Pythonワークショップ(入門編)

M.Kohana

15 Feb 2023

Chuo Univ

Contents

導入

Python とは

プログラマの作業性とコードの信頼性を高めることを重視して
デザインされた高水準汎用プログラミング言語

- 動的型付け
- ガベージコレクション
- オブジェクト指向
- マルチパラダイム
- インデントを使ったブロック表現

Colab とは

Colab (Colaboratory) はブラウザ上で Python を
記述、実行できるプラットフォーム

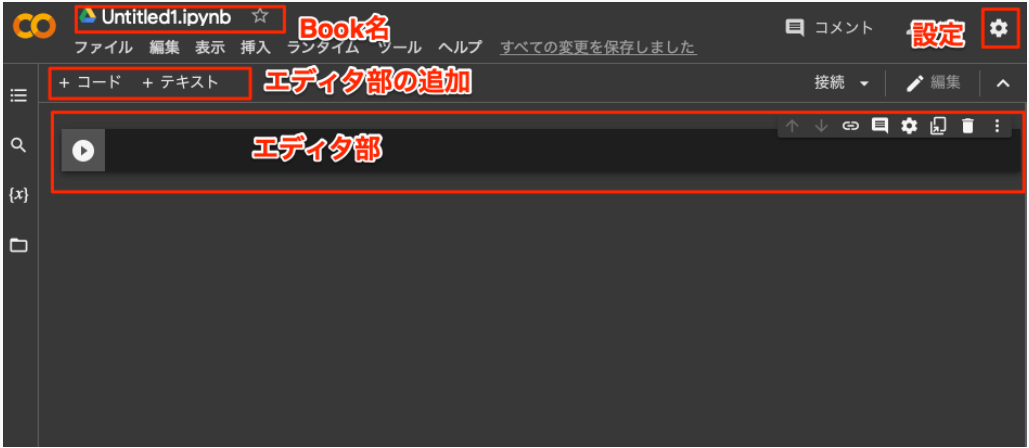
- 環境構築が不要
- GPU が利用可能
- 他のユーザと共有が容易
- Google ドライブ等からのデータインポートが可能

Colab の画面

新規作成

Colabの画面

メイン画面



Pythonの変数とデータ型

プログラムの構成要素

値 データ
- 変数 変更可能なデータ
- 即値 値そのもの

式 計算をして値を返す

制御 分岐やループ

データの画面出力

print 関数

print(式 or 値)

```
# 即値の出力
print(10)
print("Hello World")

# 変数の出力
x = 10
print(x)
```

データの画面出力

```
# 式の出力
print(10 + 20)

h = 4
w = 7
print(w * h)
```

基本データ型

基本データ型	
str	文字列
int	整数
float	浮動小数点数
bool	真偽値

コレクション型	
list	リスト
tuple	タプル
set	集合
dictionary	辞書

文字列

シングルクォーテーション (’)、ダブルクォーテーション (”)、またはそれを 3 つ繋げたもの (”””) で囲むと文字列として処理される。

ダブルクォーテーションや改行などの特殊文字を文字列として扱いたい場合には直前にバックスラッシュを付ける。

文字列

```
msg1 = "Hello World"
print(msg1)

msg2 = "10 + 20"
print(msg2)

msg3 = "\"Hello \n Python\""
print(msg3)
```

変数

データを格納しておくための場所。
名前を決めて初期値を設定する。

名前は半角英字かアンダースコアで始める。使える文字は半角英数字とアンダースコア。

変数

変数の初期化や値の代入は代入演算子 (=) を使用する。

新たに代入する値は以前の値の型と異なっても構わない。

```
val = 10
print(val)
val = 3.14
print(val)
val = "Message"
print(val)
```

演算子

- 基本演算子: `+`, `-`, `*`, `/`, `%`
- 切り捨て除算: `//`
- べき乗: `**`

演算子

```
# 基本演算子の例
x = 10
y = 3
print(x + y) # 加算
print(x - y) # 減算
print(x * y) # 乗算
print(x / y) # 除算
print(x % y) # 剰余
```

演算子

Python の除算演算子は整数同士の除算であっても浮動小数点数で結果を出す。

整数として計算したい場合は切り捨て除算と剰余を使用する。

```
x = 10
y = 4
print(x // y)
print(x % y)
```

出力の桁数合わせ

出力するデータの桁数を調整したい場合はプレースホルダを使った出力フォーマットを使用する。

プレースホルダの種類

%s 文字列

%d 整数

%f 浮動小数点数

出力の桁数合わせ

桁数を調整する場合には%と英字の間に出力したい桁数を入力する。

```
name = "kohana"
age = 38
height = 170.2

print("name : %10s, age: %5d, height: %5.3f" %(name, age, height))
```

