

BLACKBOXAI

Generated on: 14/08/2025 16:22:01

User

PREMIÈRE PARTIE Les notions essentielles	1
CHAPITRE 1 Comment un site web fonctionne-t-il ?	3
Web ?	4 Fonctionnement d'un site web
statique	5 Le protocole HTTP
..... 7 L'URL, adresse d'une page web	
..... 8 Le serveur web : à la fois ordinateur et logiciel	
..... 9 Des sites web qui se mettent à jour tout seuls	
..... 9 Fonctionnement d'un site web dynamique	
..... 11 Utilisation d'une base de données	
..... 12 Passage de paramètres à une page web	
..... 13 Ai-je bien compris ?	
..... 14 CHAPITRE 2 Programmation orientée objet et framework	
MVC..... 15 Des programmes modulaires	
... 16 Une écriture simplifiée.....	16 Des
modules réutilisables	16 Un travail d'équipe
facilité et plus efficace	16 Les langages orientés objet
..... 17 Les cas d'utilisation (use cases) et le	
MVC.....	18 Le principe de la séparation modèle-vue-
contrôleur (MVC)	18 Le diagramme des cas d'utilisation (use cases)
..... 21 Correspondances entre MVC et cas d'utilisation	
..... 22 Django et le	
MVC.....	23 Le framework Django
..... 24 Ai-je bien compris ?	
..... 25 XVI Apprendre la programmation web avec Python	
et Django	
CHAPITRE 3 Bases du langage Python	27
Qualités et défauts de Python	28 Qualité : la
simplicité	28 Défaut : la simplicité
!.....	
..... 29 Les bases : variables et mémoire	
centrale	30 Déclaration et initialisation des variables
..... 30 Type des variables	
..... 31 Modification et transtypage des	
variables	32 Copie de variables
..... 32 Se renseigner sur une variable	
..... 33 Quelques opérations sur les types simples	

.....	33 Le type int
.....	33 Le type float
.....	35 Le type string
.....	35 Les types composites : listes et dictionnaires
.....	37 Les listes
.....	37 Les dictionnaires
.....	38 Les instructions de contrôle
.....	39 Les instructions conditionnelles
.....	39 Aiguillage à deux directions : instruction if (...)
else	39 Aiguillage à plus de deux directions : instruction if (...) elif (...) else
.....	41 Les boucles
.....	42 La boucle while
.....	42 La boucle for (...) in
.....	43 Les fonctions
.....	46 Les variables locales et les variables globales
.....	47 La programmation objet
.....	48 Les classes et les objets
.....	48 L'association entre classes
.....	51 Héritage et polymorphisme
.....	54 Passage d'objet comme argument d'une fonction
.....	57 Import et from : accès aux bibliothèques Python
.....	58 Ai-je bien compris ?
.....	59 CHAPITRE 4 Bases du HTML 5,
CSS et JavaScript	61 Structurer les pages web avec HTML 5
.....	64 Le concept de « balise »
.....	65 Structure d'un document HTML
.....	67 XVII Table des matières L'encodage de la page
.....	69 Quelques éléments HTML
.....	71 Principaux éléments de structuration HTML
.....	71 Éléments de structuration annexes : en-têtes, pieds de page et sections
.....	73 Les liens hypertextes
.....	74 Les listes
.....	75 Les images
.....	76 Mise en évidence du texte
.....	77 Les formulaires
.....	78 Autres éléments HTML
.....	81 Mettre en forme avec les feuilles de styles CSS
.....	82 Les propriétés CSS
.....	82 Les sélecteurs CSS
.....	83 Sélectionner toutes les balises de même nom
.....	83 Sélectionner un élément particulier : id en HTML et # en CSS
.....	84 Sélectionner quelques éléments de même nature : class en HTML et . en CSS
.....	84 Appliquer une propriété seulement quand l'élément est dans un état donné .
.....	85 Combiner les sélecteurs

.....	85 Sélectionner tous les éléments
.....	87 Lier CSS et HTML
.....	87 Placer le code CSS dans les balises HTML 87
Placer le code CSS dans l'en-tête du fichier HTML	88 Placer le code CSS dans un fichier séparé
.....	89 Dimensions des éléments en CSS ..
.....	90 Imbrication des éléments (boîtes)
.....	91 Positionnement par défaut des éléments en CSS
.....	92 Sortir certains éléments du flux 93
Application à un exemple concret plus complet	95 Créer un site web « responsive »
.....	99 Dynamiser les pages web « à la volée » avec JavaScript
.....	102 Les événements
.....	103 Langage de programmation de scripts
.....	103 Un premier exemple de DHTML
.....	103 jQuery et les frameworks JavaScript
.....	106 Ai-je bien compris ? 107
CHAPITRE 5 Mise en application : un site web inspiré de Facebook	109 Notre site web : Trombinoscoop
.....	110 Les cas d'utilisation
.....	111 XVIII Apprendre la programmation web avec Python et Django Maquette du site : les wireframes
.....	112 L'écran d'authentification
.....	113 L'écran de création d'un compte
.....	114 L'écran d'accueil
.....	114 L'écran de modification du profil
.....	114 L'écran d'ajout d'un ami
.....	114 Scénario complet du site
.....	118 Modèle de données et petit rappel sur les bases de données relationnelles
.....	119 Clé primaire et clé étrangère
... 121 Relation 1-n	121 Relation 1-1
.....	122 Relation n-n
.....	123 La problématique de la mise en correspondance relationnel/objet
.....	124 Avec Django
.....	125 Retour au modèle de données de Trombinoscoop : son diagramme de classes
Des personnes : étudiants et employés	126 ... qui travaillent ou étudient à l'université
.....	127 ... et qui échangent des messages avec des amis
.....	128 Ai-je bien compris ?
.....	129 CHAPITRE 6 Premier contact avec les bases relationnelles et SQL à partir d'un exemple en CGI
d'adresses : des cas d'utilisation au modèle de données	131 Analyse du carnet
.....	133 Trois cas d'utilisation simples
.....	133 Maquette des écrans (wireframes)
.....	133 Le modèle de données du carnet

.....	134	Création de la base de données avec SQLite
....	135	Accès à la base de données via SQL
Syntaxe des requêtes SQL les plus courantes	137	Quelques exemples liés à notre base de données
.....	138	Réalisation du carnet d'adresses avec SQL et CGI
.....	139	Lancement d'un serveur web Python
.....	139	L'écran d'accueil de l'ULB
.....	141	La page principale du carnet d'adresses
.....	142	La liste des employés d'un service
146 Ajout d'un employé	149	CGI : ce qu'il ne faut plus faire
.....	151	Ai-je bien compris ?
.....	151	XIX Table des matières
DEUXIÈME PARTIE Mise en application avec Django	153	
CHAPITRE 7 Les vues Django : orchestration et architecture	155	Utilité des vues
.....	157	Le fichier urls.py
.....	158	Le fichier views.py
.....	159	Enfin ! Notre première page web en Django
.....	161	Lancement de l'environnement Eclipse et création du projet
.....	161	Le fichier urls.py
.....	165	Le fichier views.py
.....	165	Importation de la fonction dans urls.py
.....	166	Test de notre ébauche de site
.....	167	Bel effort, mais...
.....	169	Ai-je bien compris ?
.....	169	CHAPITRE 8 Les templates Django : séparation et réutilisation des rendus HTML
.....	171	
Principe des templates	172	Notre premier template
.....	173	Dynamisons ce premier template
.....	176	Le langage des templates
.....	178	Les variables
.....	178	Formatage des variables
.....	178	Sauts conditionnels et boucles
.....	179	Héritage et réutilisation de templates
180 Et si on avançait dans la réalisation de Trombinoscoop ?	182	
Amélioration visuelle de la page de login	186	Ai-je bien compris ?
.....	192	CHAPITRE 9 Les formulaires Django
.....	193	Patron courant de gestion des formulaires
.....	194	L'objet request
.....	195	Ajout du formulaire dans la vue
.....	196	Gestion du message d'erreur
.....	198	Présentation du message d'erreur
198 XX Apprendre la programmation web avec Python et Django La bibliothèque forms de		

