OpenEDG™ Python Institute
Certifying Python Skills for Real Careers
https://pythoninstitute.org

PCED™ – Certified Entry-Level Data Analyst with Python 試験シラバス

（試験コード：PCED-30-02）

最終更新日：2025年7月14日

モジュール1：データとデータ分析の基礎概念（9）

1.1 データの定義と分類（2）

1.1.1 データの定義と意味の説明

A. データの定義と意思決定、ビジネス、日常生活における役割の説明
B. データ、情報、知識の違い、および生データが処理・解釈によって意味をもつ仕組みの説明
C. 生データを意思決定に活用できる洞察へと変換する流れの説明

1.1.2 データの型と形式による分類

A. 定量データ・定性データへの分類と識別
B. 構造化データ、半構造化データ、非構造化データを実世界の例を用いて区別

1.2 データソース・収集方法・保存方法の説明（2）

1.2.1 データソースと収集方法の特定

A. API、Webページ、データベース、IoT機器、調査、ログなどさまざまなデータソースの特定と説明
B. 調査、インタビュー、観察、自動化システム、Webスクレイピングなどの代表的なデータ収集方法の説明
C. 代表サンプリングの役割、および偏った・不完全なデータの影響に関する説明
D. 定性・定量研究における各収集技術の長所・短所の比較

1.2.2 データの保存と整理方法の説明

A. データの形式（CSV、JSON、Excel、データベース）や保存システム（データレイク、ウェアハウス、リレーショナルデータベース）の説明
B. メタデータの役割、データの型・構造・用途に応じたストレージソリューションの比較
C. 構造・規模・用途に応じた最適なストレージオプションの評価

1.3 データのライフサイクルとその管理の説明（2）

1.3.1 データライフサイクルの説明

A. データライフサイクルの各段階：収集、保存、処理、分析、可視化/レポート、アーカイブ、削除のリスト化と説明
B. 各段階でのエラー（例：欠損、不正確、不適切な保存）が、結果や意思決定に与える影響の説明
C. ライフサイクルの各段階に関連するツールや技術の特定

1.3.2 ライフサイクル管理の価値と課題の検討

A. 品質・セキュリティ・コンプライアンス保証における、ライフサイクル全体を管理する重要性の説明
B. 大規模データ管理における課題と対応策（例：クライドストレージやデータパイプライン）の説明

1.4 データサイエンス・分析・解析範囲の理解（2）

1.4.1 データ分析・データアナリティクス・データサイエンスの違いの理解

A. データ分析、データアナリティクス、データサイエンスの定義と、それぞれの関係性の説明
B. 実例を用いたそれぞれの範囲、使用ツール、目的の比較
C. 各領域におけるプロフェッショナルの役割と責任の説明
D. 各分野に該当する典型的なタスク（例：要約統計量と機械学習モデリング）の特定

1.4.2 データアナリティクスのワークフロー

A. アナリティクスの4大分類の説明：記述的・診断的・予測的・処方的
B. 各種類のアナリティクスが答える質問と、ビジネス上の重要性の説明
C. データアナリティクスの主要プロセスの説明：データ収集・事前処理・分析・レポート
D. 各アナリティクスごとに、実際の現場でのシナリオ例を対応させる

1.5 データアナリティクスにおける倫理・法的配慮（1）

1.5.1 重要な倫理原則と法的枠組み

A. データ取り扱いにおける透明性、同意、プライバシー、公平性、責任の説明
B. 主要な法規（GDPR、HIPAA、CCPA）の特定および責任あるデータ利用の指針の説明
C. 匿名化や暗号化などの倫理的・公的準拠を支える手法の説明

モジュール2：データ分析のためのPython基礎（13）

2.1 変数とデータ型の操作（1）

2.1.1 変数やデータ型の活用と基本演算の実施

A. 代入演算子「=」で変数を定義・代入
B. 数値の基本的な演算（加算・減算など）、文字列の操作（連結・繰り返し）を行う
C. type() や instance() で変数の型を調べる
D. 代表的なPythonデータ型（int、float、str、bool）の識別

