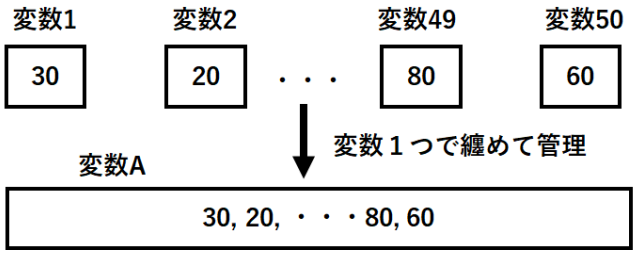
2021年度 東京成徳大学 特別講座 資料
東京工科大学コンピュータサイエンス学部
福西広晃

第02回 Python基礎（データの扱い方）

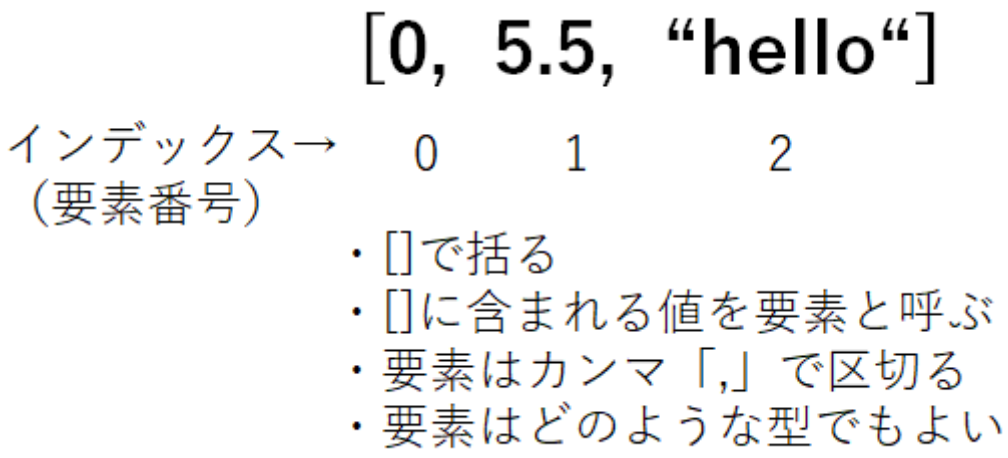
- 前回は、個々の数値や文字を扱ったが、プログラミングでは複数の値から成るデータを扱うことも多い
- 複数の値を1つの変数で管理しながら、個々の値（ 要素 と呼ぶ）を取り出したり、場合によっては変更したりできると便利なことがある。そのためにPythonでは **リスト** 、 **タプル** 、 **辞書** 、 **集合** のデータ型が提供されている
- 今回は、特に重要な、 **リスト** を学習する

【例】50人の学生のテスト結果をプログラムで処理したいときに、個々の点数を変数に代入していては変数の管理が大変になる。50人分の点数を纏めた１つの変数で管理するほうが扱い易い



リスト

- **リスト** は、数値や文字列などの基本データ型と並んでPythonでは高頻度で使われるデータ型である
- **リスト** は、複数の値をまとめて扱う場合に使用する
- **リスト** は全体を [ ] で囲み、各要素は , で区切る
- **リスト** はどのような型でも格納することができ、リストの中にリストを格納することも可能である
- **リスト** では **インデックス** (要素番号)を用いて、要素の追加や入れ替えなどが可能である



リストの操作の基礎

リストを変数に代入

In [ ]:
```
# 5人のテスト結果が90点、85点、75点、95点、80点であったとする
listA = [90, 85, 75, 95, 80]
print(listA)
```

要素数をカウント

- len関数 を用いる

  len(リスト)

In [ ]:
```
print("要素数:", len(listA))
```

要素の最大値、最小値を抽出

- 最大値の抽出には、 max関数 を用いる

  max(リスト)

- 最小値の抽出には、 min関数 を用いる

  min(リスト)

In [ ]:
```
print('最大値: ', max(listA))
print('最小値: ', min(listA))
```

練習

listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]がある。

In [ ]:
```
listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]
```

(1) listXの要素数を出力してみよう

In [ ]:
```

```

(2) listXの最小値を出力してみよう

In [ ]:
```

```

解答例

In [ ]:
```
#(1)
print("要素数:", len(listX))

#(2)
print('最小値: ', min(listX))
```

要素の取り出し

- インデックス（要素番号）を指定
  リスト[インデックス]