コレクション

1つの変数に対して複数のデータを割り当てるための方法。
いずれも異なるデータをまとめて扱うことができる。

- リスト
- タプル
- 集合
- 辞書

リスト

順序付きのデータ構造。1つの変数で複数の値を格納することができる。
また、要素の追加や削除をすることも可能。

リストを作成するには[]で値を囲む。

```
arry = [0, 2, 4, 6, 8]
print(arry)
```

リスト

個々のデータにアクセスするには添字を使用する。

```
arry = [0, 2, 4, 6, 8]
print(arry[2])
arry[4] = 20
print(arry[4])
```

添字番号は 0 から始まることに注意する。
要素数 5 のリストの添字は 0-4 となる。

リスト

要素の追加

- append
- extend
- insert

リスト

append

リストの末尾にデータを追加する。

```
arry = [10, 20, 30]
print(arry)
arry.append(40)
print(arry)
```

リスト

extend

リストの末尾に別のリストを追加する。

```
arry = [10, 20, 30]
print(arry)
arry2 = [40, 50, 60]
arry.extend(arry2)
print(arry)
```

リスト

insert

指定した位置にデータを追加する。

```
arry = [10, 20, 30]
print(arry)
# 要素の１番目にデータを追加する
# 元々１番目にあったデータは２番目に移る
arry.insert(1, 40)
print(arry)
```

リスト

要素の削除

- clear
- pop
- remove

リスト

clear

リスト内の全てのデータを削除する。

```
arry = [10, 20, 30]
print(arry)
arry.clear()
print(arry)
```

リスト

pop

指定した位置のデータを取り出し、リストから削除する。

```
arry = [10, 20, 30, 40]
arry.pop(1)
print(arry)
x = arry.pop(2)
print(x)
print(arry)
```

リスト

remove

指定したデータを削除する。

```
arry = [10, 20, 30, 10]
print(arry)
arry.remove(20)
print(arry)
arry.remove(10)
print(arry)
```

リスト

len

要素の数を取得する。

```
arry = [10, 20, 30]
print(len(arry))
arry.append(40)
print(len(arry))
```

タプル

基本的にはリストと同じ挙動をする。
ただし、タプルはデータの追加・削除ができない。
タプルは () で囲む。

```
tpl = (10, 20, 30)
print(tpl)
```

集合

順序なし、重複なしのデータ型。
更新可能なオブジェクト(リストや辞書)は登録不可能。

```
grp = {1, 2, 3, 4, 4}
print(grp)

# 空集合を作る場合
grp = set()
print(grp)
```

集合の操作

- 確認: in
- 追加: add
- 削除: remove

```
grp = {1, 2, 3, 4, 5}
print(3 in grp)
grp.add(6)
grp.remove(4)
print(grp)
print(4 in grp)
```

辞書型

key と value を使ってデータを管理する型。
リストのような構造を持っているが、添字の番号の代わりに文字列を使用する。
辞書型は{}で作成する。

辞書型

```
dic = {
    "apple": 100,
    "orange": 200,
    "grape": 150
}
print(dic)
print(dic["orange"])
```

辞書型

要素の削除

- clear: リストの挙動と同じ
- pop: リストの挙動と同じ
- del

del は指定したキーの値を削除する。

```
dic = {"apple": 100, "orange": 200, "grape": 150}
print(dic)
dic dic["orange"]
print(dic)
```

辞書型

要素の更新と追加

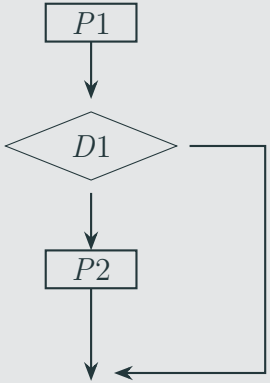
```
dic = {"apple": 100, "orange": 200, "grape": 150}
dic["orange"] = 120

# 存在しないキーに対して値を設定すれば要素を追加できる
dic["melon"] = 1000
print(dic)
```

