

データサイエンス講座

Level 01-2

第2章：Python基礎の習得

第2章：Python基礎の習得

Pythonとは？

データサイエンスの定義

Pythonは、シンプルで読みやすい文法を特徴とする汎用プログラミング言語であり、データ処理、機械学習、可視化に適した豊富なライブラリ（例: NumPy、Pandas、scikit-learn、Matplotlib）を備えており、データサイエンスに欠かせないツール。他にも、R（統計解析に強い）、SQL（データベース操作に特化）、Julia（高速計算に適した言語）などがデータサイエンスの分野で使用されているが、Pythonはその汎用性と広範なコミュニティサポートで最も人気がある。

Windows版 Python 環境構築ステップ



公式サイトからPythonをダウンロード

[Python公式サイト](#)にアクセスし、「Download Python」をクリックしてインストーラーをダウンロード。



インストール実行

ダウンロードしたインストーラーを実行し、「Add Python to PATH」にチェックを入れてから「Install Now」を選択。



動作確認

コマンドプロンプトを開き、`python --version`と入力してインストールが成功したか確認。



必要なライブラリをインストール

コマンドプロンプトで `pip install numpy pandas matplotlib` などを実行してライブラリをインストール。

Mac版 Python 環境構築ステップ



Homebrewのインストール

ターミナルを開き、以下を実行してHomebrewをインストール。

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```



Pythonのインストール

Homebrewを使ってPythonをインストール。

```
brew install python
```



動作確認

ターミナルで `python3 --version` と入力し、インストールが成功したか確認。



必要なライブラリをインストール

ターミナルで `pip3 install numpy pandas matplotlib` を実行してライブラリをインストール。

エディタ

エディタ名	特徴	ダウンロードリンク
Visual Studio Code (VS Code)	無料で拡張性が高く、Jupyterと統合可能、軽量で多言語対応	https://code.visualstudio.com/
PyCharm Community Edition	Python専用、強力なデバッグ機能、Gitや仮想環境の管理が可能	https://www.jetbrains.com/pycharm/download/
Jupyter Notebook	対話型データ分析に最適、可視化対応、プロトタイピングに便利	https://jupyter.org/install
Thonny	初心者向け、視覚的なデバッグ、Pythonがすぐに利用可能	https://thonny.org/
Atom	オープンソースで高いカスタマイズ性、HydrogenでPython対応	https://atom.io/
Sublime Text	軽量で高速、パッケージで拡張可能、無料版利用可能	https://www.sublimetext.com/
Spyder	科学計算向け、Anacondaに統合、強力なデバッグ機能	https://www.spyder-ide.org/
IDLE	シンプルなインターフェース、Pythonに同梱、初心者最適	https://www.python.org/

Pythonサンプルコード：文字列表示

コード

```
# 画面に文字を出力する  
print("Hello, World!")
```

出力結果

Hello, World!

Pythonサンプルコード：変数とデータ型

2-02.py

```
# 変数に値を代入
name = "Alice"
age = 25
height = 1.68 # メートル

# 変数を使った出力
print(f"{name}さんは{age}歳で、身長は{height}mです。")
```

出力結果

Aliceさんは25歳で、身長は1.68mです。

Pythonサンプルコード：四則演算と数値操作

2-03.py

```
# 基本的な計算  
a = 10  
b = 3  
  
print("足し算:", a + b)  
print("引き算:", a - b)  
print("掛け算:", a * b)  
print("割り算:", a / b)  
print("余り:", a % b)  
print("べき乗:", a ** b)
```

出力結果

```
足し算: 13  
引き算: 7  
掛け算: 30  
割り算: 3.3333333333333335  
余り: 1  
べき乗: 1000
```


Pythonサンプルコード：条件分岐

2-04.py

```
# 条件分岐の例
score = 75

if score >= 90:
    print("評価: A")
elif score >= 70:
    print("評価: B")
else:
    print("評価: C")
```