In [ ]:
```
print(listA[0]) # インデックス0を取り出して出力
print(listA[1]) # インデックス1を取り出して出力
print(listA[2]) # インデックス2を取り出して出力
print(listA[3]) # インデックス3を取り出して出力
print(listA[4]) # インデックス4を取り出して出力
```

要素の置換

- インデックス(要素番号)で置換前の要素を指定し、置換したい値を代入

In [ ]:
```
# 変更前をリストを表示
print("置換前:", listA)

# インデックス(要素番号)0を置換
#  （最初の学生の点数は70点だった）
listA[0] = 70

# 変更後をリストを表示
print("置換後:", listA)
```

練習

listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]がある。

In [ ]:
```
listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]
```

(1) listXの右から4番目の要素をを出力してみよう

In [ ]:
```

```

(2) listXの右から6番目の要素を7.7に変更した後、listXを出力してみよう

In [ ]:
```

```

解答例

In [ ]:
```
#(1)
print(listX[3])

#(2)
listX[5] = 7.7
print(listX)
```

複数のリストの結合

- リストA = リストB + リストC
- 最初のテストの受験者5人の結果は、90点、85点、75点、95点、80点であったとする。その後、新たに4人がテストを受験して、70点, 80点, 75点, 90点であった。さらに、1人がテストを受験して85点であった。

In [ ]:
```
listA = [90, 85, 75, 95, 80]
listB = [70, 80, 75, 90]
listC = [85]

# listAとlistBを結合させてlistDを作成
listD = listA + listB

# リストを出力
print("1回目の結合：", listD)

# listDに対してlistCを結合させてlistDを拡張
listD = listD + listC     # listD += listC でも可

# リストを出力
print("2回目の結合：", listD)
```

練習

listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]がある。listXに対してlistY=[44.8]を結合してlistXを拡張した後、listXを表示してみよう

In [ ]:
```
listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]
listY = [44.8]
```

In [ ]:
```

```

解答例

In [ ]:
```
listX = listX + listY
print(listX)
```

要素の並び替え

- sorted関数 を用いる
  - 昇順に並び替え（ただし要素は数値のみか、文字列のみ）

    sorted(リスト)
  - 降順に並び替え

    sorted(リスト,reverse=True)`

要素が数値の場合

In [ ]:
```
listD = [90, 85, 75, 95, 80, 70, 80, 75, 90, 85]

# listDを昇順に並び替えてlistDに代入
listE = sorted(listD)
print("昇順に並び替え：", listE)

# listDを降順に並び替えてlistFに代入
listF = sorted(listD, reverse=True)
print("降順に並び替え：", listF)
```

要素が文字列の場合(アルファベット順)

In [ ]:
```
listG = ["dog", "cat", "tiger", "bird", "horse"]
```

```
listH = sorted(listG)
print("昇順に並び替え：", listH)

listI = sorted(listG, reverse=True)
print("降順に並び替え：", listI)
```

練習

listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]がある。listXの要素を降順に並び替えたlistYを作成して、listYを出力しよう

In [ ]:
```
listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]
```

In [ ]:
```

```

解答例

In [ ]:
```
listY = sorted(listX,reverse=True)
print(listY)
```

反復（for文）による要素の取得

```
for 要素を代入する変数 in リスト:
    実行する処理
```

要素を1個ずつ変数に代入して繰り返す

要素の平均値を計算

10人のテストの点数の平均値

In [ ]:
```
# 10人のテストの点数が格納されたlistD
listD = [90, 85, 75, 95, 80, 70, 80, 75, 90, 85]

sum = 0                 # 合計値の初期値を0とする
for ele in listD:       # 要素を１個ずつ取り出して変数eleに代入
    sum = sum + ele     # 要素を足す　（sum += eleも可）

# 平均値 = 合計/要素数
ave = sum / len(listD)  # 平均値をaveという変数に代入

print("合計値：", sum)
print("平均値：", ave)
```

練習

listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]がある。listXの合計と平均値を出力してみよう

