regardons maintenant une boucle aussi très utilisée en programmation : la boucle « **pour** »  On va écrire un algorithme qui demande un nombre de départ (1 à 10), et qui ensuite écrit la table de multiplication de ce nombre, présentée comme suit (cas où l'utilisateur entre le nombre 7) :  Table de 7 :  7 x 1 = 7  7 x 2 = 14  7 x 3 = 21  …  7 x 10 = 70  **Variables** N, i : **Entier**  **Début**  **Ecrire** "Entrez un nombre : (1 à 10) "  **Lire** N  **Ecrire** "Table : " , N  **Pour** i ← 1 à 10  **Ecrire** N, " x ", i, " = ", n\*i  i **Suivant**  **Fin**  Une **fonction** permet de réaliser une tâche complexe, en écrivant un seul mot.   1. Une **fonction** reçoit **aucun**, **un** ou **plusieurs arguments**, qu’on transmet à la fonction.   Un argument est une valeur ou une variable.   1. Une **fonction** réalise une **série d’opérations** à partir des arguments : calculs, conversions, manipulations de chaine, etc. 2. Une **fonction** renvoie ensuite un **résultat**, qui est une valeur.   Ce résultat peut être affecté à une variable, utilisé dans une instruction de lecture ou d’écriture.  Voici quelques exemples de fonctions.   * **max**(arg1,arg2) renvoie le plus grand nombre parmi les deux nombres passés en arguments. * **alea**(arg1,arg2) donne un nombre entier aléatoire, compris entre arg1 et arg2. * **long**(arg) calcule la longueur, en nombre de caractères, d’une chaine. * **concat**(arg1,arg2) met bout à bout deux chaines. * **chaine**(arg) convertit un nombre ou un booléen en chaine. * **Nombre**(arg) convertit une chaine en nombre.   *Remarques :*   * *Il n’y a pas de fonctions universelles en pseudo-code. Les fonctions qui peuvent être utilisées seront indiquées dans l’énoncé.* * *Les fonctions sont en réalité des morceaux de code déjà écrits, parfois longs et complexes, qu’on utilise pour faciliter l’écriture d’algorithmes plus élaborés.*   Voyons comment utiliser ces fonctions :  L'ordinateur qui exécute l’algorithme ci-dessous demande à l'utilisateur la longueur et la largeur d'un rectangle, il calcule l'aire du rectangle, puis il l’écrit sur l'écran.  **Variable** longueur, largeur, aire : **nombre**  **Début**  **Écrire** “Quelle est la longueur du rectangle ?”  **Lire** longueur  **Écrire** “Quelle est la largeur du rectangle ?”  **Lire** largeur  aire ← longueur \* largeur  **Écrire** “L'aire du rectangle vaut ”, **chaine**(aire),“ m2.”  **Fin**  *Remarque :*  *La fonction* ***chaine****(arg) permet de convertir la valeur de la variable aire (de* ***type nombre****) en* ***chaine****. Cette fonction est indispensable, car l’instruction* ***d’écriture*** *ne peut afficher que des* ***chaines de caractères****, or aire est un nombre.*  Le programme ci-dessous propose à l’utilisateur de générer un nombre aléatoire, entre deux bornes choisies par l’utilisateur.    **Variable** borneInf, borneSup, nombreAleatoire : **nombre**  **Début**  **Écrire** “Je vais générer un nombre aléatoire entre deux bornes.”  **Écrire** “Veuillez entrer la borne inférieure.”  **Lire** borneInf  **Écrire** “Veuillez entrer la borne supérieure.”  **Lire** borneSup  nombreAleatoire ← **alea**(borneInf,borneSup)  **Écrire** “Le nombre aléatoire est : “,  **chaine**(nombreAleatoire)  **Fin**  Pour aller encore plus loin, vous pouvez aussi écrire des fonctions personnalisées. En effet, Il est possible d’écrire ses propres fonctions. Ce sont en fait des petits algorithmes, séparés de l’algorithme principal.  Plutôt que de commencer par le mot-clé **Début**, une fonction personnalisée commence par le mot-clé **Fonction**, suivi du nom qu’on donne à la fonction, puis des éventuels arguments de la fonction.  Le mot-clé **Renvoyer** permet de préciser la valeur que la fonction renvoie.  La fonction peut posséder ses propres variables internes.  Voyons ça dans un exemple :  Voici l’écriture d’une fonction qui permet de calculer le **cube** d’un nombre. Elle se nomme **cube**, et prend un seul argument, de type **nombre**, qu’on nomme **arg**. La valeur renvoyée est la variable résultat, de type nombre.  **Variable** resultat : **nombre**  **Fonction** **cube**(arg : nombre)  resultat ← arg \* arg \* arg  **Renvoyer** resultat  **Fin** | Leçon |
| **Activité 2** |  | Leçon |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 3**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :** Nous allons

**Objectifs :** Vous

**Consignes spécifiques :**

**Durée indicative :** 10 mn

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Leçon |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Section 3**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :** Savoir

**Consignes spécifiques :**

**Durée indicative :** 30 mn

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Atelier |
| **Activité 2** |  | Fichier |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **Q4** |  |  |  |
| **Q5** |  |  |  |
| **Q6** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section 2**
   1. **Présentation de la section**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :**

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Type d'activité |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |
| **…** |  | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section 3**
   1. **Présentation de la section**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :**

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Type d'activité |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |
| **…** |  | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section 4**
   1. **Présentation de la section**

**Titre :**

**Image/Gif/Vidéo d’introduction :**

**Introduction/Résumé de la section :**

**Objectifs :**

**Consignes spécifiques :**

**Durée indicative :**

* 1. **Contenu**

|  |  |  |
| --- | --- | --- |
|  | **Contenu** | **Type d’activité** |
| **Activité 1** |  | Type d'activité |
| **Activité 2** |  | Type d'activité |
| **Activité 3** |  | Type d'activité |
| **Activité 4** |  | Type d'activité |
| **Activité 5** |  | Type d'activité |
| **…** |  | Type d'activité |

* 1. **Evaluation de la section**

Au choix :

* + 1. **Test**

Pour les tests intermédiaires, privilégiez les tests courts, autour de 5 questions

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

1. **Section finale**
   1. **Evaluation du module**

Pour l’évaluation du module, l’exercice final portera sur l’ensemble du contenu des différentes sections, il ne devra pas se concentrer uniquement sur une section.

Au choix :

* + 1. **Test**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Question** | **Réponses possibles** | **Bonne réponse** |
| **Q1** |  |  |  |
| **Q2** |  |  |  |
| **Q3** |  |  |  |
| **…** |  |  |  |

* + 1. **Devoir**

**Consigne pédagogique :**

**Consigne opératoire :**

* + 1. **Autre activité**

**Renseignez**

* 1. **Résumé du module et conclusion**

**Résumé et conclusion** :