Full Name

UE12P25 – AP

Programmation Elixir

Quiz du cours #2

Page	Possible	Score
1	10	
2	10	
Total:	**20**	

Cochez une seule réponse par question.

Questions 1–13 : rappels de cours. 14–20 : un cran plus techniques.

1. Que renvoie `self()` ? 1 [1]

- ☐ Nom du module
- ☐ PID courant
- ☐ Nom du nœud
- ☐ PID parent

2. Quelles fonctions pour une boucle de messages ? 2 [1]

- ☐ `spawn/1` et `send/2`
- ☐ `send/2` et `receive`
- ☐ `IO.puts/1` et `receive`
- ☐ `Task.async/1` et `Task.await/1`

3. Bonne syntaxe pour envoyer à `pid` ? 3 [1]

- ☐ `pid <- {:hello, 42}`
- ☐ `send(pid, {:hello, 42})`
- ☐ `Process.send(pid)`
- ☐ `deliver(pid, {:hello, 42})`

4. Effet de `Process.link(pid)` ? 4 [1]

- ☐ Canal de messages
- ☐ Cycle de vie lié
- ☐ Monitoring unidirectionnel
- ☐ Priorité CPU

5. Format message `monitor` quand stop ? 5 [1]

- ☐ `{:DOWN, ref, :process, pid, reason}`
- ☐ `{:EXIT, pid, reason}`
- ☐ `{:DOWN, pid, ref, reason}`
- ☐ `{:EXIT, ref, :process, pid}`

6. Retour de `Process.monitor(pid)` ? 6 [1]

- ☐ `pid`
- ☐ `reference()`
- ☐ Booléen
- ☐ `%Monitor{}`

7. Avantage de `Task.async/await` sur `spawn` ? 7 [1]

- ☐ Résultats + timeout facile
- ☐ Moins mémoire
- ☐ Exécute en C
- ☐ Ordre garanti

8. Ordre de `Task.async_stream/3` par défaut ? 8 [1]

- ☐ Ordre d'entrée
- ☐ Ordre de fin des tâches
- ☐ Aléatoire
- ☐ Trié par durée

9. API pour map parallèle avec limite ? 9 [1]

- ☐ `Task.async/1`
- ☐ `Task.async_stream/3`
- ☐ `Stream.map/2`
- ☐ `Enum.map/2`

10. Effet de `max_concurrency: 4` ? 10 [1]

- ☐ 4 tâches au total
- ☐ 4 simultanées max
- ☐ 4 nœuds distribués
- ☐ Priorité 4

Points on this page: 10

11. Callbacks classiques d'un GenServer ? 11 [1] [ ]

- [ ] `handle_info/2` et `handle_call/3`
- [ ] `handle_cast/2` et `handle_call/3`
- [ ] `init/1` et `terminate/2`
- [ ] `start_link/1` et `stop/1`

12. Retour valide de `init/1` ? 12 [1] [ ]

- [ ] `:ok`
- [ ] `{:ok, state}`
- [ ] `{:noreply, state}`
- [ ] `{:reply, state}`

13. Rôle d'un superviseur ? 13 [1] [ ]

- [ ] Planif CPU
- [ ] Redémarrer enfants
- [ ] Compiler modules
- [ ] GC

14. Stratégie qui relance seul le crashé ? 14 [1] [ ]

- [ ] `:rest_for_one`
- [ ] `:one_for_all`
- [ ] `:one_for_one`
- [ ] `:simple_one_for_one`

15. Que renvoie `Task.async_stream/3` ? 15 [1] [ ]

- [ ] Liste brute
- [ ] Flux `{:ok, res}` / `{:exit, reason}`
- [ ] PIDs
- [ ] Task structs

16. Effet de `Process.unlink(pid)` ? 16 [1] [ ]

- [ ] Détruit la tâche
- [ ] Supprime le lien
- [ ] Stoppe le monitor
- [ ] Envoie vers handle_info

17. Différence cast vs call ? 17 [1] [ ]

- [ ] `cast = async, call = sync`
- [ ] cast crée lien, call crée monitor
- [ ] cast = erreurs, call = succès
- [ ] Aucune

18. Usage de Task.Supervisor ? 18 [1] [ ]

- [ ] Superviser des tâches isolées
- [ ] Mesurer perf
- [ ] Convertir Task en GenServer
- [ ] Jamais

19. Que fait `Task.await/2` si la tâche crash ? 19 [1] [ ]

- [ ] Renvoie l'ancien résultat
- [ ] Lève une exception
- [ ] Retourne `{:error, reason}`
- [ ] Ignore l'erreur

20. Quel est le bon cas d'usabe pour `Task.sync_stream` ? 20 [1] [ ]

- [ ] IO massif avec limite parallélisme
- [ ] Serveur long avec receive
- [ ] Stocker un état persistant
- [ ] Tracer sorties avec :dbg

Points on this page: 10 [ ]