2.2 Pythonのデータコレクションとシーケンスの活用（4）

2.2.1 リストの作成と操作

A. インデックスやスライスでリスト要素にアクセス
B. append()、insert()、pool()、remove()、sort()、reverse()、count()、index() などでリストを操作
C. リスト内包表記を用いてデータ変換や抽出処理を行う

2.2.2 タプルと集合の操作

A. インデックスでタプル要素にアクセス
B. タプルの不変性とリストの使い分けを説明
C. 集合の作成、および add()、remove()、union()、intersection()、isdisjoint()、difference() などの集合を操作
D. 集合による重複排除や要素確認を行う

2.2.3 辞書（dict）による保存・分類・検索

A. キーと値で辞書を作成
B. キーを用いた値を参照・更新・削除
C. dict.get() を活用して、デフォルト値付きで値を取得

D. for key in dict や items() で辞書をループ
E. 辞書を用いた集計、検索、カテゴリ分け
F. リストと辞書の組み合わせによるデータ表現（例：[{ ‘ product ’ : ‘ Laptop’ , ‘price ‘ : 999 }, …]）

2.2.4 文字列をシーケンスとして扱い、文字列メソッドを応用

A. 文字列のインデックス・スライス・ループ処理
B. startswith()、endswith()、find()、capitalize()、isdigit()、isalpha() などの文字列メソッドの利用

2.3 関数と例外処理（3）

2.3.1 関数の定義と呼び出し

A. def キーワードで繰り返し使える関数を作成
B. パラメータを使って値を関数に渡す（位置パラメータ、キーワードパラメータ、デフォルトパラメータを区別）
C. returnによる値の返却と、returnが指定されていない場合 None がどのように使われるのかに関する説明
D. 開発中に一時的な関数本体として pass を使用

2.3.2 関数におけるスコープと変数の動作の理解

A. データスクリプト内でのローカル変数とグローバル変数の違いを区別
B. 名前のシャドーイングの説明と、関数内で変数名を再利用した場合の動作への影響の理解
C. グローバル変数は必要な場合にのみ使用し、ローカルスコープを優先的に利用すべき理由を理解

2.3.3 try-except ブロックを用いたエラー処理

A. データ処理中に発生し得る代表的な実行時エラーを特定（TypeError、ValueError、IndexError）
B. 分析スクリプトをより堅牢にするため、関数呼び出しを try-except ブロックで囲む
C. デバックや理解促進のため、意味のあるエラーメッセージを表示またはログに記録
D. FileNotFoundError、値の変換エラー、リストのインデックス範囲外エラーなど、ファイル処理時のクラッシュを防ぐために例外処理を使用

2.4 条件分岐とループによるプログラムフローを制御（3）

2.4.1 ブール値・論理演算と比較演算

A. 比較演算子（==、!=、<、>、>=、<=）を用いた式の計算
B. 論理演算子（and、or、not）を用いた複数の条件の組み合わせ
C. ブール式を用いたデータのフィルタリングや検証ロジックの構築

2.4.2 条件分岐とロジック制御

A. if、elif、else ブロックを用いた、データの値に基づく処理の分岐
B. 欠損データ、外れ値、無効な入力をチェックする条件文の作成
C. より複雑な意思決定のため、条件を入れ子に構成

2.4.3 繰り返し処理の記述

A. forループを用いた文字列、リスト、辞書、範囲などの反復処理
B. whileループを用いた条件が満たされる間、繰り返し処理
C. break、continue、else を用いたループの制御
D. 条件分岐とループを組み合わせた、データのクリーニング、集計、変換