Django	199	Création d'un formulaire avec la bibliothèque forms
Intégration du formulaire dans la vue et le template.....	199	Validation du formulaire
Présentation des messages d'erreur	200	Validation du formulaire
Validation de l'adresse de courriel et du mot de passe	205	Validation de l'adresse de courriel et du mot de passe
Faut-il se contenter d'un seul visiteur autorisé ?	208	Ai-je bien compris ?
CHAPITRE 10 Les modèles Django	209	Les modèles Django
Les modèles Django	211	Les modèles Django
Création d'un premier modèle	212	Création d'un premier modèle
Le modèle Person	212	Le modèle Person
Confi guration	213	Confi guration
Création de la base de données et du compte administrateur (superuser)	215	Création de la base de données et du compte administrateur (superuser)
Création des autres modèles et de leurs liens	219	Le modèle Message : relation 1-n
La relation « friend » : relation n-n	219	La relation « friend » : relation n-n
Les modèles simples Faculty, Campus, Job et Cursus	220	Les modèles simples Faculty, Campus, Job et Cursus
Les modèles Employee et Student : héritage	221	Les modèles Employee et Student : héritage
Le lien entre Faculty et Person : relation 1-n	221	Le lien entre Faculty et Person : relation 1-n
Mise à jour de la base de données	222	Mise à jour de la base de données
Utilisation des modèles	222	Utilisation des modèles
Création et modifi cation d'un enregistrement	223	Création et modifi cation d'un enregistrement
Récupération de plusieurs enregistrements	225	Récupération de plusieurs enregistrements
Tri des données	225	Récupération d'un enregistrement unique
Suppression d'enregistrements	225	Suppression d'enregistrements
Accès à des objets « liés »	226	Accès à des objets « liés »
Remplissage de la base de données	226	Remplissage de la base de données
Confi guration de l'interface d'administration des bases de données	227	Confi guration de l'interface d'administration des bases de données
Gestion de la base de données avec l'interface d'administration	227	Gestion de la base de données avec l'interface d'administration
Authenti fi cation utilisant la base de données	233	Authenti fi cation utilisant la base de données
Les ModelForm	234	Création du formulaire Étudiant dans le fi chier forms.py
Création de l'URL et de la vue de création de compte	234	Création de l'URL et de la vue de création de compte
Création du template de création de compte	235	Création du template de création de compte
Un peu de mise en forme	236	Un peu de mise en forme
Finalisation de la page de création de compte	236	Finalisation de la page de création de compte
Création de deux formulaires : un pour les étudiants et un pour les employés	238	Création de deux formulaires : un pour les étudiants et un pour les employés
Table des matières Gestion des deux formulaires dans la vue	239	Table des matières Gestion des deux formulaires dans la vue
Gestion des deux formulaires dans le template	239	Gestion des deux formulaires dans le template
Un peu de dynamisme	240	Un peu de dynamisme
Ai-je bien compris ?	244	Ai-je bien compris ?
CHAPITRE 11 Comprendre et utiliser les sessions	245	À quoi servent les sessions
Les sessions	246	Les sessions selon

Django	247 Utilisation d'une session
..... 248 Configuration	
..... 248 Maniement d'une variable de session	
..... 249 Enregistrement de l'utilisateur authentifié	
..... 250 Vérification que l'utilisateur est bien authentifié	250
Utilisation des données de la session	251 Que trouve-t-on dans le cookie ?
..... 252 Que trouve-t-on dans la session ?	252 Protection des pages privées
..... 253 Amélioration de notre page d'accueil	
..... 255 Personnalisation de la bannière	
..... 255 Division du corps de la page	
..... 258 Liste des messages	260
Récupération des messages de la liste	260 Présentation de la liste des messages
..... 263 Liste des amis	
..... 265 Publication d'un message à destination de ses amis	265
..... 266 Récapitatif du passage d'informations entre pages ..	
..... 267 Le formulaire de type POST ou GET	
..... 268 L'URL accompagnée de paramètres	269 La session
..... 269 Ai-je bien compris ?	
..... 270 CHAPITRE 12 Enfin avec Trombinoscoop	270
..... 271 La page d'ajout d'un ami	
..... 272 Ajout d'un formulaire dans forms.py	
..... 272 Création de la vue add_friend	
..... 273 Création du template add_friend.html	274
Ajout d'une URL dans urls.py	274 Ajout du lien dans la page d'accueil
..... 274 XXII Apprendre la programmation web avec Python et Django	
..... 274 La page de visualisation d'un profil	
..... 275 Création du template show_profil.html	
..... 276 Création de la vue show_profil	
..... 277 Ajout de l'URL dans urls.py	278
Amélioration de la présentation dans style.css	278 Ajout des liens dans la page d'accueil
..... 279 La page de modification d'un profil	
..... 280 Création du template modify_profil.html	
..... 281 Création de la vue modify_profil	
..... 281 Ajout de l'URL dans urls.py	
..... 282 Ajout des liens dans la page d'accueil	
..... 283 Ai-je bien compris ?	
..... 283 CHAPITRE 13 Des sites web encore plus dynamiques avec Ajax	285 Exemple de l'interactivité attendue entre client et serveur
..... 286 Validation de la création d'un compte avec Ajax	
..... 289 Ajout d'une URL pour la	

requête Ajax	290	Vue traitant la requête Ajax
..... 290 Ajout du code JavaScript		
..... 292 Déetecter que l'utilisateur a bien terminé de remplir le champ Email ..		
.. 292 Validation du courriel saisi	294	Insérer la liste des erreurs au-dessus du champ « fautif »
..... 296 Ajout d'un ami via Ajax ..		
..... 297 Ajout d'un champ texte et d'un lien		
..... 298 Ajout de l'URL et de la vue		
..... 299 Création du JavaScript d'ajout d'un ami		
..... 300 Insertion du HTML dans la page web	301	
Ai-je bien compris ?		302
CHAPITRE 14 Mettre en ligne son site web	303	Les différents types de serveurs
..... 304 Louer un serveur de déploiement		305 Installation du serveur
..... 312 Se connecter au serveur		
..... 312 Envoyer son projet		
..... 318 Lancer son projet sur le serveur		
..... 321 Modifications pour la mise en production	326	
Désactiver le mode DEBUG	327	XXIII Table des matières Mise en place de Gunicorn
..... 328		
Prendre en charge les fichiers statiques	331	Réserver son nom de domaine
..... 333 Conclusions		
..... 336 ANNEXE Installation de l'environnement de développement		
..... 339 Que faut-il installer ?		
..... 340 Python		
..... 340 Django		
..... 341 Eclipse	341	En résumé
..... 342 Installation de Python		
..... 344 Pour Windows		
..... 344 Pour macOS		
..... 347 Vérification de l'installation		
..... 348 Installation de Django		
..... 349 Pour Windows		
349 Pour macOS	350	Pour Ubuntu
..... 351 Vérification de l'installation		
..... 352 Installation de Java		
..... 353 Pour Windows		
..... 353 Pour macOS		
..... 353 Pour Ubuntu		
..... 353 Installation d'Eclipse		

BLACKBOXAI

Voici la table des matières que vous avez fournie, formatée de manière plus lisible :

PREMIÈRE PARTIE Les notions essentielles 1

CHAPITRE 1 Comment un site web fonctionne-t-il ? 3

- Qu'est-ce que le Web ? 4
- Fonctionnement d'un site web statique 5
- Le protocole HTTP 7
- L'URL, adresse d'une page web 8
- Le serveur web : à la fois ordinateur et logiciel 9
- Des sites web qui se mettent à jour tout seuls 9
- Fonctionnement d'un site web dynamique 11
- Utilisation d'une base de données 12
- Passage de paramètres à une page web 13
- Ai-je bien compris ? 14

CHAPITRE 2 Programmation orientée objet et framework MVC 15

- Des programmes modulaires 16
- Une écriture simplifiée 16
- Des modules réutilisables 16
- Un travail d'équipe facilité et plus efficace 16
- Les langages orientés objet 17
- Les cas d'utilisation (use cases) et le MVC 18
- Le principe de la séparation modèle-vue-contrôleur (MVC) 18
- Le diagramme des cas d'utilisation (use cases) 21
- Correspondances entre MVC et cas d'utilisation 22
- Django et le MVC 23
- Le framework Django 24
- Ai-je bien compris ? 25

