

IT 資格 道 場

Python 3 エンジニア認定デ ータ分析実践試験 学習用テ キスト

予備知識ゼロから合格ラインへ —— 読んで理解する1冊目

無料サンプル(第1章まで)

Pythonエンジニア育成推進協会「Python データ分析実践試験」出題範囲(主教材
『Pythonデータ分析 実践ハンドブック』)準拠
2026年10月版

ご利用にあたって

- このテキストは「IT資格道場」(www.shikaku-doujou.com)の購入者限定テキストのうち、学習用テキストの第1章までを収録した**無料サンプル**です。どなたでもご覧いただけます。
- 無料サンプルも著作物です。転載・改変しての再配布・二次販売はできません。
- AI モデルの学習データとしてのご利用(学習目的の入力・データセット化を含む)はできません。
- 製品版(学習用テキスト・頻出度別ノート・暗記用ノートの3点)は、ご購入者を識別できる透かし入りのPDFとして提供されます。
- 本PDFはスマートフォン・タブレットでの閲覧に合わせたA5版面・文字サイズで組版しています。

目次

この教材群の使い方(3点セット)

この試験が問うもの

第1章 データ加工概論 [1.1/1.2]

データ加工の目的と流れ [1.1]

分析環境とライブラリの役割 [1.2]

サンプルはここまでです

SAMPLE

この教材群の使い方(3点セット)

種類	役割
① 学習用テキスト	これだけで問題が解けるようになる本編
② 頻出度別ノート	テキストを読んだら次はこれ。頻出順に絞った問題と解説で「解ける」に橋渡し。試験直前の最終チェックにも
③ 暗記用ノート	覚えるべき要点だけを一問一答に凝縮。空き時間の反復と用語の再確認用

使い方: ① 学習用を読む → ② 頻出度別で解き方を掴む → ③ 問題演習で腕試し → ④ 頻出度別に戻って再確認(試験直前もここ)。暗記用は空き時間や用語の再確認に使います。

この試験が問うもの

Python 3 エンジニア認定データ分析実践試験は、pandas と NumPy を中心に、データを読み込んで加工し、評価する力を問う試験です。出題範囲は主教材『Pythonデータ分析 実践ハンドブック 実務で使えるデータ加工のテクニック』の章立てで、章ごとの出題数が公式に決まっています。このテキストの章立ても、主教材の章の順番とおりです。11章は出題が無いので章を置きません。

項目	内容
運営	一般社団法人Pythonエンジニア育成推進協会
問題数	40問(すべて選択問題)
試験時間	75分
合格ライン	正答率70%
出題の多い章	5章(8問)・2章(7問)・3章(6問)・4章と6章(各4問)

本番はコードの実行結果を問う問題が多く出ます。例のコードは、手元の Python で一度実行して出力を確かめてください。出力は pandas 2.3.3・NumPy 2.0.2 の実行結果です。pandas 3 系で結果が変わる所は、その章の本文に書いてあります。

第1章 データ加工概論 [1.1/1.2]

データ加工の目的と流れ [1.1]

データ分析では、集めたデータをそのまま使えることが少なく、整える作業が欠かせません。この章は、準備から提示までの流れと、道具になるライブラリの役割を扱います^[^1]。

データ分析は、前の工程の結果を次の工程が使う順に、5つの工程で進みます。

1. 収集: ファイル、データベース、Web の API などから元のデータを集めます
2. 整形: 欠損値、重複、表記ゆれ、型の不一致を直します
3. 変換: 分析しやすい形や単位にします (文字列を日付型にする、表を結合する、などがこれにあたります)
4. 分析: 集計、統計量の計算、モデルの作成を行います
5. 可視化: 結果をグラフや表にして見せます

収集・整形・変換の3つをデータ加工と呼びます。整える対象は集めた後にしか存在せず、分析は整った入力を前提にし、可視化は分析の結果を描く、という依存があります (前の工程の結果を次の工程が使います)。そのため、順序は収集、整形、変換、分析、可視化の1通りだけです。分析を加工より前に置くと、乱れた値をそのまま集計することになり、可視化を前に置いても、加工前のデータしか描けません。

データ品質は、データが分析に使える状態かどうかの度合いです (欠損値、重複、外れ値、型の不一致、表記ゆれが下げます)。表記ゆれは、同じものが違う

文字列で入っている状態です。人は「りんご」と「リンゴ」を同じと読みますが、Python は別の値として数えます。

```
import pandas as pd

s = pd.Series(['りんご', 'リンゴ', 'りんご', None])
print(s.value_counts().to_dict())
print(s.replace({'リンゴ': 'りんご'}).value_counts().to_dict())
# {'りんご': 2, 'リンゴ': 1}
# {'りんご': 3}
```

`value_counts` は、値ごとの出現回数を、回数の多い順に並べた Series で返します。表記が違えば別の行になります。欠損値だけは数えから外れます（既定は `dropna=True`）。`replace` で表記をそろえると、件数がまとまります。