2.5 モジュールとパッケージの活用（2）

2.5.1 Pythonのモジュールとパッケージのインポート

A. Import、from…import、およびエイリアスを用いた組み込みモジュールのインポート
B. math、random、statistics、collections、os、datetime などの標準ライブラリ関数をデータ処理に活用
C. csvモジュールを用いたCSVファイルの読み書き
D. 組み込みパッケージとサードパーティパッケージの違いを理解し、データ分析の目的に応じて適切に選択
E. 公式ドキュメント（docs.python.org）を参照し、モジュールの正しい使い方を理解

2.5.2 外部ライブラリをデータ処理ワークフローに利用

A. pipを用いた外部ライブラリ（例：numpy）のインストールとインポート
B. numpyを用いた配列操作と数値分析
C. 組み込みパッケージとサードパーティパッケージの違いの理解
D. 公式ドキュメント（numpy.org）を参照し、関数や機能を学習
E. ドキュメントを利用したエラーの解決、新しい関数の習得、未知の動作の理解

モジュール3：データの操作と基本的な分析（13）

3.1 ファイルを用いたデータの読み書き（2）

3.1.1 Pythonの組み込み機能を用いたテキストファイルの読み書き

A. open()、read()、readlines()、write()を用いたデータの保存・読み込み
B. with文を用いてファイルを安全に開き、自動で閉じる
C. osモジュールを用いてファイルパスを扱い、ファイルの存在を確認（os.path.exists()）
D. FileNotFoundErrorなどのファイル関連エラーを try-except ブロックで処理

3.1.2 csv モジュールによる CSV ファイルの読み書き

A. csv.reader() を用いて CSV ファイルを1行ずつ読み込む
B. csv.writer() を用いて表形式のデータを CSV 形式で書き出す
C. strip() や split(‘ , ‘) を用いた行の整形と解析
D. f-strings を用いた整った出力形式の結果の書き出し

3.2 分析のためのデータクリーニングおよび整形（4）

3.2.1 欠損値や無効なデータの処理・特定

A. 条件分岐やリスト内包表記を用いた、欠損値や null に近い値（例：None、空文字）の検出
B. 論理チェックを用いた欠損値の置き換えまたは削除
C. データ処理前に、無効な型、想定外のフォーマット、範囲外の値（例：負の年齢、空の名前フィールド）を if 文でチェック

3.2.2 重複データの除去と値の正規化

A. set()、辞書のキー、または内包表記によるフィルタリングを用いた重複の削除
B. リスト内包表記を用いて最小値・最大値正規化を手動で行う
C. enumerate() を用いた変換の適用（インデックスの追跡が必要な場合）

3.2.3 文字列のクリーニングおよびフォーマット

A. strip()、lower()、upper()、replace()、title() などの組み込み文字列メソッドを用いたテキストの正規化
B. 複数の文字列操作を連鎖した多段階のクリーニング（例：strip() lower() replace()）

3.2.4 分析／保存用途に応じたデータ変換・整形

A. int()、float()、str()、bool() を用いて一般的な型に変換
B. f-stringsを用いて数値を指定した精度でフォーマット
C. split() と join() を用いた文字列フィールドの操作
D. datetime.strptime() と strftime() を用いた日付・時刻の解析・整形・時系列処理

3.3 基本解析・計算処理（4）

3.3.1 Python組み込み関数による集計処理

A. len()、sum()、min()、max()、round() を用いたデータの要約と簡単な集計の計算
B. count() や辞書を用いた累積パターンによる値のカウント

3.3.2 組み込みライブラリによる記述統計量の計算

A. statistics.mean()、statistics.median()、statistics.stdev() を用いた平均、中央値、標準偏差の計算
B. math.sqrt()、math.ceil()、math.floor() などの math モジュールを用いた基本的な数値計算
C. collections.Counter() を用いたカテゴリデータ頻度の計算

3.3.3 NumPy による数値演算

A. numpy.array() を用いてリストを配列に変換
B. numpy.mean()、numpy.median()、numpy.std()、numpy.sum() などの numpy 関数を用いて配列ベースの統計を計算
C. numpy.arange() と numpy.linspace() を用いた数列の生成