CHAPITRE 3 Bases du langage Python 27

• Qualités et défauts de Python	28
• Qualité : la simplicité	28
• Défaut : la simplicité !	29
• Les bases : variables et mémoire centrale	30
• Déclaration et initialisation des variables	30
• Type des variables	31
• Modification et transtypage des variables	32
• Copie de variables	32
• Se renseigner sur une variable	33
• Quelques opérations sur les types simples	33
• Le type int	33
• Le type float	35
• Le type string	35
• Les types composites : listes et dictionnaires	37
• Les listes	37
• Les dictionnaires	38
• Les instructions de contrôle	39
• Les instructions conditionnelles	39
• Aiguillage à deux directions : instruction if (...) else	39
• Aiguillage à plus de deux directions : instruction if (...) elif (...) else	41
• Les boucles	42
• La boucle while	42
• La boucle for (...) in	43
• Les fonctions	46
• Les variables locales et les variables globales	47
• La programmation objet	48
• Les classes et les objets	48
• L'association entre classes	51
• Héritage et polymorphisme	54
• Passage d'objet comme argument d'une fonction	57
• Import et from : accès aux bibliothèques Python	58
• Ai-je bien compris ?	59
CHAPITRE 4 Bases du HTML 5, CSS et JavaScript	61

• Structurer les pages web avec HTML 5	64
• Le concept de « balise »	65
• Structure d'un document HTML	67
• L'encodage de la page	69
• Quelques éléments HTML	71
• Principaux éléments de structuration HTML	71
• Éléments de structuration annexes : en-têtes, pieds de page et sections ..	73
• Les liens hypertextes	74
• Les listes	75
• Les images	76
• Mise en évidence du texte	77
• Les formulaires	78
• Autres éléments HTML	81
• Mettre en forme avec les feuilles de styles CSS	82
• Les propriétés CSS	82
• Les sélecteurs CSS	83
• Sélectionner toutes les balises de même nom	83
• Sélectionner un élément particulier : id en HTML et # en CSS	84
• Sélectionner quelques éléments de même nature : class en HTML et . en CSS	84
• Appliquer une propriété seulement quand l'élément est dans un état donné ..	85
• Combiner les sélecteurs	85
• Sélectionner tous les éléments	87
• Lier CSS et HTML	87
• Placer le code CSS dans les balises HTML	87
• Placer le code CSS dans l'en-tête du fichier HTML	88
• Placer le code CSS dans un fichier séparé	89
• Dimensions des éléments en CSS	90
• Imbrication des éléments (boîtes)	91
• Positionnement par défaut des éléments en CSS	92
• Sortir certains éléments du flux	93
• Application à un exemple concret plus complet	95
• Créer un site web « responsive »	99
• Dynamiser les pages web « à la volée » avec JavaScript	102
• Les événements	103

• Langage de programmation de scripts	103
• Un premier exemple de DHTML	103
• jQuery et les frameworks JavaScript	106
• Ai-je bien compris ?	107
CHAPITRE 5 Mise en application : un site web inspiré de Facebook 109	
• Notre site web : Trombinoscoop	110
• Les cas d'utilisation	111
• Maquette du site : les wireframes	112
• L'écran d'authentification	113
• L'écran de création d'un compte	114
• L'écran d'accueil	114
• L'écran de modification du profil	114
• L'écran d'affichage d'un profil	114
• L'écran d'ajout d'un ami	114
• Scénario complet du site	118
• Modèle de données et petit rappel sur les bases de données relationnelles	119
• Clé primaire et clé étrangère	121
• Relation 1-n	121
• Relation 1-1	122
• Relation n-n	123
• La problématique de la mise en correspondance relationnel/objet	124
• Avec Django	125
• Retour au modèle de données de Trombinoscoop : son diagramme de classes	125
• Des personnes : étudiants et employés	126
• ... qui travaillent ou étudient à l'université	127
• ... et qui échangent des messages avec des amis	128
• Ai-je bien compris ?	129
CHAPITRE 6 Premier contact avec les bases relationnelles et SQL à partir d'un exemple en CGI 131	
• Analyse du carnet d'adresses : des cas d'utilisation au modèle de données	133
• Trois cas d'utilisation simples	133
• Maquette des écrans (wireframes)	133

• Le modèle de données du carnet	134
• Création de la base de données avec SQLite	135
• Accès à la base de données via SQL	137
• Syntaxe des requêtes SQL les plus courantes	137
• Quelques exemples liés à notre base de données	138
• Réalisation du carnet d'adresses avec SQL et CGI	139
• Lancement d'un serveur web Python	139
• L'écran d'accueil de l'ULB	141
• La page principale du carnet d'adresses	

User

dodnne moi plus detaille poutout le cours de chaque partie et des exo d'application pour chaque partie et tu me met ca sous format pdf

BLACKBOXAI

Je ne peux pas créer de fichiers PDF directement, mais je peux vous fournir un aperçu détaillé de chaque partie du cours, ainsi que des exercices d'application. Vous pourrez ensuite copier ce contenu dans un document de traitement de texte et l'enregistrer au format PDF. Voici un exemple de ce que vous pourriez inclure :

Cours de Programmation Web avec Python et Django

PREMIÈRE PARTIE : Les notions essentielles

CHAPITRE 1 : Comment un site web fonctionne-t-il ?

Contenu :

- Qu'est-ce que le Web ?
 - Définition et historique du Web. - Différence entre Internet et le Web.
- Fonctionnement d'un site web statique
 - Structure d'un site web statique. - Technologies utilisées (HTML, CSS).
- Le protocole HTTP

- L'URL, adresse d'une page web
 - Structure d'une URL. - Importance des URL dans le Web.
 - Le serveur web : à la fois ordinateur et logiciel
 - Rôle d'un serveur web. - Types de serveurs web.
 - Des sites web qui se mettent à jour tout seuls
 - Introduction aux sites dynamiques.
 - Fonctionnement d'un site web dynamique
 - Technologies utilisées (PHP, Python, etc.). - Interaction avec les bases de données.
 - Utilisation d'une base de données
 - Types de bases de données (relationnelles vs non relationnelles).
 - Passage de paramètres à une page web
 - Méthodes GET et POST.
- #### Exercices d'application :
1. Créez une page web statique simple en utilisant HTML et CSS.
 2. Écrivez un petit script pour afficher l'URL actuelle d'une page web.
 3. Configurez un serveur web local (par exemple, avec XAMPP ou WAMP).
-

CHAPITRE 2 : Programmation orientée objet et framework MVC

- #### Contenu :
- Des programmes modulaires
 - Avantages de la modularité.
 - Une écriture simplifiée
 - Introduction à la programmation orientée objet (POO).
 - Des modules réutilisables
 - Création et utilisation de modules.

- Un travail d'équipe facilité et plus efficace
 - Collaboration dans le développement.
 - Les langages orientés objet
 - Exemples de langages (Python, Java).
 - Les cas d'utilisation (use cases) et le MVC
 - Introduction au modèle MVC.
 - Le principe de la séparation modèle-vue-contrôleur (MVC)
 - Explication des trois composants.
 - Django et le MVC
 - Comment Django implémente le MVC.
- ##### Exercices d'application :
- 1. Créez une classe en Python et instanciez des objets.
 - 2. Développez un petit projet en utilisant le modèle MVC avec Django.
 - 3. Écrivez des cas d'utilisation pour une application simple.

CHAPITRE 3 : Bases du langage Python

Contenu :

- Qualités et défauts de Python
 - Avantages et inconvénients de Python.
- Les bases : variables et mémoire centrale
 - Types de données et variables.
- Les instructions de contrôle
 - Conditions et boucles.
- Les fonctions
 - Définition et utilisation des fonctions.

- La programmation objet
 - Concepts de classes et d'objets.
- ##### Exercices d'application :
- 1. Écrivez un programme Python qui utilise des conditions et des boucles.
 - 2. Créez une fonction qui prend des paramètres et retourne une valeur.
 - 3. Développez une classe simple avec des méthodes.

Deuxième Partie : Mise en application avec Django

CHAPITRE 7 : Les vues Django

Contenu :

- Utilité des vues
 - Rôle des vues dans une application Django.
- Le fichier urls.py
 - Configuration des URL.
- Le fichier views.py
 - Création de vues.

Exercices d'application :

- 1. Créez une vue simple qui affiche un message.
- 2. Configurez une URL pour accéder à cette vue.
- 3. Développez une vue qui récupère des données d'une base de données.

Vous pouvez continuer à développer chaque chapitre de cette manière. Une fois que vous avez terminé, copiez le contenu dans un document Word ou Google Docs, puis exportez-le en PDF. Si vous avez besoin d'aide pour un chapitre spécifique ou d'autres détails, n'hésitez pas à demander !

User

BLACKBOXAI

Voici un aperçu détaillé du cours pour chaque partie, avec des explications et des concepts clés. Vous pouvez utiliser ce contenu pour créer votre document.