制御構造

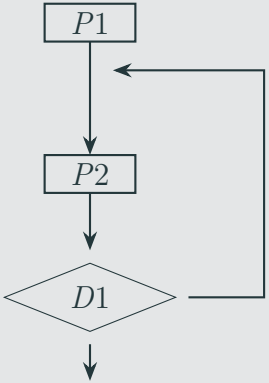
制御構造

逐次

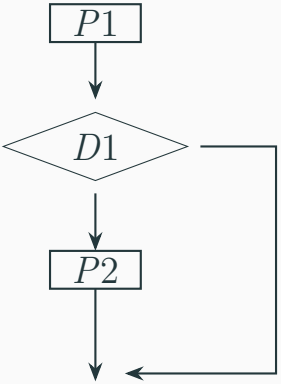
分岐



繰返し



条件分岐



条件式(D1)の判定結果に応じて次の処理を決定する方法。
図の例ではD1の結果に応じてP2を処理するかどうかを決定する。

条件分岐

条件式

条件式では、条件を満たす場合はTrue、そうでなければFalseが結果となる。

$A == B$: AとBが等しければTrue
$A > B$: AがBより大きければTrue
$A >= B$: AがB以上ならばTrue
$A != B$: AとBが等しくなければTrue
$AinB$: AがBに含まれればTrue
$AnotinB$ AがBに含まれなければTrue

条件分岐

if-elif-else 文

```
if 条件1:
    処理1
elif 条件2:
    処理2
else:
    処理3
```

処理の範囲はインデント (字下げ) で決定する。
if ブロックは必ず必要。elif ブロックは 0 個以上。else ブロックは 0 個か 1 個に限定される。

条件分岐

```
score = int(input("Score?:"))

if score > 80:
    print("A")
elif score > 60:
    print("B")
else:
    print("C")

print("done")
```

条件分岐

複数の条件を組み合わせるには and/or を使う。
また、括弧によって条件の処理順を調整できる。

```
year = int(input("year?:"))
if (year % 400 == 0 or (year % 100 != 0 and year % 4 == 0)):
    print("leap year")
```

繰り返し

条件を満たすまで同じ処理を繰り返す。
条件式やブロック表現については if 文と同じ。

- while
- for

while ループ

条件式が True の間は処理を繰り返す。

while

while 条件式:
　処理内容

```
i = 0
while i < 10:
    print(i)
    i += 1
print("finishi")
```

forループ

リストなどのコレクション型などを走査するときに使われる。
コレクションの中身を順番に処理していく。

```
arry = [0, 1, 2, 3, 4]
for i in arry:
    print(i)
print("finish")
```

arryの要素を1つずつiに取り出して処理をしていく。
iは変数なので名前は自由に付けられる。

forループ

要素の値と添字番号を一緒に取り出したい場合にはenumerateを使う。

```
arry = [0, 1, 2, 3, 4]
for i, v in enumerate(arry):
    print("%d: %d" %(i, v))
```

for ループ

辞書型を for ループで走査する場合には items を使う。

```
dic = {"apple": 100, "orange": 200, "grape": 150}
for k, v in dic.items():
    print(k, v)
```

items を使用しない場合は key のみ取り出される。

関数

関数

関数

ある目的のための一連の処理をまとめたもの。
関数にしておくことで繰り返し呼び出すことができるので、同じ処理を何度も記述する必要がなくなる。

```
def 関数名(引数):
    処理
```

関数

関数は 0 個以上の引数をとり、0 個または 1 個の返値を返す。

引数　関数に与えるデータ
返値　関数が結果として返すデータ

関数は定義と呼び出しの両方で成立する。定義されていない関数は呼び出しできない。また、呼び出しのない関数は定義されていたとしても動かない。

関数

引数0、返値0の関数の例

```
def showPrompt():
    print("--->")

showPrompt()
showPrompt()
```

関数

引数ありの関数の例

```
def showProfile(name, age):
    print("Name: %s, Age: %d" %(name, age))

showProfile("kohana", 38)
```

関数

返値のある関数の例

```
def say_hello(name):
    return "Hello " + name

print(say_hello("kohana"))
```

結果を返すには return を使う。

データ分析

データ分析

- 相関分析
- 推定と検定
- コサイン類似度

データ分析ツール

代表的なデータ分析ライブラリ

NumPy 行列計算などの数値計算をするためのライブラリ

Pandas 表形式のデータフレームを扱うライブラリ

SciPy 統計分析などの計算をするためのライブラリ

scikit-learn 機械学習ライブラリ

ライブラリの使用

```
# 数学関数郡のインポート
import math

# 数学関数の平方根を使用
a = math.sqrt(7)
print(a)
```