出力結果

評価: B

Pythonサンプルコード：繰り返し処理（ループ）

2-05.py

```
# 1から5までの数字を出力  
for i in range(1, 6):  
    print(i)
```

出力結果

1
2
3
4
5

Pythonサンプルコード：繰り返し処理（ループ）

2-06.py

```
# カウントダウン  
count = 5  
while count > 0:  
    print(count)  
    count -= 1  
print("カウントダウン終了！")
```

出力結果

```
5  
4  
3  
2  
1  
カウントダウン終了！
```


Pythonサンプルコード：関数

2-07.py

```
# 関数の定義
def greet(name):
    return f"こんにちは、{name}さん！"

# 関数の呼び出し
message = greet("太郎")
print(message)
```

出力結果

こんにちは、太郎さん！

Pythonサンプルコード：リスト

2-08.py

```
# リストの操作  
fruits = ["リンゴ", "バナナ", "オレンジ"]  
fruits.append("ブドウ")  
print(fruits)  
print("最初の果物:", fruits[0])
```

出力結果

```
['リンゴ', 'バナナ', 'オレンジ', 'ブドウ']  
最初の果物: リンゴ
```


Pythonサンプルコード：辞書

2-09.py

```
# 辞書の操作
person = {"名前": "花子", "年齢": 30, "職業": "エンジニア"}
print(person)
print("名前:", person["名前"])
```

出力結果

```
{'名前': '花子', '年齢': 30, '職業': 'エンジニア'}
名前: 花子
```


Pythonサンプルコード：ファイル操作

2-10.py

```
# ファイルへの書き込み
with open("example.txt", "w") as file:
    file.write("これはテストファイルです。¥n")

# ファイルの読み込み
with open("example.txt", "r") as file:
    content = file.read()
    print(content)
```

出力結果

これはテストファイルです。

Pythonサンプルコード：ライブラリの活用

2-11.py

```
import math

radius = 5
area = math.pi * (radius ** 2)
print("半径5の円の面積:", area)
```

出力結果

半径5の円の面積: 78.53981633974483

Pythonサンプルコード：ユーザー入力

2-12.py

```
# ユーザー入力を受け取る  
name = input("あなたの名前は何ですか?: ")  
age = int(input("年齢を教えてください: "))  
  
print(f"{name}さん、{age}歳ですね!")
```

出力結果

```
あなたの名前は何ですか?: 太郎  
年齢を教えてください: 30  
太郎さん、30歳ですね!
```


Pythonサンプルコード：例外処理

2-13.py

```
try:
    num = int(input("数字を入力してください: "))
    print(f"入力された数字は {num} です。")
except ValueError:
    print("数字を入力してください!")
```

出力結果

数字を入力してください: abc
数字を入力してください!

Pythonサンプルコード：セット（集合型）

2-14.py

```
# セットの操作  
numbers = {1, 2, 3, 3, 4}  
numbers.add(5)  
numbers.remove(2)  
print(numbers)
```

出力結果

{1, 3, 4, 5}

Pythonサンプルコード：タプル（変更不可のリスト）

2-15.py

```
# タプルの操作  
dimensions = (1920, 1080)  
print(f"幅: {dimensions[0]}, 高さ: {dimensions[1]}")
```

出力結果

幅: 1920, 高さ: 1080

Pythonサンプルコード：真偽値と論理演算

2-16.py

```
# 真偽値と論理演算
x = 5
print(x > 3 and x < 10) # 両方の条件がTrue
print(x < 3 or x > 10) # どちらかの条件がTrue
print(not(x > 3)) # 条件を反転
```

出力結果

```
True
False
False
```


Pythonサンプルコード：スライス操作

2-17.py

```
# 文字列のスライス  
text = "Hello, Python!"  
print(text[0:5]) # 最初の5文字  
print(text[-7:]) # 後ろから7文字
```