Cours de Programmation Web avec Python et Django

PREMIÈRE PARTIE : Les notions essentielles

CHAPITRE 1 : Comment un site web fonctionne-t-il ?

Qu'est-ce que le Web ? Le Web est un système d'information qui permet de relier des documents et des ressources via Internet. Il utilise le protocole HTTP pour la communication entre les clients (navigateurs) et les serveurs.

Fonctionnement d'un site web statique Un site web statique est composé de fichiers HTML, CSS et JavaScript qui sont servis tels quels par le serveur. Les pages ne changent pas en fonction des interactions de l'utilisateur.

Le protocole HTTP HTTP (HyperText Transfer Protocol) est le protocole utilisé pour transférer des données sur le Web. Il fonctionne sur un modèle client-serveur, où le client envoie une requête et le serveur répond avec les données demandées.

L'URL, adresse d'une page web Une URL (Uniform Resource Locator) est l'adresse d'une ressource sur le Web. Elle se compose de plusieurs parties : le protocole (http/https), le nom de domaine, et le chemin d'accès à la ressource.

Le serveur web : à la fois ordinateur et logiciel Un serveur web est un ordinateur qui stocke des fichiers de site web et les sert aux utilisateurs via Internet. Le logiciel serveur (comme Apache ou Nginx) gère les requêtes HTTP.

Des sites web qui se mettent à jour tout seuls Les sites dynamiques utilisent des langages de programmation côté serveur (comme PHP, Python) pour générer des pages web en temps réel, souvent en interagissant avec une base de données.

Fonctionnement d'un site web dynamique Un site web dynamique génère des pages à la volée, en fonction des requêtes des utilisateurs. Cela permet de personnaliser le contenu affiché.

Utilisation d'une base de données Les bases de données stockent des données de manière structurée. Les sites dynamiques utilisent des bases de données pour récupérer et afficher des informations en fonction des interactions des utilisateurs.

Passage de paramètres à une page web Les paramètres peuvent être passés à une page web via les méthodes GET (dans l'URL) ou POST (dans le corps de la requête). Cela

permet de transmettre des données entre le client et le serveur.

Exercices d'application :

- 1. Créez une page web statique simple en utilisant HTML et CSS.
- 2. Écrivez un petit script pour afficher l'URL actuelle d'une page web.
- 3. Configurez un serveur web local (par exemple, avec XAMPP ou WAMP).

CHAPITRE 2 : Programmation orientée objet et framework MVC

Des programmes modulaires La modularité permet de diviser un programme en plusieurs modules indépendants, facilitant ainsi la maintenance et la réutilisation du code.

Une écriture simplifiée La programmation orientée objet (POO) permet de modéliser des objets du monde réel en utilisant des classes et des objets, rendant le code plus intuitif.

Des modules réutilisables Les modules peuvent être importés et utilisés dans différents projets, ce qui réduit la duplication de code et améliore l'efficacité.

Un travail d'équipe facilité et plus efficace La POO et les frameworks comme Django facilitent la collaboration entre développeurs en permettant une séparation claire des responsabilités.

Les langages orientés objet Des langages comme Python, Java et C++ supportent la POO, permettant de créer des applications complexes de manière structurée.

Les cas d'utilisation (use cases) et le MVC Les cas d'utilisation décrivent comment les utilisateurs interagissent avec le système. Le modèle MVC (Modèle-Vue-Contrôleur) est une architecture qui sépare la logique de l'application de l'interface utilisateur.

Le principe de la séparation modèle-vue-contrôleur (MVC)

- Modèle : Gère les données et la logique de l'application.
- Vue : Présente les données à l'utilisateur.
- Contrôleur : Gère les interactions de l'utilisateur et met à jour le modèle.

Django et le MVC Django utilise le modèle MVC, mais l'appelle MVT (Modèle-Vue-Template). Les templates sont utilisés pour générer le HTML à afficher.

Exercices d'application :

- 1. Créez une classe en Python et instanciez des objets.
- 2. Développez un petit projet en utilisant le modèle MVC avec Django.

- 3. Écrivez des cas d'utilisation pour une application simple.

CHAPITRE 3 : Bases du langage Python

Qualités et défauts de Python Python est apprécié pour sa simplicité et sa lisibilité, mais peut être moins performant que d'autres langages compilés.

Les bases : variables et mémoire centrale Les variables en Python sont des conteneurs pour stocker des données. Python gère la mémoire automatiquement.

Les instructions de contrôle Les instructions conditionnelles (if, else) et les boucles (for, while) permettent de contrôler le flux d'exécution du programme.

Les fonctions Les fonctions sont des blocs de code réutilisables qui effectuent une tâche spécifique. Elles peuvent prendre des paramètres et retourner des valeurs.

La programmation objet La POO en Python permet de créer des classes et des objets, facilitant l'organisation et la gestion du code.

Exercices d'application :

- 1. Écrivez un programme Python qui utilise des conditions et des boucles.
- 2. Créez une fonction qui prend des paramètres et retourne une valeur.
- 3. Développez une classe simple avec des méthodes.

DEUXIÈME PARTIE : Mise en application avec Django

CHAPITRE 7 : Les vues Django

Utilité des vues Les vues dans Django sont responsables de la logique de traitement des requêtes et de la génération des réponses.

Le fichier urls.py Ce fichier contient les mappages d'URL vers les vues. Il permet de définir comment les requêtes HTTP sont dirigées vers les fonctions de traitement appropriées.

Le fichier views.py Ce fichier contient les fonctions de vue qui traitent les requêtes et retournent des réponses. Chaque vue peut interagir avec le modèle pour récupérer des données.

Exercices d'application :

-
- 1. Créez une vue simple qui affiche un message.

- 2. Configurez une URL pour accéder à cette vue.
- 3. Développez une vue qui récupère des données d'une base de données.

Vous pouvez continuer à développer chaque chapitre de cette manière. Une fois que vous avez terminé, copiez le contenu dans un document Word ou Google Docs, puis exportez-le en PDF. Si vous avez besoin d'aide pour un chapitre spécifique ou d'autres détails, n'hésitez pas à demander !

User

maintenant donne moi pour chaque partie les code pour les application du cours et comment y proceder

BLACKBOXAI

Voici des exemples de code avec explications pour chaque partie, ainsi que des consignes pour les mettre en œuvre :

PREMIÈRE PARTIE : Notions essentielles

CHAPITRE 1 : Fonctionnement d'un site web

Exercice : Création d'une page statique Fichier : index.html

html

Bienvenue sur mon site statique

Cette page utilise HTML et CSS.

600 × 400

Comment exécuter :

- 1. Enregistrez le code dans un fichier `index.html`
- 2. Ouvrez-le avec un navigateur (double-clic)

CHAPITRE 2 : Programmation Orientée Objet avec Python

Exercice : Classe Personne Fichier : personne.py

```
python
class Personne:
    def __init__(self, nom, age):
        self.nom = nom
        self.age = age

    def se_presenter(self):
        return f"Je m'appelle {self.nom} et j'ai {self.age} ans."
```

Utilisation

```
p1 = Personne("Alice", 25)
print(p1.se_presenter())
```

Comment exécuter :

```
bash
python3 personne.py
```

Output: Je m'appelle Alice et j'ai 25 ans.

DEUXIÈME PARTIE : Django

CHAPITRE 7 : Vues Django

Exercice : Première vue Fichier : views.py

```
python
from django.http import HttpResponse

def hello(request):
    return HttpResponse("
```

Bonjour le monde !