In [ ]:
```
listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]
```

In [ ]:
```

```

解答例

In [ ]:
```
sum = 0                 # 合計値の初期値を0とする
for ele in listX:       # 要素を１個ずつ取り出して変数eleに代入
    sum = sum + ele     # 要素を足す　（sum += eleも可）

# 平均値 = 合計/要素数
ave = sum / len(listX)  # 平均値をaveという変数に代入

print("合計値：", sum)
print("平均値：", ave)
```

特定の値の要素の削除

- 使用メソッド
  リスト.remove(削除する要素)`
- リストの中に該当する要素が見つからなかった場合はエラー(ValueError)が発生する
- 削除される要素は「最初に該当したもの」のため、同じ値が2回以上出てくると最初に出る要素だけ削除される

In [ ]:
```
# 削除前のリストを表示
listA = [90, 85, 75, 95, 80] # 5人の点数
print("削除前:", listA)

# 85点を削除
listA.remove(85)

# 削除後のリストを表示
print("削除後:", listA)
```

同じ値の2つの要素を削除する場合

In [ ]:
```
# 削除前のリストを表示
listC = [90, 85, 75, 95, 85] # 5人の点数
print("削除前:", listC)

# 1つ目の85点を削除
listC.remove(85)

# 削除後のリストを表示
print("1回目削除後:", listC)

# 2つ目の85点を削除
listC.remove(85)

# 削除後のリストを表示
print("2回目削除後:", listC)
```

インデックスを用いた要素の削除

- 使用メソッド

  リスト.pop(削除する要素のインデックス)

In [ ]:
```
# 削除前のリストを表示
listA = [90, 85, 75, 95, 80] # 5人の点数
print("削除前:", listA)
```

```
# インデックス(要素番号)0番目を削除
listA.pop(0)

# 削除後のリストを表示
print("削除後:", listA)
```

練習

listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]がある。右から7番目の要素を削除した後、listXを出力してみよう

In [ ]:
```
listX = [112.3, 332.9, 573.8, 552.8, 123.8, 379.1, 332.6, 768.2, 249.8]
```

In [ ]:
```

```

解答例

In [ ]:
```
listX.pop(6)
print(listX)
```

練習

5人の身長を測った結果、170cm、175cm、160cm、155cm、165cmであったとする

(1) 5人の身長の数値を要素とするリストをhigh1の変数名で作成して、要素数を表示しよう

In [ ]:
```

```

(2) 前から4番目の人の身長に間違いがあったため150cmに変更して、リストを表示しよう

In [ ]:
```

```

(3) 新たな5人の身長が180cm、155cm、160cm、165cm、170cmとする。 新たな5人の身長を要素とするリストをhigh2の変数名で作成してみよう。 さらに、high1とhigh2を結合したリストをhigh3の変数名で作成し、表示しよう

In [ ]:
```

```

(4) 10人の身長を降順で並び替えたリストをhigh4で作成し、表示しよう

In [ ]:
```

```

(5) for文を利用し、10人の身長の平均値を計算して表示しよう

In [ ]:
```

```

(6) 身長158cm以下の人をhigh4から削除して表示しよう

In [ ]:
```

```

解答例

In [ ]:

```
# (1)
high1 = [170, 175, 160, 155, 165]
print("要素数: ", len(high1))

# (2)
high1[3] = 150 # 4番目の人のインデックスは3であることに注意
print("修正:", high1)

# (3)
high2 = [180, 155, 160, 165, 170]
high3 = high1 + high2
print("結合:", high3)

# (4)
high4 = sorted(high3, reverse=True)
print("並び替え:", high4)

# (5)
sum = 0
for ele in high4:
    sum = sum + ele      # ele += eleでもOK
ave = sum / len(high4)
print("平均", ave)

# (5)
high4.remove(150)
high4.remove(155)
print(high4)
```

リスト内の要素の統計計算

- 標準ライブラリの statistics を用いれば、平均、中央値、最頻値、分散、標準偏差を算出できる
  - ライブラリとは、よく使う機能・関数をまとめて、簡単に使えるようにしたもの
  - ライブラリの中身のプログラムがブラックボックスであっても、入力と出力がわかっていれば簡単に実装し、実行することができる
  - Pythonに便利な機能を持つライブラリが多数ある
- 標準ライブラリを使う場合、import文で呼び出す

```
import statistics
statistics.mean(リスト)       # 平均値
statistics.median(リスト)     # 中央値
statistics.median(リスト)     # 最頻値
statistics.pvariance(リスト) # 分散(データのばらつき)
statistics.pstdev(リスト)     # 標準偏差(分散の平方根)
```