3.3.4 条件付きメトリクスの計算（フィルタ・カテゴリ基準）

A. if 文やリスト内包表記を用いたデータのサブセットに対するメトリクス（例：平均やカウント）の計算
B. シンプルなカテゴリ（例：性別、地域、合格/不合格）ごとの値のグループ化と、辞書やループを用いたグループごとの要約の計算
C. and や or を用いて複数の条件を組み合わせた、より具体的なフィルタの作成（例：80点以上かつ特定のクラス）

3.4 基本的な探索的データ分析（EDA）の実施（3）

3.4.1 ソートとフィルタを用いたパターンや傾向の特定

A. sorted() や numpy.sort() を用いたデータの並べ替え
B. filter()、リスト内包表記、論理条件を用いたデータのフィルタリング

3.4.2 固有値と頻度の特定

A. set() や numpy.unique() を用いた異なる値の特定
B. Counter を用いたリスト内項目の出現頻度のカウント

3.4.3 簡単な相関チェックと外れ値の検出

A. numpy.corrcoef() を用いた数値リストや配列間の相関の計算
B. 単純なルール（例：閾値、標準偏差）や条件分岐を用いた外れ値の検出
C. numpy のブーリアンインデックスや条件分岐を用いた外れ値のフィルタリング
D. コードによる検索で見つかった基本的なパターンや異常値の解釈

モジュール4：洞察の共有とレポート作成（5）

4.1 データ可視化における基本原則の理解（2）

4.1.1 一般的な可視化の種類と目的の認識

A. 棒グラフ、折れ線グラフ、円グラフの特定および各用途の説明
B. 各可視化タイプの長所と短所について議論
C. データの種類や伝達目的に応じた適切な可視化の選択

4.1.2 単純なデータ可視化の解釈

A. 基本的な可視化が示す傾向、比較、割合の説明
B. 誤解を招く、または不明瞭な可視化の特定および改善方法の説明
C. 可視化が意図した洞察をサポートしているか、混乱を招いていないかの評価

4.2 データストーリーテリングの基礎を適用（1）

4.2.1 データから得られた洞察の構造化と伝達

A. データストーリーにおける基本構成（導入・洞察・結論）の説明
B. 根拠に基づく主要メッセージの提示
C. セクション間の流れを作ることを目的とするつなぎやサインポストの使用
D. 対象者の知識やニーズに応じたトーン、言語、深さの調整

4.3 明確で簡潔な分析レポートの作成（1）

4.3.1 分析結果の効果的な要約と整理

A. 主要なパターンや発見を短くまとめ、データ（例：平均、割合）で裏付ける
B. 問題、分析、洞察、推奨という論理的な構成の使用
C. 見出し、箇条書き、可視化を用いて明確さと読みやすさを向上

4.4 プレゼンテーションにおける洞察の効果的な伝達（1）

4.4.1 視覚・言語技術を用いたデータから得られる洞察の明確な伝達

A. アクセシブルで洗練されたデザイン原則（ラベル、タイトル、色、フォントサイズ）の使用
B. プレゼンテーションにおけるグラフや結果の明確な説明

C. 可視化や数値結果の証拠を用いた質問への応答

MQC プロファイル

PCED™ - Certified Entry-Level Data Analyst with Python の MQC（最小合格候補）は、データ分析の基礎原則、基本的なプログラミング、および分析的思考に関する基盤的な理解を有することが求められます。候補者はデータライフサイクル、データ型とデータ構造、データ収集方法、倫理的実践についての基礎知識を持っている必要があります。また、変数、制御構造、コレクション、関数、構造化データの分析に応用できることが期待されます。

さらに、math、random、csv、statistics、collections などの組み込みライブラリを用いて基本的な記述統計や単純な集計、探索的データ分析（EDA）の初歩を行える必要があります。また、numpy などのサードパーティライブラリを初歩的に使いこなすことも含まれます。結果を明確なストーリー構造と適切な可視化（棒グラフ、サマリーテーブルなど）で解釈・伝達するスキルも求められます。コミュニケーションは正確で対象者に応じて調整され、データに基づく根拠で裏付けられていなければなりません。