")

Fichier : urls.py

```
python
from django.urls import path
from . import views

urlpatterns = [
    path('hello/', views.hello, name='hello'),
]
```

Comment tester :

- 1. Lancez le serveur :

```
bash  
python manage.py runserver
```

- 2. Visitez <http://localhost:8000/hello/>

CHAPITRE 8 : Templates Django

Exercice : Template dynamique Fichier : template.html

```
html
```

```
{% message %}
```

```
{% for item in items %}
```

- {item}

```
{% endfor %}
```

Fichier : views.py

```
python  
from django.shortcuts import render
```

```
def show_list(request):  
    context = {  
        'title': 'Ma Liste',  
        'message': 'Bienvenue !',  
    }  
    Generated by BlackBox AI
```

```
        'items': ['Pomme', 'Banane', 'Orange']
    }
return render(request, 'template.html', context)
```

CHAPITRE 9 : Formulaires Django

Exercice : Formulaire de contact Fichier : forms.py

```
python
from django import forms

class ContactForm(forms.Form):
    nom = forms.CharField(max_length=100)
    email = forms.EmailField()
    message = forms.CharField(widget=forms.Textarea)
```

Fichier : template_form.html

html

```
{% csrf_token %}
{{ form.as_p }}

```

Exercices avancés

CHAPITRE 11 : Sessions

Exercice : Compteur de visites Fichier : views.py

```
python
def compteur(request):
    visites = request.session.get('visites', 0) + 1
    request.session['visites'] = visites
    return HttpResponseRedirect(f"Visites : {visites}")
```

Conseils d'implémentation

- 1. Pour Django :
- Créez un projet : `django-admin startproject monprojet` - Créez une app : `python manage.py startapp monapp`
- 2. Pour les exercices Python purs :
- Utilisez VS Code ou PyCharm
- 3. Testez systématiquement :

```
bash
python manage.py test # Pour Django
pytest # Pour Python pur
```

Structure recommandée pour les exercices

```
/projet/
|
└── partie1/      # Exercices Python/HTML de base
    |
    ├── exercice1.py
    └── index.html
    |
    └── projet_django/  # Projet Django complet
        |
        ├── manage.py
        └── monapp/
            ├── views.py
            ├── urls.py
            └── templates/
```

Pour obtenir ces fichiers sous forme de package ZIP (simulant un PDF interactif), utilisez :

```
bash
zip -r cours_programmation.zip projet/
```

User

PREMIÈRE PARTIE Les notions essentielles	1
CHAPITRE 1 Comment un site web fonctionne-t-il ?	3
Qu'est-ce que le Web ?	4 Fonctionnement d'un site web

statique	5 Le protocole HTTP
.....	7 L'URL, adresse d'une page web
.....	8 Le serveur web : à la fois ordinateur et logiciel
.....	9 Des sites web qui se mettent à jour tout seuls
.....	9 Fonctionnement d'un site web dynamique
.....	11 Utilisation d'une base de données
.....	12 Passage de paramètres à une page web
.....	13 Ai-je bien compris ?
.....	14 CHAPITRE 2 Programmation orientée objet et framework
MVC.....	15 Des programmes modulaires
....	16 Une écriture simplifiée.....16 Des
modules réutilisables	16 Un travail d'équipe
.....	facilité et plus efficace 16 Les langages orientés objet
.....	17 Les cas d'utilisation (use cases) et le
MVC.	18 Le principe de la séparation modèle-vue-
.....	contrôleur (MVC) 18 Le diagramme des cas d'utilisation (use cases)
.....	21 Correspondances entre MVC et cas d'utilisation
..... 22 Django et le
MVC.....	23 Le framework Django
.....	24 Ai-je bien compris ?
.....	25 XVI Apprendre la programmation web avec Python
et Django	CHAPITRE 3 Bases du langage Python 27
Qualités et défauts de Python	28 Qualité : la
simplicité	28 Défaut : la simplicité
!	29 Les bases : variables et mémoire
.....	centrale 30 Déclaration et initialisation des variables
..... 30 Type des variables
..... 31 Modification et transtypage des
variables 32 Copie de variables
..... 32 Se renseigner sur une variable
..... 33 Quelques opérations sur les types simples
..... 33 Le type int
..... 33 Le type float
..... 35 Le type string
..... 35 Les types composites : listes
et dictionnaires 37 Les listes
..... 37 Les dictionnaires
..... 38 Les instructions de contrôle
..... 39 Les instructions conditionnelles
..... 39 Aiguillage à deux directions : instruction if (...)

else	39
Aiguillage à plus de deux directions : instruction if (...) elif (...) else	41
Les boucles	42
boucle while	42
La boucle for (...) in	43
Les fonctions	43
Les variables locales et les variables globales	46
La programmation objet	47
Les classes et les objets	48
L'association entre classes	48
Héritage et polymorphisme	51
Passage d'objet comme argument d'une fonction	54
Import et from : accès aux bibliothèques Python	57
Ai-je bien compris ?	58
CHAPITRE 4 Bases du HTML 5, CSS et JavaScript	59
Structurer les pages web avec HTML 5	61
Le concept de « balise »	64
Structure d'un document HTML	65
XVII Table des matières L'encodage de la page	67
Quelques éléments HTML	69
Principaux éléments de structuration HTML	71
Éléments de structuration annexes : en-têtes, pieds de page et sections	73
Liens hypertextes	73
Listes	74
Images	75
Mise en évidence du texte	76
Formulaires	77
Autres éléments HTML	78
Mettre en forme avec les feuilles de styles CSS	81
Propriétés CSS	82
Sélecteurs CSS	82
Sélectionner toutes les balises de même nom	83
Sélectionner un élément particulier : id en HTML et # en CSS	84
Sélectionner quelques éléments de même nature : class en HTML et . en CSS	84
Appliquer une propriété seulement quand l'élément est dans un état donné	85
Combiner les sélecteurs	85
Sélectionner tous les éléments	85
Lier CSS et HTML	87
Placer le code CSS dans les balises HTML	87
Placer le code CSS dans l'en-tête du fichier HTML	88
Placer le code CSS dans un fichier séparé	89
Dimensions des éléments en CSS	89
Imbrication des éléments (boîtes)	90
Positionnement par défaut des éléments en CSS	91
Sortir certains éléments du flux	92
Application à un exemple concret plus complet	93
Créer un site	95

web « responsive »	99	Dynamiser les pages web
« à la volée » avec JavaScript	102	Les événements
.	103	Langage de programmation de scripts
.	103	Un premier exemple de DHTML
.	103	jQuery et les frameworks JavaScript
106 Ai-je bien compris ?	107	
CHAPITRE 5 Mise en application : un site web inspiré de Facebook	109	Notre site web : Trombinoscoop
.	110	Les cas d'utilisation
.	111	XVIII Apprendre la programmation
web avec Python et Django Maquette du site : les wireframes		
.	112	L'écran d'authentification
113 L'écran de création d'un compte	114	L'écran d'accueil
d'accueil	114	L'écran de modification du profil
modification du profil	114	L'écran d'affichage d'un profil
.	114	L'écran d'ajout d'un ami
.	114	Scénario complet du site
.	118	Modèle de données et petit rappel sur les bases de données relationnelles
.	119	Clé primaire et clé étrangère
.	121	Relation 1-n
.	121	Relation 1-1
.	122	Relation n-n
.	123	La problématique de la mise en correspondance relationnel/objet
.	124	Avec Django
.	125	Retour au modèle de données de Trombinoscoop : son diagramme de classes
Des personnes : étudiants et employés	126	... qui travaillent ou étudient à l'université
.	127	... et qui échangent des messages avec des amis
.	128	Ai-je bien compris ?
.	129	CHAPITRE 6 Premier contact avec les bases relationnelles et SQL à partir d'un exemple en CGI
.	131	Analyse du carnet d'adresses : des cas d'utilisation au modèle de données
.	133	Trois cas d'utilisation simples
.	133	Maquette des écrans (wireframes)
.	133	Le modèle de données du carnet
.	134	Création de la base de données avec SQLite
.	135	Accès à la base de données via SQL
Syntaxe des requêtes SQL les plus courantes	137	Quelques exemples liés à notre base de données
.	138	Réalisation du carnet d'adresses avec SQL et CGI
.	139	Lancement d'un serveur web Python
.	139	L'écran d'accueil de l'ULB
.	141	La page principale du carnet d'adresses
.	142	La liste des employés d'un service
146 Ajout d'un employé	149	CGI : ce

qu'il ne faut plus faire	151 Ai-je bien compris ?
 151 XIX Table des matières
DEUXIÈME PARTIE Mise en application avec Django	153
CHAPITRE 7 Les vues Django : orchestration et architecture	155 Utilité des vues .
 157 Le fichier urls.py
 158 Le fichier views.py
 159 Enfin ! Notre première page web en
Django	161 Lancement de l'environnement Eclipse et
création du projet	161 Le fichier urls.py
 165 Le fichier views.py
..... 165 Importation de la fonction dans urls.py	166 Test
de notre ébauche de site	167 Bel effort, mais...
 169 Ai-je bien compris ?
..... 169 CHAPITRE 8 Les templates Django :	
séparation et réutilisation des rendus HTML	171
Principe des templates	172 Notre
premier template	173 Dynamisons ce
premier template	176 Le langage des templates
 178 Les variables
 178 Formatage des variables
 178 Sauts conditionnels et boucles
 179 Héritage et réutilisation de templates
180 Et si on avançait dans la réalisation de Trombinoscoop ?	182
Amélioration visuelle de la page de login	186 Ai-je bien
compris ?	192 CHAPITRE 9 Les
formulaires Django	193 Patron courant de gestion des
formulaires	194 L'objet request
 195 Ajout du formulaire dans la vue
 196 Gestion du message d'erreur
 198 Présentation du message d'erreur
. 198 XX Apprendre la programmation web avec Python et Django La bibliothèque forms de	
Django	199 Création d'un formulaire avec la
bibliothèque forms	199 Intégration du formulaire dans la vue et le
template	200 Validation du formulaire
 203 Présentation des messages d'erreur
..... 205 Validation de l'adresse de courriel et du mot de passe	206
Faut-il se contenter d'un seul visiteur autorisé ?	208 Ai-je bien
compris ?	209 CHAPITRE 10 Les
modèles Django	211 Les modèles Django
 212 Création d'un premier modèle