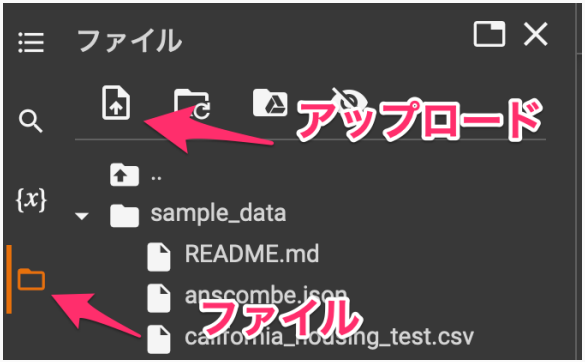
Pandasのを使った相関係数

```
# pandas をインポートして pd という名前で使う
import pandas as pd

# 相関をとりたいデータ
sc1 = [12, 24, 18, 36, 46, 52, 17]
sc2 = [73, 82, 83, 62, 50, 48, 70]

# データを pandas のデータフレームに変換
sc1_ser = pd.Series(sc1)
sc2_ser = pd.Series(sc2)

# 相関係数の計算
res = sc1_ser.corr(sc2_ser)
print(res)
```

Pandas を使った相関係数

```
# pandas のデータフレームを csv ファイルから作成
clf_house = pd.read_csv("ファイル名")

# 列を指定してデータフレームを作成
clf_house = pd.read_csv("ファイル名", usecols=["列名1", "列
名2", ...])

# csv ファイルを元に相関係数の表を作成
res = clf_house.corr()
print(res)
```

Goolge Colaboratoryでのファイルの扱い



ファイルを右クリックして”パスをコピー”をすればファイルの場所がコピーできる。

t検定

標本の平均に差があるかどうかを確認する統計的手法。

対応がある2標本t検定　同一の対象から抽出した、対となる標本の比較に使う。(投薬前後や勉強前後などの比較)

対応のない2標本t検定　異なる対象から抽出した2つの標本の比較に使う。(A社とB社の比較や国別などの比較)

t検定

```
scipy ライブラリから stats のみインポートする
from scipy import stats

# R のデフォルトデータセットである sleep のデータを使用する。
# 2 種類の睡眠薬を 10 人の被験者にそれぞれ投与した際の睡眠時間の増減のデータ
a = [0.7, -1.6, -0.2, -1.2, -0.1, 3.4, 3.7, 0.8, 0.0, 2.0]
b = [1.9, 0.8, 1.1, 0.1, -0.1, 4.4, 5.5, 1.6, 4.6, 3.4]

res = stats.ttest_rel(a, b)
```

クラスタ分析

様々な性質を持った集団の中から、似た性質のものを集めてクラスターを作る手法。

K-means

教師なし学習のクラスタリングアルゴリズムの1つ。

K-means

```
import numpy as np
import pandas as pd

# K-means ライブラリのインポート
from sklearn.cluster import KMeans
# グラフ描画ライブラリのインポート
from matplotlib import pyplot as plt
```

K-means

```
clf = pd.read_csv("california_housing_train.csv", usecols=["longitude", "latitude"])
# 読み込んだデータを numpy の配列形式に変換
clf_arry = np.array([clf["longitude"].tolist(), clf["latitude"].tolist()], np.float32)

# 配列を転置
clf_arry = clf_arry.T

# 分割するクラスタ数の指定とクラスタリングの実行
pred = KMeans(n_clusters=4)
pred = pred.fit(clf_arry)
```

K-means

```
# クラスタリング結果の描画
plt.scatter(clf_arry[:,0], clf_arry[:, 1], c=pred.labels_)

# クラスタの重心の描画
# s はマーカーサイズ、marker は描画記号、c は色
plt.scatter(pred.cluster_centers_[:,0],
            pred.cluster_centers_[:,1],
            s=250, marker='*', c='red')
# 画面出力
plt.show()
```

csvを作成する

```
# csv操作ライブラリをインポート
import csv

header = ["name", "age"]
data = [ ["kohana", 38], ["masaki", 36],
         ["chuo", 34], ["taro", 40] ]

path = "/content/name.csv"
with open(path, mode='w') as f:
    writer = csv.writer(f)
    writer.writerow(header)
    writer.writerows(data)
```

まとめ

1. 導入
2. Python の変数とデータ型
3. 制御構造
4. 関数
5. データ分析

Python の参考書

- 独習 Python, 翔泳社
- Python で学ぶあたらしい統計学の教科書, 翔泳社
- Python によるあたらしいデータ分析の教科書, 翔泳社