6．オブジェクト指向データベース

1. 基礎知識

1.1. Pythonのデータ型

Pythonでのデータ型の確認は，type関数を使用して行うことができる．Pythonは動的型付け言語であり，変数の型は実行時に変更可能である．以下のプログラムは，数値型（int）と文字列型（str）の基本的な使用例を示している．

a = 100

print(type(a))

b = "Hello!"

print(type(b))

1.2. リスト（list）

リストはデータの配列構造である．以下のプログラムでは，連続する4つの整数を含むリストの生成例を示している．Pythonのリストは可変長で，異なる型の要素も格納可能である．

v = [100, 101, 102, 103]

print(v)

数列の生成には range 関数が使用できる．range(開始値, 終了値, 増分) の形式で，等差数列を効率的に生成する．

u = range(1, 11, 2)

print(u)

1.3. 辞書（dictionary）

辞書は，キーと値のペアを管理するデータ構造である．ハッシュテーブルを使用しており，キーによる高速なデータアクセスが可能である．以下は辞書の基本的な使用例である．

d = {'x': 100, 'y': 200}

print(d)

辞書は動的に要素を追加できる．

d = {'x': 100, 'y': 200}

d['z'] = 300

print(d)

keys() メソッドを使用して，すべてのキーを取得できる．

print(d.keys())

1.4. デコレータ（@記号）

Pythonのデコレータは，関数やクラスの動作を変更・拡張するための機能である．@記号を使用して関数やクラスの定義の前に記述する．デコレータを使用することで，コードの再利用性と可読性が向上する．

2. データ処理基盤

2.1. pandas

pandasは，Pythonにおける強力なデータ解析ライブラリである．表形式データの操作や統計解析に特化した機能を提供する．

2.1.1. PandasのSeries

Seriesは，インデックス付きの1次元配列構造である．NumPyをベースとした高速な演算と，統計分析機能を提供する．

import pandas as pd

s = pd.Series([100, 101, 102, 103])

print(s)

s.describe()

2.1.2. Pandasのデータフレーム（DataFrame）

DataFrameは，2次元のテーブルデータを扱うための主要なデータ構造である．以下は，基本的な使用例である．

import pandas as pd

x = pd.DataFrame()

x['id'] = [1, 2]

x['name'] = ['apple', 'orange']

x['price'] = [80, 150]

print(x)

新しい列の追加も容易に行える．

x['color'] = ['red', 'orange']

print(x)

DataFrameの各列はSeries型として扱われ，高度な統計分析機能を利用できる．

import pandas as pd

x = pd.DataFrame()

x['id'] = [1, 2]

x['name'] = ['apple', 'orange']

x['price'] = [80, 150]

print(x.keys())

print(x['price'])

print(type(x['price']))

print(x['price'].describe())

条件に基づくデータの抽出には，queryメソッドを使用する．

import numexpr

x.query("price > 100")

2.2. CSVファイルの操作

2.2.1. CSVファイルの読み込み

Windowsシステムでのファイルパスは，バックスラッシュをエスケープする必要がある．ヘッダー行がある場合は，header=0 を指定する．

import pandas as pd

iris = pd.read\_csv('C:\\iris.csv', header=0)

print(iris)

2.2.2. CSVファイルの書き出し

DataFrameをCSVファイルとして保存する．

x.to\_csv('x.csv')

3. データベース

3.1. オブジェクト指向データベース

オブジェクト指向データベースは，プログラムで使用しているオブジェクトを直接永続化できる特徴を持つ．

3.2. ZODB（Zope Object Database）

ZODBは，Pythonオブジェクトの永続化に特化したデータベースシステムである．オブジェクトリレーショナルマッピングを必要とせず，トランザクション機能を備えている．

3.3. ZODBの基本操作

3.3.1. データベースの接続

以下のコードは，ZODBデータベースへの接続と初期化を行う．指定したファイルが存在しない場合，自動的に作成される．

import ZODB

import ZODB.FileStorage

storage = ZODB.FileStorage.FileStorage('hoge.fs')

db = ZODB.DB(storage)

conn = db.open()

3.3.2. ルートオブジェクトの操作

ZODBのルートオブジェクトは辞書として実装されており，キーと値のペアとしてデータを格納できる．

import ZODB

import ZODB.FileStorage

storage = ZODB.FileStorage.FileStorage('hoge.fs')

db = ZODB.DB(storage)

conn = db.open()

root = conn.root()

print(root.keys())

root['rose'] = 'red'

root['banana'] = 'yellow'

print(root.keys())

print(root)

4. 演習

4.1. Pandasを使用したデータ分析

以下の手順で，CSVファイルのデータをPandasで分析する．

1. CSVファイルの読み込み

import pandas as pd

iris = pd.read\_csv('C:\\iris.csv', header=0)

print(iris)

2. 基本的な統計分析

print(iris.describe())

print(type(iris))

print(iris.keys())

print(iris['sepal\_length'])

print(iris['sepal\_width'])

4.2. ZODBでのデータ永続化

以下の手順で，データをZODBに保存する．

1. データベースへの接続と初期状態の確認

import ZODB

import ZODB.FileStorage

storage = ZODB.FileStorage.FileStorage('hoge.fs')

db = ZODB.DB(storage)

conn = db.open()

root = conn.root()

print(root.keys())

root['rose'] = 'red'

root['banana'] = 'yellow'

print(root.keys())

print(root)

2. Irisデータセットの保存

root['iris\_data'] = iris

print(root.keys())

print(root)

print(root['iris\_data'])

3. 追加データの保存

root['v'] = [100, 101, 102, 103]

root['d'] = {'x':100, 'y':200}

print(root.keys())

print(root)