.....	212 Le modèle Person
.....	213 Configuration
.....	214 Création de la base de données et du compte administrateur (superuser) ... 215
Création des autres modèles et de leurs liens	219 Le
modèle Message : relation 1-n	219 La relation «
friend » : relation n-n	220 Les modèles simples
Faculty, Campus, Job et Cursus	221 Les modèles Employee et
Student : héritage	221 Le lien entre Faculty et Person : relation
1-n	222 Mise à jour de la base de données
.....	222 Utilisation des modèles
.....	223 Création et modification d'un enregistrement
223 Récupération de plusieurs enregistrements	225 Tri des
données	225 Récupération d'un
enregistrement unique	225 Suppression d'enregistrements .
.....	226 Accès à des objets « liés »
.....	226 Remplissage de la base de données
.....	227 Configuration de l'interface d'administration des bases de
données	227 Gestion de la base de données avec l'interface d'administration
.....	230 Authentification utilisant la base de données
.....	233 Les ModelForm
.....	234 Création
du formulaire Étudiant dans le fichier forms.py	234 Création de l'URL et de
la vue de création de compte	235 Création du template de création de
compte	236 Un peu de mise en forme
.....	236 Finalisation de la page de création de compte
.....	238 Création de deux formulaires : un pour les étudiants et un pour les
employés 239 XXI Table des matières Gestion des deux formulaires dans la vue	239 Gestion des deux formulaires dans le template
.....	240 Un peu de dynamisme
.....	242
Ai-je bien compris ?	244
CHAPITRE 11 Comprendre et utiliser les sessions	245 À quoi servent
les sessions	246 Les sessions selon
Django	247 Utilisation d'une session ..
.....	248 Configuration
.....	248 Maniement d'une variable de session
.....	249 Enregistrement de l'utilisateur authentifié
.....	250 Vérification que l'utilisateur est bien authentifié
Utilisation des données de la session	251 Que trouve-t-
on dans le cookie ?	252 Que trouve-t-on dans la
session ?	252 Protection des pages privées
.....	253 Amélioration de notre page d'accueil

.....	255 Personnalisation de la bannière
.....	255 Division du corps de la page
.....	258 Liste des messages 260
Récupération des messages de la liste	260 Présentation de la liste des messages
.....	263 Liste des amis
.....	265 Publication d'un message à destination de ses amis
.....	266 Récapitulatif du passage d'informations entre pages
.....	267 Le formulaire de type POST ou GET
....	268 L'URL accompagnée de paramètres 269 La session
.....	269 Ai-je bien compris ?
.....	270 CHAPITRE 12 En finir avec Trombinoscoop
.....	271 La page d'ajout d'un ami
.....	272 Ajout d'un formulaire dans forms.py
.....	272 Création de la vue add_friend
....	273 Création du template add_friend.html 274
Ajout d'une URL dans urls.py	274 Ajout du lien dans la page d'accueil
.....	274 XXII Apprendre la programmation web avec Python et Django La page de visualisation d'un profil
.....	275 Création du template show_profil.html
.....	276 Création de la vue show_profil
.....	277 Ajout de l'URL dans urls.py 278
Amélioration de la présentation dans style.css	278 Ajout des liens dans la page d'accueil
.....	279 La page de modification d'un profil
.....	280 Création du template modify_profil.html
.....	281 Création de la vue modify_profil
.....	281 Ajout de l'URL dans urls.py
.....	282 Ajout des liens dans la page d'accueil
.....	283 Ai-je bien compris ?
283 CHAPITRE 13 Des sites web encore plus dynamiques avec Ajax	285 Exemple de l'interactivité attendue entre client et serveur
.....	286 Validation de la création d'un compte avec Ajax
.....	289 Ajout d'une URL pour la requête Ajax
.....	290 Vue traitant la requête Ajax
.....	290 Ajout du code JavaScript
.....	292 Déetecter que l'utilisateur a bien terminé de remplir le champ Email
.....	292 Validation du courriel saisi
.....	294 Insérer la liste des erreurs au-dessus du champ « fautif »
.....	296 Ajout d'un ami via Ajax
.....	297 Ajout d'un champ texte et d'un lien
.....	298 Ajout de l'URL et de la vue
.....	299 Création du JavaScript d'ajout d'un ami
.....	300 Insertion du HTML dans la page web 301

Ai-je bien compris ?	302
CHAPITRE 14 Mettre en ligne son site web	303
types de serveurs	304
déploiement	305
.	312
.	312
.	312
.	318
.	321
.	326
Désactiver le mode DEBUG	327
matières Mise en place de Gunicorn	328
Prendre en charge les fichiers statiques	331
son nom de domaine	333
.	336
l'environnement de développement 339 Que faut-il installer ?	
.	340
.	340
.	341
.	341
.	342
.	344
.	344
.	347
.	348
.	349
.	349
.	350
.	351
.	352
.	353
.	353
.	353
.	354
Premie	

BLACKBOXAI

Voici un aperçu détaillé des concepts et des exercices pour chaque chapitre de votre table des matières, avec des exemples de code et des instructions sur la façon de procéder.

Cours de Programmation Web avec Python et Django

PREMIÈRE PARTIE : Les notions essentielles

CHAPITRE 1 : Comment un site web fonctionne-t-il ?

Qu'est-ce que le Web ? Le Web est un système d'information qui permet d'accéder à des documents et des ressources via Internet. Il utilise le protocole HTTP pour la communication entre les clients (navigateurs) et les serveurs.

Fonctionnement d'un site web statique Un site web statique est composé de fichiers HTML, CSS et JavaScript qui sont servis tels quels par le serveur. Les pages ne changent pas en fonction des interactions de l'utilisateur.

Exercice : Créer une page web statique

- 1. Créez un fichier `index.html` avec le contenu suivant :

```
html
```

Bienvenue sur mon site statique

Cette page utilise HTML et CSS.

- 2. Ouvrez ce fichier dans un navigateur pour voir le résultat.

Le protocole HTTP HTTP (HyperText Transfer Protocol) est le protocole utilisé pour transférer des données sur le Web. Il fonctionne sur un modèle client-serveur.

L'URL, adresse d'une page web Une URL (Uniform Resource Locator) est l'adresse d'une ressource sur le Web. Elle se compose de plusieurs parties : le protocole (http/https), le nom de domaine, et le chemin d'accès à la ressource.

Le serveur web : à la fois ordinateur et logiciel Un serveur web est un ordinateur qui stocke des fichiers de site web et les sert aux utilisateurs via Internet. Le logiciel serveur (comme Apache ou Nginx) gère les requêtes HTTP.

Des sites web qui se mettent à jour tout seuls Les sites dynamiques utilisent des langages de programmation côté serveur (comme PHP, Python) pour générer des pages web en temps réel, souvent en interagissant avec une base de données.

Fonctionnement d'un site web dynamique Un site web dynamique génère des pages à la volée, en fonction des requêtes des utilisateurs. Cela permet de personnaliser le contenu affiché.

Utilisation d'une base de données Les bases de données stockent des données de manière structurée. Les sites dynamiques utilisent des bases de données pour récupérer et afficher des informations en fonction des interactions des utilisateurs.

Passage de paramètres à une page web Les paramètres peuvent être passés à une page web via les méthodes GET (dans l'URL) ou POST (dans le corps de la requête). Cela permet de transmettre des données entre le client et le serveur.

CHAPITRE 2 : Programmation orientée objet et framework MVC

Des programmes modulaires La modularité permet de diviser un programme en plusieurs modules indépendants, facilitant ainsi la maintenance et la réutilisation du code.