このプロフィールは、現代のデータ主導型環境におけるエントリーレベル職種に必要な、基本的な技術スキル・データリテラシー・コミュニケーション能力のバランスを示しています。

ブロック1：データとデータ分析の基本概念

配点比率：全体の22.5%（9問）

出題範囲：
MQCはデータとは何か、それがどのように情報に変換され、どのように分類されるか（定量、定性、構造化など）を理解している必要があります。一般的なデータソースと収集方法を説明でき、保存形式（CSV、JSON、データベースなど）の違いを説明できます。データライフサイクルの各段階を理解し、各段階での誤管理が結果に与える影響を説明できることが求められます。また、データサイエンス、データアナリティクス、データ分析の違いを区別し、4種類の分析タイプを理解し、倫理的および法的観点（例：プライバシー、同意、GDPR）を認識していることも求められます。

ブロック2：データ分析のための Python 基礎

配点比率：全体の32.5%（13問）

出題範囲：
MQCは変数、組み込みデータ型（int、float、str、bool）、およびリスト、セット、タプル、辞書などのコレクションを使用した簡単な Python スクリプトを作成・理解できることが求められます。関数を定義・呼び出すことができ、引数の扱いや基本的な例外処理（try-except）を行えます。また、if 文や for、while ループを用いてプログラムフローを制御し、条件を組み合わせてデータのフィルタリングや変換を行えることが必要です。組み込みライブラリ（math、random、csvなど）をインポートして利用することができ、公式ドキュメントを参照して情報を調べたり、問題を解決したりする力を持つことも求められます。また、numpy などサードパーティライブラリの基本的な使い方に触れていることも含まれます。

ブロック3：データの操作と基本的な分析

配点比率：全体の32.5%（13問）

出題範囲：
MQCは Python のファイル操作機能と csv モジュールを用いてデータを読み書きすることができます。欠損値や無効値を識別して処理し、数値や文字列の正規化を行い、分析のためのデータ型整形ができる必要があります。また、statistics や numpy を用いて平均・中央値・標準偏差などの基本統計量を計算でき、データをグループ化・フィルタリングして条件付き集計を行います。さらに、傾向、ユニーク値、頻度、基本的な相関を特定し、閾値や標準偏差を利用して外れ値を検出するスキルを有している必要があります。

ブロック4：インサイトの伝達とレポート作成

配点比率：全体の12.5%（5問）

出題範囲：
MQCはどのタイプがどの状況で適切かを認識し、棒グラフ、円グラフ、折れ線グラフなどの基本的なチャートが有効化、または誤解を与えるものになっていないか評価できます。また、対象者に合わせた構造化されたストーリーとしてインサイトを伝えることができます。レポートでは要約・見出し・箇条書きを用いて結果を整理し、視覚化や結果をわかりやすく口頭で説明できる能力を持つことが、求められます。

合格要件

PCED試験に合格するためには、すべての出題セクションにおける平均正答率が**75%**以上であることが求められます。

PCED-30-02 試験構成概要

PCED™試験は、Python を用いたデータ分析の主要な実務能力を評価するために設計された、単一選択式および複数選択式の全40問で構成されています。各問題は形式に関わらず最大1点として採点されます。試験終了後、総得点は正規化され、受験者の成績はパーセンテージで報告されます。試験は4つのブロックから成り、それぞれのブロックは問題数および内容の難易度に応じて比例的に配点が設定されています。.

ブロック番号	ブロック名	出題数	配点
1	データとデータ分析の基礎概念	9	22.5%
2	データ分析のためのPython基礎	13	32.5%
3	データの操作と基本的な分析	13	32.5%
4	インサイトの伝達とレポート作成	5	12.5%
		40	**100%**