In [ ]:

```
import statistics

listJ = [90, 85, 75, 95, 80, 70, 75, 90, 80, 75] # 5人の点数

mean = statistics.mean(listJ)
print("平均値: ",  mean)

median = statistics.median(listJ)
print("中央値: ", median)

mode = statistics.mode(listJ)
print("最頻値: ", mode)
```

```
pvariance = statistics.pvariance(listJ)
print("分散: ", pvariance)

pstdev = statistics.pstdev(listJ)
print("標準偏差: ", pstdev)
```

練習

10人の身長をまとめたリストとしてhighA = [172, 175, 165, 168, 178, 172, 170, 166, 169, 164]がある。平均値、分散、標準偏差を求めてみよう。ただし、statisticsのライブラリをインポートして使用するものとする。

In [ ]:
```
highA = [172, 175, 165, 168, 178, 172, 170, 166, 169, 164]
```

In [ ]:
```

```

解答例

In [ ]:
```
import statistics

mean = statistics.mean(highA)
print("平均値: ",  mean)

var = statistics.pvariance(highA)
print("分散: ", var)

std = statistics.pstdev(highA)
print("標準偏差: ", std)
```

リストの応用例

- `wikipediaの記事を取得するプログラム` を作成してみよう。また、このプログラムが日常生活で使えそうかを考えてみよう。
- wikipediaというライブラリを使用する
  - ライブラリとは、よく使う機能・関数をまとめて、簡単に使えるようにしたもの
  - ライブラリの中身のプログラムがブラックボックスであっても、入力と出力がわかっていれば簡単に実装し、実行することができる
  - Pythonに便利な機能を持つライブラリが多数ある
- ライブラリに関する参考ページ

ライブラリのインストール

Google Colaboratory (略してGoogle Colab)を使用する場合

- 以下セルのコメントアウト//を削除して実行

Anacondaを使用する場合

- Anaconda promptで、 `conda install -c conda-forge wikipedia` を実行
- 参考

In [ ]:
```
# ! pip install wikipedia
```

関連記事の検索

- 今回は Python の関連記事を検索する
- 検索結果は リスト で出力される
- 使用方法

  関連記事のリスト = wikipedia.search(検索ワード, results=検索数)

In [ ]:
```
import wikipedia # ライブラリのインポート
wikipedia.set_lang('ja') # 日本語版wikipediaの指定

# 検索単語を入力
word = input("検索単語を入力:")

# 関連記事のタイトルを検索して取得（今回は10件検索）
list_titles = wikipedia.search(word, results=10)

# len関数を使って個数を表示
print("関連記事:", len(list_titles))

# 関連記事のタイトルを表示
print(list_titles)
```

関連記事のurlを検索

- 使用方法

  url = wikipedia.page(タイトル).url

In [ ]:
```
for title in list_titles:           # 反復処理でタイトルを１つずつ取り出す
    url = wikipedia.page(title).url # urlを取得
    print(title, url)               # タイトルとurlの出力
```

関連記事の要約文を取得

- 使用方法

  要約文 = wikipedia.summary(タイトル, sentences=文章数)

In [ ]:
```
for title in list_titles:                           # 反復処理でタイトルを１つずつ取り
    summary = wikipedia.summary(title, sentences=1) # 最初の1文だけ取得
    print(title)                                    # タイトル表示
    print(summary)                                  # 要約表示
    print("------------------")                     # 区切り線
```

関連記事の全文取得

全文 = wikipedia.page(タイトル).content

全文はfor文を用いると出力結果が膨大になるので、１つ記事のみ出力

In [ ]:
```
# インデックス0番目の関連記事の全文取得
title   = list_titles[0]
content = wikipedia.page(title).content
print(content)
```