Une écriture simplifiée La programmation orientée objet (POO) permet de modéliser des objets du monde réel en utilisant des classes et des objets, rendant le code plus intuitif.

Des modules réutilisables Les modules peuvent être importés et utilisés dans différents projets, ce qui réduit la duplication de code et améliore l'efficacité.

Un travail d'équipe facilité et plus efficace La POO et les frameworks comme Django facilitent la collaboration entre développeurs en permettant une séparation claire des responsabilités.

Les langages orientés objet Des langages comme Python, Java et C++ supportent la POO, permettant de créer des applications complexes de manière structurée.

Les cas d'utilisation (use cases) et le MVC Les cas d'utilisation décrivent comment les utilisateurs interagissent avec le système. Le modèle MVC (Modèle-Vue-Contrôleur) est une architecture qui sépare la logique de l'application de l'interface utilisateur.

Le principe de la séparation modèle-vue-contrôleur (MVC)

- Modèle : Gère les données et la logique de l'application.
- Vue : Présente les données à l'utilisateur.
- Contrôleur : Gère les interactions de l'utilisateur et met à jour le modèle.

Django et le MVC Django utilise le modèle MVC, mais l'appelle MVT (Modèle-Vue-Template). Les templates sont utilisés pour générer le HTML à afficher.

CHAPITRE 3 : Bases du langage Python

Qualités et défauts de Python Python est apprécié pour sa simplicité et sa lisibilité, mais peut être moins performant que d'autres langages compilés.

Les bases : variables et mémoire centrale Les variables en Python sont des conteneurs pour stocker des données. Python gère la mémoire automatiquement.

Les instructions de contrôle Les instructions conditionnelles (if, else) et les boucles (for, while) permettent de contrôler le flux d'exécution du programme.

Exercice : Écrire un programme Python

```
python
```

Exemple de programme simple

```
age = int(input("Entrez votre âge : "))
if age < 18:
    print("Vous êtes mineur.")
else:
    print("Vous êtes majeur.")
```

Les fonctions Les fonctions sont des blocs de code réutilisables qui effectuent une tâche spécifique. Elles peuvent prendre des paramètres et retourner des valeurs.

La programmation objet La POO en Python permet de créer des classes et des objets, facilitant l'organisation et la gestion du code.

CHAPITRE 4 : Bases du HTML 5, CSS et JavaScript

Structurer les pages web avec HTML 5 HTML 5 est la dernière version du langage HTML, qui permet de structurer le contenu des pages web.

Exercice : Créer une page HTML simple

```
html
```

Bienvenue sur ma page

Ceci est un paragraphe.

Mise en forme avec les feuilles de styles CSS CSS (Cascading Style Sheets) est utilisé pour styliser les pages web.

Exercice : Ajouter du CSS à votre page

html

Bienvenue sur ma page stylisée

Ceci est un paragraphe avec un fond gris.

CHAPITRE 5 : Mise en application : un site web inspiré de Facebook

Notre site web : Trombinoscoop Dans ce chapitre, vous allez créer un site web inspiré de Facebook, en utilisant Django pour gérer les utilisateurs et les interactions.

Exercice : Créer un projet Django

- 1. Installez Django :

bash

```
pip install django  
Generated by BlackBox AI
```

- 2. Créez un nouveau projet :

```
bash
django-admin startproject trombinoscoop
cd trombinoscoop
python manage.py startapp users
```

- 3. Ajoutez l'application users dans settings.py :

```
python
INSTALLED_APPS = [
    ...
    'users',
]
```

- 4. Créez un modèle d'utilisateur dans models.py :

```
python
from django.db import models

class User(models.Model):
    username = models.CharField(max_length=100)
        email = models.EmailField()
    password = models.CharField(max_length=100)
```

- 5. Créez les migrations et appliquez-les :

```
bash
python manage.py makemigrations
python manage.py migrate
```

DEUXIÈME PARTIE : Mise en application avec Django

CHAPITRE 7 : Les vues Django

Exercice : Créer une vue simple Fichier : views.py

```
python
from django.http import HttpResponse
```

```
def hello(request):
    return HttpResponseRedirect("")
```

Bonjour le monde !

Fichier : urls.py

```
python
from django.urls import path
from . import views

urlpatterns = [
    path('hello/', views.hello, name='hello'),
]
```

CHAPITRE 8 : Templates Django

Exercice : Utiliser un template Fichier : template.html

html

`{{ message }}`

Fichier : views.py

```
python
from django.shortcuts import render

def show_message(request):
    context = {
        'title': 'Mon Template',
        'message': 'Bienvenue sur mon site avec Django !'
    }
    Generated by BlackBoxAI
```

```
        }
    return render(request, 'template.html', context)
```

CHAPITRE 9 : Les formulaires Django

Exercice : Créer un formulaire Fichier : forms.py

```
python
from django import forms

class ContactForm(forms.Form):
    nom = forms.CharField(max_length=100)
    email = forms.EmailField()
    message = forms.CharField(widget=forms.Textarea)
```

Fichier : views.py

```
python
from django.shortcuts import render
from .forms import ContactForm

def contact(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            # Traitez les données du formulaire
    return HttpResponse("Merci pour votre message !")
    else:
        form = ContactForm()
    return render(request, 'contact.html', {'form': form})
```

Conclusion

Ces exercices vous permettront de mettre en pratique les concepts abordés dans chaque chapitre. N'hésitez pas à expérimenter et à modifier le code pour mieux comprendre son fonctionnement. Si vous avez des questions ou besoin d'aide supplémentaire, n'hésitez pas à demander !

User

ok donne moi le tout sous forme de livre

BLACKBOXAI

Voici une version structurée de votre cours sous forme de livre. Vous pouvez copier ce contenu dans un document de traitement de texte (comme Microsoft Word ou Google Docs) et l'enregistrer au format PDF.

Apprendre la Programmation Web avec Python et Django

Table des matières

PREMIÈRE PARTIE : Les notions essentielles

- - 1. Comment un site web fonctionne-t-il ?
 - Qu'est-ce que le Web ? - Fonctionnement d'un site web statique - Le protocole HTTP - L'URL, adresse d'une page web - Le serveur web : à la fois ordinateur et logiciel - Des sites web qui se mettent à jour tout seuls - Fonctionnement d'un site web dynamique - Utilisation d'une base de données - Passage de paramètres à une page web - Ai-je bien compris ?
 - 2. Programmation orientée objet et framework MVC
 - Des programmes modulaires - Une écriture simplifiée - Des modules réutilisables - Un travail d'équipe facilité et plus efficace - Les langages orientés objet - Les cas d'utilisation (use cases) et le MVC - Le principe de la séparation modèle-vue-contrôleur (MVC) - Django et le MVC - Ai-je bien compris ?
 - 3. Bases du langage Python
 - Qualités et défauts de Python - Les bases : variables et mémoire centrale - Déclaration et initialisation des variables - Type des variables - Modification et transtypage des variables - Copie de variables - Se renseigner sur une variable - Quelques opérations sur les types simples - Les types composites : listes et dictionnaires - Les instructions de contrôle - Les fonctions - La programmation objet - Ai-je bien compris ?
 - 4. Bases du HTML 5, CSS et JavaScript
 - Structurer les pages web avec HTML 5 - Le concept de « balise » - Structure d'un document HTML - L'encodage de la page - Quelques éléments HTML - Mise en forme avec les feuilles de styles CSS - Ai-je bien compris ?
 - 5. Mise en application : un site web inspiré de Facebook

- Notre site web : Trombinoscoop - Les cas d'utilisation - Maquette du site : les wireframes - Scénario complet du site - Ai-je bien compris ?

DEUXIÈME PARTIE : Mise en application avec Django

- 6. Les vues Django : orchestration et architecture
 - Utilité des vues - Le fichier urls.py - Le fichier views.py - Ai-je bien compris ?
- 7. Les templates Django : séparation et réutilisation des rendus HTML
 - Principe des templates - Dynamisons ce premier template - Ai-je bien compris ?
- 8. Les formulaires Django
 - Patron courant de gestion des formulaires - Validation du formulaire - Ai-je bien compris ?
- 9. Comprendre et utiliser les sessions
 - À quoi servent les sessions - Utilisation d'une session - Ai-je bien compris ?
- 10. En finir avec Trombinoscoop
 - La page d'ajout d'un ami - Ai-je bien compris ?
- 11. Des sites web encore plus dynamiques avec Ajax
 - Exemple de l'interactivité attendue entre client et serveur - Ai-je bien compris ?
- 12. Mettre en ligne son site web
 - Les différents types de serveurs - Ai-je bien compris ?