出力結果

```
Hello  
Python!
```


Pythonサンプルコード：モジュールのインポート

2-18.py

```
import random

# ランダムな数値を生成
number = random.randint(1, 100)
print(f"1から100のランダムな数: {number}")
```

出力結果

1から100のランダムな数: 42

Pythonサンプルコード：リスト内包表記

2-19.py

```
# リスト内包表記  
squares = [x**2 for x in range(1, 6)]  
print(squares)
```

出力結果

```
[1, 4, 9, 16, 25]
```


Pythonサンプルコード：時刻と日付

2-20.py

```
from datetime import datetime

# 現在の日時を取得
now = datetime.now()
print(f"現在の日時: {now}")
```

出力結果

現在の日時: 2025-01-05 12:34:56.789123

Pythonサンプルコード：コマンドライン引数

2-21.py

```
import sys

# コマンドライン引数を取得
print("引数リスト:", sys.argv)
```

出力結果

```
$ python script.py arg1 arg2
引数リスト:['script.py', 'arg1', 'arg2']
```


Pythonサンプルコード：クラスとオブジェクトの基本例

2-22.py

```
# クラスの定義
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def bark(self):
        return f"{self.name}はワンワンと吠えます！"

# オブジェクトの生成
dog1 = Dog("ポチ", 3)
dog2 = Dog("タマ", 5)

# クラスのメソッドや属性を使用
print(f"{dog1.name}は{dog1.age}歳です。")
print(dog1.bark())
```

出力結果

ポチは3歳です。
ポチはワンワンと吠えます！

Pythonサンプルコード：クラスの継承

2-23.py

```
# 基本クラス（親クラス）
class Animal:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return "何か音を出します。"

# 子クラス（Animalを継承）
class Cat(Animal):
    def speak(self):
        return f"{self.name}はニャーニャーと鳴きます！"

# オブジェクトの生成
cat = Cat("ミケ")
print(cat.speak())
```

出力結果

ミケはニャーニャーと鳴きます！

Pythonサンプルコード：プロパティを使ったカプセル化

2-24.py

```
class Circle:
    def __init__(self, radius):
        self._radius = radius # プライベート属性として定義

    @property
    def radius(self):
        return self._radius

    @radius.setter
    def radius(self, value):
        if value > 0:
            self._radius = value
        else:
            raise ValueError("半径は正の数である必要があります。")

    def area(self):
        return 3.14 * (self._radius ** 2)

# オブジェクトの生成
circle = Circle(5)
print(f"半径: {circle.radius}")
print(f"面積: {circle.area()}")

# 半径の変更
circle.radius = 10
print(f"新しい半径: {circle.radius}")
print(f"新しい面積: {circle.area()}")
```

出力結果

半径: 5
面積: 78.5
新しい半径: 10
新しい面積: 314.0

Pythonサンプルコード：静的メソッドとクラスメソッド

2-25.py

```
class MathOperations:
    @staticmethod
    def add(a, b):
        return a + b

    @classmethod
    def multiply_by_two(cls, num):
        return num * 2

# 静的メソッドの呼び出し
print(MathOperations.add(3, 5))

# クラスメソッドの呼び出し
print(MathOperations.multiply_by_two(10))
```

出力結果

8
20

Pythonサンプルコード：特殊メソッド

2-26.py

```
class Book:
    def __init__(self, title, author):
        self.title = title
        self.author = author

    def __str__(self):
        return f"タイトル: {self.title}, 著者: {self.author}"

    def __repr__(self):
        return f"Book(title={self.title}, author={self.author})"

# オブジェクトの生成
book = Book("Python入門", "山田 太郎")

# __str__ と __repr__ の確認
print(book) # __str__ が呼び出される
print(repr(book)) # __repr__ が呼び出される
```

出力結果

タイトル: Python入門, 著者: 山田 太郎
Book(title=Python入門, author=山田 太郎)