再現性は、同じ入力から別の人と同じ結果を作り直せることです。保つ方法は次の3つです。

1. 加工の手順を、スクリプトやノートブックのコードとして残します
2. 元のデータは書き換えません。加工後のデータは別のファイルや変数に作ります
3. 乱数を使う処理は、シード(種)を固定します

加工後のファイルに残るのは結果だけで、作り方はコードで伝えます。元のデータを上書きすると、やり直しの出発点に戻れません。手作業の編集は記録に残る形のコードに置き換えると、同じ操作を再現できます。

```
import numpy as np

print(np.random.default_rng(0).integers(0, 10, 3),
      np.random.default_rng(0).integers(0, 10, 3))

# [8 6 5] [8 6 5]
```

文字列の日付は `pd.to_datetime` で日付型に変換します。日付型の列は `dt` アクセサで部品を取り出せ、`dt.year`、`dt.month`、`dt.day` は整数を返します。

```
d = pd.to_datetime(pd.Series(['2026-03-09', '2026-12-25']))
print(d.dt.year.tolist(), d.dt.month.tolist(), d.dt.day.tolist())
print(d.dt.strftime('%m').tolist(), d.dt.strftime('%Y-%m').tolist())

# [2026, 2026] [3, 12] [9, 25]
# ['03', '12'] ['2026-03', '2026-12']
```

'03' のような文字列がほしい時は `strftime('%m')` を使います。月別の集計では、`'%Y-%m'` の列をキーにすると別の年の同じ月が混ざりません。

分析環境とライブラリの役割 [1.2]

Jupyter Notebook は、ブラウザで開く対話的な実行環境です。コードを「セル」の単位で実行でき、結果がセルの直下に出ます。説明を書く Markdown のセルも同じ画面に置けます（ファイルの拡張子は `.ipynb` です）。読み込み、集計、グラフの確認を試行錯誤しながら進める作業に向きます。セルごとに実行して、結果をその場で確かめながら進められるのが Jupyter Notebook の特徴です。

ライブラリ	役割	慣用の import
NumPy	数値の配列(ndarray)を高速に計算する	<code>import numpy as np</code>
pandas	ラベル付きの表(DataFrame)の読み込み、集計、結合	<code>import pandas as pd</code>
matplotlib	グラフを描く	<code>import matplotlib.pyplot as plt</code>
BeautifulSoup	HTML や XML を解析する	<code>from bs4 import BeautifulSoup</code>

行と列にラベルが付いた表の集計や結合は pandas の役割です。NumPy はラベルを持たない配列の計算を担当します (pandas の内部でも使われます)。

pandas の基本の型は 2 つです。Series は値とラベル (index) を持つ 1 次元、DataFrame は各列が Series の 2 次元の表です。

```
df = pd.DataFrame({'a': [1, 2], 'b': [3, 4]})
print(type(df['b']).__name__, type(df[['b']]).__name__)
print(df['b'].to_numpy(), type(df['b'].to_numpy().__name__))

# Series DataFrame
# [3 4] ndarray
```

列名を 1 つ渡す `df['b']` は Series を返します (ndarray がほしい時は `to_numpy()` を呼びます)。列名のリスト `df[['b']]` は、1 列だけでも DataFrame のままです。上の 2 つを並べて表示すると `Series DataFrame` と出

力されます(どちらも列名のリストで指定すれば `DataFrame DataFrame`、どちらも列名 1 つなら `Series Series` です)。

`ndarray` の演算は要素ごとに行われ、Python のリストとは動きが違います。

```
a = np.array([4, 5])
print(a * 3, [4, 5] * 3)
print(a + 10, [4, 5] + [10])
# [12 15] [4, 5, 4, 5, 4, 5]
# [14 15] [4, 5, 10]
```

配列の `* 3` は各要素を 3 倍し、リストの `* 3` は中身を 3 回繰り返します。`+` も、配列では各要素への加算になります(リストでは連結になります)。表示にも違いがあり、`ndarray` は要素を空白で区切った `[12 15]` の形で出力されます。リストはコンマで区切った `[4, 5, 4, 5, 4, 5]` の形で出力されます。

サンプルはここまでです

続き(第2章～巻末)は、Python 3 エンジニア認定データ分析実践試験 問題集のご購入で、頻出度別ノート・暗記用ノートと合わせて3点すべてダウンロードできます。

SAMPLE

Python 3 エンジニア認定データ分析実践試験 学習用テキスト(無料サンプル)

版

2026年10月版(Pythonエンジニア育成推進協会「Python データ分析実践試験」出題範囲(主教材『Pythonデータ分析 実践ハンドブック』) 準拠)

発行

IT資格道場(<https://www.shikaku-doujou.com>)

お問い合わせ

info@shikaku-doujou.com

Python は Python Software Foundation の登録商標です。記載されているその他の会社名・製品名・サービス名は、各社の商標または登録商標です。

本教材は公式の出題範囲(主教材の章立て)と pandas・NumPy などの公式ドキュメントの内容を基に作成したもので、一般社団法人Pythonエンジニア育成推進協会との提携・承認関係はありません。

本文・図表・設問はすべてオリジナルの著作物です。無断での転載・複製・再配布・二次販売はできません。お気づきの点は info@shikaku-doujou.com までご連絡ください。