PREMIÈRE PARTIE : Les notions essentielles

CHAPITRE 1 : Comment un site web fonctionne-t-il ?

Qu'est-ce que le Web ? Le Web est un système d'information qui permet d'accéder à des documents et des ressources via Internet. Il utilise le protocole HTTP pour la communication entre les clients (navigateurs) et les serveurs.

Fonctionnement d'un site web statique Un site web statique est composé de fichiers HTML, CSS et JavaScript qui sont servis tels quels par le serveur. Les pages ne changent pas en fonction des interactions de l'utilisateur.

Exercice : Créer une page web statique

- 1. Créez un fichier `index.html` avec le contenu suivant :

```
html
```

Bienvenue sur mon site statique

Cette page utilise HTML et CSS.

- 2. Ouvrez ce fichier dans un navigateur pour voir le résultat.

Le protocole HTTP HTTP (HyperText Transfer Protocol) est le protocole utilisé pour transférer des données sur le Web. Il fonctionne sur un modèle client-serveur.

L'URL, adresse d'une page web Une URL (Uniform Resource Locator) est l'adresse d'une ressource sur le Web. Elle se compose de plusieurs parties : le protocole (http/https), le nom de domaine, et le chemin d'accès à la ressource.

Le serveur web : à la fois ordinateur et logiciel Un serveur web est un ordinateur qui stocke des fichiers de site web et les sert aux utilisateurs via Internet. Le logiciel serveur (comme Apache ou Nginx) gère les requêtes HTTP.

Des sites web qui se mettent à jour tout seuls Les sites dynamiques utilisent des langages de programmation côté serveur (comme PHP, Python) pour générer des pages web en temps réel, souvent en interagissant avec une base de données.

Fonctionnement d'un site web dynamique Un site web dynamique génère des pages à la volée, en fonction des requêtes des utilisateurs. Cela permet de personnaliser le contenu affiché.

Utilisation d'une base de données Les bases de données stockent des données de manière structurée. Les sites dynamiques utilisent des bases de données pour récupérer et

afficher des informations en fonction des interactions des utilisateurs.

Passage de paramètres à une page web Les paramètres peuvent être passés à une page web via les méthodes GET (dans l'URL) ou POST (dans le corps de la requête). Cela permet de transmettre des données entre le client et le serveur.

CHAPITRE 2 : Programmation orientée objet et framework MVC

Des programmes modulaires La modularité permet de diviser un programme en plusieurs modules indépendants, facilitant ainsi la maintenance et la réutilisation du code.

Une écriture simplifiée La programmation orientée objet (POO) permet de modéliser des objets du monde réel en utilisant des classes et des objets, rendant le code plus intuitif.

Des modules réutilisables Les modules peuvent être importés et utilisés dans différents projets, ce qui réduit la duplication de code et améliore l'efficacité.

Un travail d'équipe facilité et plus efficace La POO et les frameworks comme Django facilitent la collaboration entre développeurs en permettant une séparation claire des responsabilités.

Les langages orientés objet Des langages comme Python, Java et C++ supportent la POO, permettant de créer des applications complexes de manière structurée.

Les cas d'utilisation (use cases) et le MVC Les cas d'utilisation décrivent comment les utilisateurs interagissent avec le système. Le modèle MVC (Modèle-Vue-Contrôleur) est une architecture qui sépare la logique de l'application de l'interface utilisateur.

Le principe de la séparation modèle-vue-contrôleur (MVC)

- Modèle : Gère les données et la logique de l'application.
- Vue : Présente les données à l'utilisateur.
- Contrôleur : Gère les interactions de l'utilisateur et met à jour le modèle.

Django et le MVC Django utilise le modèle MVC, mais l'appelle MVT (Modèle-Vue-Template). Les templates sont utilisés pour générer le HTML à afficher.

CHAPITRE 3 : Bases du langage Python

Qualités et défauts de Python Python est apprécié pour sa simplicité et sa lisibilité, mais peut être moins performant que d'autres langages compilés.

Les bases : variables et mémoire centrale Les variables en Python sont des conteneurs pour stocker des données. Python gère la mémoire automatiquement.

Les instructions de contrôle Les instructions conditionnelles (if, else) et les boucles (for, while) permettent de contrôler le flux d'exécution du programme.

Exercice : Écrire un programme Python

python

Exemple de programme simple

```
age = int(input("Entrez votre âge : "))
if age < 18:
    print("Vous êtes mineur.")
else:
    print("Vous êtes majeur.")
```

Les fonctions Les fonctions sont des blocs de code réutilisables qui effectuent une tâche spécifique. Elles peuvent prendre des paramètres et retourner des valeurs.

La programmation objet La POO en Python permet de créer des classes et des objets, facilitant l'organisation et la gestion du code.

CHAPITRE 4 : Bases du HTML 5, CSS et JavaScript

Structurer les pages web avec HTML 5 HTML 5 est la dernière version du langage HTML, qui permet de structurer le contenu des pages web.

Exercice : Créer une page HTML simple

html

Bienvenue sur ma page

Ceci est un paragraphe.

Mise en forme avec les feuilles de styles CSS CSS (Cascading Style Sheets) est utilisé pour styliser les pages web.

Exercice : Ajouter du CSS à votre page

html

Bienvenue sur ma page stylisée

Ceci est un paragraphe avec un fond gris.

CHAPITRE 5 : Mise en application : un site web inspiré de Facebook

Notre site web : Trombinoscoop Dans ce chapitre, vous allez créer un site web inspiré de Facebook, en utilisant Django pour gérer les utilisateurs et les interactions.

Exercice : Créer un projet Django

- 1. Installez Django :

```
bash  
pip install django
```

- 2. Créez un nouveau projet :

```
bash  
django-admin startproject trombinoscoop  
cd trombinoscoop  
python manage.py startapp users
```

- 3. Ajoutez l'application users dans settings.py :

```
python  
INSTALLED_APPS = [  
Generated by BlackBox AI
```

```
'users',
```

```
]
```

- 4. Créez un modèle d'utilisateur dans `models.py` :

```
python
from django.db import models

class User(models.Model):
    username = models.CharField(max_length=100)
        email = models.EmailField()
    password = models.CharField(max_length=100)
```

- 5. Créez les migrations et appliquez-les :

```
bash
python manage.py makemigrations
python manage.py migrate
```

DEUXIÈME PARTIE : Mise en application avec Django

CHAPITRE 7 : Les vues Django

Exercice : Créer une vue simple Fichier : `views.py`

```
python
from django.http import HttpResponse

def hello(request):
    return HttpResponse("
```

Bonjour le monde !

")

Fichier : `urls.py`

```
python
from django.urls import path
from . import views
```

```
urlpatterns = [
    path('hello/', views.hello, name='hello'),
]
```

CHAPITRE 8 : Templates Django

Exercice : Utiliser un template Fichier : template.html

html

`{{ message }}`

Fichier : views.py

```
python
from django.shortcuts import render

def show_message(request):
    context = {
        'title': 'Mon Template',
        'message': 'Bienvenue sur mon site avec Django !'
    }
return render(request, 'template.html', context)
```

CHAPITRE 9 : Les formulaires Django

Exercice : Créer un formulaire Fichier : forms.py

```
python
from django import forms
```

```
class ContactForm(forms.Form):
    nom = forms.CharField(max_length=100)
        email = forms.EmailField()
    message = forms.CharField(widget=forms.Textarea)
```

Fichier : views.py

```
python
from django.shortcuts import render
from .forms import ContactForm


def contact(request):
    if request.method == 'POST':
        form = ContactForm(request.POST)
        if form.is_valid():
            # Traitez les données du formulaire
        return HttpResponseRedirect("Merci pour votre message !")
    else:
        form = ContactForm()
    return render(request, 'contact.html', {'form': form})
```

Conclusion

Ce livre vous a fourni une introduction complète à la programmation web avec Python et Django. En suivant les exercices et en expérimentant avec le code, vous serez en mesure de créer vos propres applications web dynamiques. Si vous avez des questions ou besoin d'aide supplémentaire, n'hésitez pas à demander !

Vous pouvez maintenant copier ce contenu dans un document de traitement de texte, le formater selon vos préférences, et l'enregistrer au format PDF. Si vous avez besoin d'autres sections ou d'informations supplémentaires, faites-le moi savoir !