演習：冒険ゲーム



学習のポイント

1. クラスとオブジェクト: プレイヤーや敵のデータ構造をクラスで管理。
2. 条件分岐とループ: プレイヤーの選択やゲームの進行にif文やwhileループを使用。
3. ランダム要素: randomモジュールを使ってイベントやダメージをランダム化。
4. リスト操作: プレイヤーのアイテムをリストで管理。
5. 入力と出力: input()でプレイヤーの選択を取得し、print()で結果を表示。
6. モジュールの活用: datetimeを使ってプレイ開始時間を記録するなどの応用が可能。



動作の流れ

1. プレイヤーが名前を入力し、ゲームが始まります。
2. 洞窟を探索し、宝箱や敵、回復の泉などランダムイベントに遭遇します。
3. イベントごとに選択肢を与え、戦闘や探索を続けます。
4. プレイヤーが倒れるか、探索を終了するまで続きます。

2-27.py

```
import random

from datetime import datetime

# プレイヤーキャラクターのクラス
class Player:

    def __init__(self, name, health=100):

        self.name = name

        self.health = health

        self.items = []

    def take_damage(self, damage):

        self.health -= damage

        print(f"{self.name}は{damage}のダメージを受けました。残り体力: {self.health}")

        . . .
```


演習：AI性格診断ゲーム



学習のポイント

1. インタラクティブ性: ユーザーが選択肢を選ぶことでゲームに参加。
2. リストと辞書の活用: 質問データを構造的に保存。
3. 条件分岐: ユーザーの選択に応じた性格診断結果を表示。
4. ユーモアとカスタマイズ性: 性格結果にユーモアを加えることで楽しい体験に。



動作の流れ

1. プレイヤーが名前を入力し、AI性格診断を開始。
2. 各質問に選択肢を選び、性格タイプをポイント化。
3. 最終結果として性格診断とコメントを表示。

2-28.py

```
import random

# 質問データ
questions = [
    {"question": "朝起きたらまず何をする?",
     "options": ["コーヒーを飲む", "SNSをチェック", "布団でゴロゴロ", "エクササイズ"],
     "traits": ["冷静", "社交的", "怠け者", "アクティブ"]},
    {"question": "休日の過ごし方は?",
     "options": ["友達と出かける", "家で映画を観る", "新しい趣味に挑戦", "ひたすら寝る"],
     "traits": ["社交的", "内向的", "好奇心旺盛", "怠け者"]},
    {"question": "好きな動物を1つ選ぶなら?",
     "options": ["猫", "犬", "鳥", "イルカ"],
     "traits": []},
    ...
]
```


演習：エコゲーム



学習のポイント

1. 環境意識の向上: 選択肢を通して環境保護について学べます。
2. リストと辞書の活用: 各シナリオのデータを効率的に管理。
3. 条件分岐とスコア計算: 選択肢に応じてポイントを加減算。
4. ランダム化: 毎回異なる順序でシナリオを出題するため、何度でも楽しめます。



動作の流れ

1. プレイヤーが名前を入力し、ゲームがスタート。
2. 各シナリオに対して環境保護に関する選択を行い、ポイントを獲得。
3. 最終的なスコアと評価が表示され、環境に対する意識を促します。

2-29.py

```
import random

# 選択肢データ
scenarios = [
    {
        "scenario": "あなたはゴミを分別せずに捨てようとしています。",
        "options": ["そのまま捨てる", "ゴミを分別して捨てる", "ゴミを拾ってリサイクルへ運ぶ"],
        "points": [-5, 5, 10]
    },
    {
        "scenario": "コンビニで買い物をします。レジ袋はどうしますか?",
        "options": ["レジ袋を使う", "マイバッグを使う", "何も買わない"],
        "points": [-3, 7, 2]
    },
    . . .
```


DS Lv.1-2

お疲れ様でした

info@mjin.space

mjin.space