Examen Blanc

Dans MVT, la logique métier est principalement dans :
☐ Template
☑ Model
☐ View
☐ URL

settings.py contient surtout :
☐ Les commandes pip d'installation
☐ Les routes HTML uniquement
☐ Les templates graphiques
☑ La configuration globale

Le rôle du point . dans startproject config . est de :
☑ Créer le projet dans le dossier courant
☐ Supprimer un dossier inutile automatiquement
☐ Installer Django en même temps
☐ Lancer le serveur directement

Vérifier la version de Django installé se fait via :
☐ pip show --version django
☑ django-admin --version
☐ manage.py version
☐ py django –v

Pour activer l'environnement virtuel sous Windows (chemin donné), on utilise
☑ venv/Scripts/activate
☐ venv\Scripts\activate
☐ activate venv
☐ source venv

La commande de création d'un environnement virtuel (selon le support) est :
☐ pip install venv
☑ py -m venv venv
☐ django-admin venv
☐ python new venv

Un environnement virtuel sert surtout à :
☐ Accélérer l'exécution des projets
☐ Éviter d'écrire tous le code manuellement
☐ Remplacer le Python natif par Django
☑ Isoler les bibliothèques et versions

Une méthode abstraite :
☐ Peut être ignorée sans conséquence
☑ Doit être implémentée dans les sous-classes concrètes
☐ Empêche toute création d'objet Python
☐ Transforme une classe en module

La composition est plutôt adaptée quand :
☐ On veut une relation "est-un"
☑ On veut une relation "contient-un"
☐ On veut faire du SQL
☐ On veut remplacer Python

L'héritage est plutôt adapté quand :
☐ On veut éviter toute réutilisation
☐ On a une relation "contient-un"
☑ On a une relation "est-un"
☐ On veut supprimer les classes

Le couple "données + actions" correspond à :
☑ Attributs + méthodes
☐ Imports + modules
☐ URL + serveur
☐ HTML + CSS

Une méthode statique est particulièrement utile quand :
☐ On veut modifier cls à chaque appel
☐ On veut rendre la classe abstraite
☐ On veut accéder à self absolument
☑ On veut une fonction liée à la classe

Dans une méthode d'instance, modifier l'état de l'objet se fait via :
☐ cls.attribut = ...
☑ self.attribut = ...

☐ global attribut
☐ settings.attribut = ...

Pour lancer le serveur de développement, on utilise :
☐ django-admin runserver
☑ py manage.py runserver
☐ python server start
☐ pip run django

Une property sert surtout à :
☐ Remplacer complètement les méthodes
☐ Créer et liée à une base de données
☐ Empêcher toute validation réalisée
☑ Exposer un attribut avec contrôle

Un attribut dynamique est :
☑ Un attribut ajouté après la création de l'objet
☐ Un attribut défini uniquement dans settings.py
☐ Un attribut qui peut changer de valeur lors de la création
☐ Un attribut qui compile le code

Les attributs d'instance sont en général définis :
☑ Dans __init__ via self.attribut = ...
☐ Uniquement dans __repr__
☐ Seulement avec global
☐ Uniquement hors de la classe

Python est décrit comme un langage :
☐ Mono-paradigme uniquement procédural
☑ Multi-paradigme
☐ Uniquement logique
☐ Uniquement déclaratif

L'encapsulation consiste surtout à :
☐ Mettre tout en variables globales
☑ Cacher les détails internes
☐ Rendre toutes les méthode « private »
☐ Interdire l'héritage systématiquement

Une instance est :
☑ Un objet créé à partir d'une classe
☐ Un modèle théorique de la classe mère
☐ Une variable qui stock l'état instantané
☐ Une variable de configuration instantané

bool(27) renvoie :
☑ True
☐ False
☐ "True"
☐ Aucune réponse

Pour concaténer deux chaînes a et b, on utilise généralement :
☐ a & b
☐ a && b
☑ a + b
☐ concat(a,b)

Pourquoi séparer projet et application ?
☐ Pour accélérer Django
☑ Pour modularité et réutilisation
☐ Pour éviter Python
☐ Pour des raisons graphiques

Le rôle principal d'une vue Django est :
☐ Générer du HTML
☑ Traiter la requête et retourner une réponse
☐ Définir la base de données
☐ Créer des URLs

Quel fichier n'est PAS créé par startproject ?
☐ settings.py
☐ urls.py
☑ views.py
☐ manage.py