

Page	Possible	Score
1	12	
2	8	
Total:	20	

UE12P26 – PE

Python numérique - Quiz du cours #3

6 octobre 2026

Cochez une seule réponse par question.

Ce quiz porte sur la séance 3 (pandas : tables étiquetées). Les questions 1 à 12 sont accessibles, les questions 13 à 20 sont plus techniques : lisez bien le code avant de répondre. Vous disposez de 10 minutes pour répondre aux 20 questions.

1. Quelle instruction lit le fichier `f.csv`, dont le séparateur est `;`, en prenant la colonne `id` comme index des lignes ?

- ☐ `pd.read_csv("f.csv", sep=";", index_col="id")`
- ☐ `pd.read_csv("f.csv", separator=";", index="id")`
- ☐ `pd.read_table("f.csv", ";", index=0)`
- ☐ `pd.open_csv("f.csv", sep=";", index_col="id")`

2. Quelle est la différence entre `df["Age"]` et `df[["Age"]]` ?

- ☐ Aucune, les deux écritures sont équivalentes
- ☐ La première renvoie une liste Python, la seconde une `Series`
- ☐ La première renvoie une `Series`, la seconde une `DataFrame` à une colonne
- ☐ La première renvoie une vue, la seconde une copie

3. Que fournit `df.info()` ?

- ☐ La moyenne, l'écart-type et les quartiles des colonnes numériques
- ☐ Pour chaque colonne : son nom, son nombre de valeurs non manquantes et son `dtype`
- ☐ Les 5 premières lignes de la table
- ☐ Le couple (nombre de lignes, nombre de colonnes)

4. Si `df.shape` vaut `(891, 12)`, que vaut `len(df)` ?

- ☐ 12
- ☐ 10692
- ☐ `(891, 12)`
- ☐ 891

5. `df` est indexée par `PassengerId`. Quelle expression renvoie le nom du passager dont l'identifiant vaut 5 ?

- ☐ `df.loc[5, "Name"]`
- ☐ `df.iloc[5, "Name"]`
- ☐ `df["Name", 5]`
- ☐ `df.loc["Name", 5]`

6. Que renvoie `df["Embarked"].value_counts()` ?

- ☐ Le nombre de valeurs distinctes de la colonne
- ☐ Une `Series` indexée par les valeurs distinctes, donnant leur nombre d'occurrences (ordre décroissant)
- ☐ Un dictionnaire Python
- ☐ Le nombre de valeurs non manquantes

7. Quelle expression donne le nombre de valeurs distinctes de `df["Pclass"]` ?

- ☐ `df["Pclass"].unique()`
- ☐ `df["Pclass"].count()`
- ☐ `len(df["Pclass"].unique())`
- ☐ `len(df["Pclass"])`

8. Quelle écriture sélectionne les femmes de plus de 60 ans ?

- ☐ `df[(df["Sex"] == "female") & (df["Age"] > 60)]`
- ☐ `df[(df["Sex"] == "female") and (df["Age"] > 60)]`
- ☐ `df[df["Sex"] == "female" & df["Age"] > 60]`
- ☐ `df.loc["Sex" == "female", "Age" > 60]`

9. On note `s = df["SibSp"] + df["Parch"] + 1`. Quelle écriture range `s` dans une nouvelle colonne `Family` de `df` ?

- ☐ `df.add_column("Family", s)`
- ☐ `df[Family] = s`
- ☐ `df.Family = s`
- ☐ `df["Family"] = s`

10. Quelle instruction supprime la colonne `Cabin` ?

- ☐ `df.remove("Cabin")`
- ☐ `df = df.drop(columns=["Cabin"])`
- ☐ `del df.Cabin`
- ☐ `df = df.drop("Cabin")`

11. Quelle écriture sélectionne les passagers dont le nom contient "Mrs" ?

- ☐ `df[df["Name"].contains("Mrs")]`
- ☐ `df["Mrs" in df["Name"]]`
- ☐ `df[df["Name"].str.contains("Mrs")]`
- ☐ `df[df["Name"].find("Mrs")]`

12. Que renvoie `df.isna().sum()` ?

- ☐ Le nombre total de valeurs manquantes de la table (un entier)
- ☐ Une `Series` : le nombre de valeurs manquantes de chaque **ligne**
- ☐ Un booléen : vrai s'il existe au moins une valeur manquante
- ☐ Une `Series` : le nombre de valeurs manquantes de chaque **colonne**

13. Qu'affiche ce code ?

```
idx = ["a", "b", "c", "d", "e"]
s = pd.Series([10, 20, 30, 40, 50],
               index=idx)
print(len(s.loc["b":"d"]), len(s.iloc[1:3]))
```

- ☐ 2 2
- ☐ 3 2
- ☐ 3 3
- ☐ 2 3

14. Que vaut `pd.Series([1, 2, None]).dtype` ?

- ☐ float64
- ☐ int64
- ☐ object
- ☐ Une erreur : `None` n'est pas un nombre

15. Qu'affiche ce code ?

```
s = pd.Series(["12", "x", "7"])
t = pd.to_numeric(s, errors="coerce")
print(t.sum())
```

- ☐ Une erreur `ValueError` à cause de "x"
- ☐ "12x7"
- ☐ 19.0
- ☐ NaN

16. Que produit l'expression suivante ?

```
df[(df["Age"] > 60) and (df["Pclass"] == 1)]
```

- ☐ La même chose qu'avec `&`
- ☐ Une `DataFrame` vide
- ☐ Les lignes vérifiant seulement la première condition
- ☐ Une erreur `ValueError` (« truth value of a Series is ambiguous »)

17. Après ce code, la colonne `Age` de `df` a-t-elle été modifiée ?

```
sub = df[df["Pclass"] == 1]
sub["Age"] = 0
```

- ☐ Non : `sub` se comporte comme une copie ; il faut écrire `df.loc[df["Pclass"] == 1, "Age"] = 0`
- ☐ Oui : `sub` est une vue sur `df`
- ☐ Oui, mais seulement sur la première ligne
- ☐ Une erreur est levée : `sub` est en lecture seule

18. Qu'affiche ce code ?

```
df = pd.DataFrame({"a": [1, 2, 3],
                   "b": [4, 5, 6]},
                  index=[10, 20, 30])
print(df.loc[20, "b"], df.iloc[-1, 0])
```

- ☐ 5 6
- ☐ 5 3
- ☐ 2 3
- ☐ Une erreur `KeyError`

13 ☐ 19. Qu'affiche ce code ?

```
df = pd.DataFrame({"id": [7, 8],
                   "v": [1, 2]})
df2 = df.set_index("id")
print(df2.shape, list(df2.columns))
```

- ☐ (2, 2) ['id', 'v']
- ☐ (2, 1) ['id']
- ☐ (1, 2) ['v']
- ☐ (2, 1) ['v']

14 ☐ 20. Qu'affiche ce code ?

```
s = pd.Series([1.0, np.nan, 3.0])
s.fillna(0)
print(s.sum(), s.dropna().sum())
```

- ☐ nan 4.0
- ☐ 4.0 nan
- ☐ 4.0 4.0
- ☐ 0.0 4.0

15 ☐ 1

16 ☐ 1

17 ☐ 1

18 ☐ 1

19 ☐ 1

20 ☐ 1