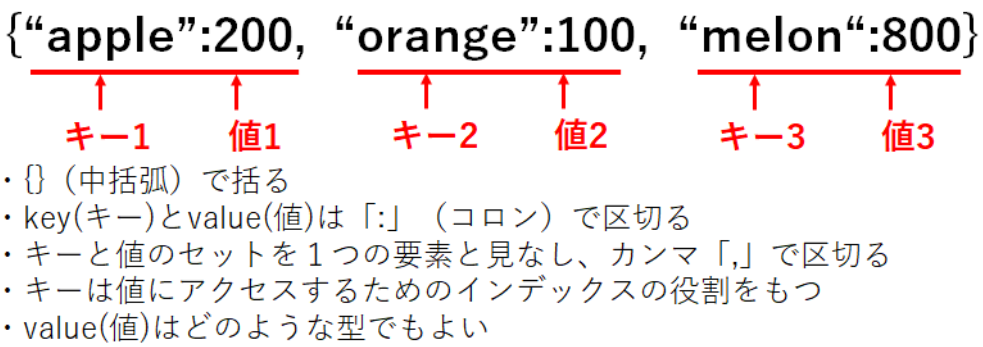
練習

「2.3.2 関連記事の検索」に戻って、自分が調べたい検索ワードを入力してみよう。その後、urlの検索、要約の表示、全文表示を試してみよう。

以降、付録

辞書（ディクショナリ）

- 辞書 は、 リスト と同じように複数の値をまとめて扱う場合に使用する
- 辞書 と リスト の違い
  - リスト: インデックス（要素番号）を使って要素を取り出す
  - 辞書: key (キー)を使って要素を取り出す(キーはユーザーが決める)
- 辞書 は、 key (キー)と value （値）のセットが1つの要素となり、1つの 辞書 の中で同じ key を使用することができない
- 辞書 は全体を {} (中括弧)で囲み、 key と値は ： （コロン）で区切り、各要素（キーと値のペア）は , （カンマ）で区切る
- 辞書 の要素の並びに順番の考え方がない(リストでは左の要素から順に順番がある、つまりインデックス)
- 辞書 のkeyを使って各要素の value (値)にアクセスする
- 辞書 にはどのような型の値でも格納することができる



辞書の操作の基礎

辞書を変数に代入

In [ ]:

```
#テストの点数がAさん90点、Bさん85点、Cさん75点、Dさん95点、Eさん80点であったとする
dicA = {"Aさん":90, "Bさん":85, "Cさん":75, "Dさん":95, "Eさん":80} # 5人の点数

# 辞書表示
print(dicA)
```

要素数（キーと値のペア）をカウント

- len関数 を用いる

  len(辞書)

In [ ]:

```
print("要素数：", len(dicA))
```

値（value）の取り出し

辞書[キー]

- リストでは リスト[インデックス] で要素を取り出したことと考え方は同じ

In [ ]:
```
# Aさんの点数をprint関数で出力
print(dicA["Aさん"], "点")

# Aさんの点数と取り出して、valという変数に代入
val = dicA["Aさん"]
print(val, "点")

# キーを変数にしたパターン
key = "Aさん"
val = dicA[key]
print(val, "点")
```

値（value）の変更

辞書[キー] = 値

In [ ]:
```
# 変更前のdicAを出力
print("変更前", dicA)

# Aさんの点数を60点に変更
dicA["Aさん"] = 60

# 変更後のdicAを出力
print("変更後", dicA)
```

新しい要素（キーと値のペア）の追加

辞書[新しいキーの名前] = 新しい値

In [ ]:
```
# 追加前のdicAを出力
print("追加前", dicA)

# Fさん70点を追加
dicA["Fさん"] = 70

# 追加後のdicAを出力
print("追加後", dicA)
```

反復（for文）による要素（キーと値）の取得

```
for キー in 辞書:
    値 = 辞書(キー)
```

- 辞書のキーのみがfor文の変数に代入される
- for文の処理の中で、キーを用いて値を取得する

要素の平均値を計算

5人のテストの点数の平均値

In [ ]:
```
dicA = {"Aさん":90, "Bさん":85, "Cさん":75, "Dさん":95, "Eさん":80} # 5人の点数
```

```
sum = 0                      # 合計値の初期値を0とする
for key in dicA:             # キーを１個ずつ取り出して変数keyに代入
    sum = sum + dicA[key]    # キーを使ってdicAの値を取り出してsumに加算

# 平均値 = 合計/要素数
ave = sum / len(dicA)  # 平均値をaveという変数に代入

print("平均値：", ave)
```

練習

5人の身長を測った結果、佐藤さん170cm、鈴木さん175cm、高橋さん160cm、田中さん155cm、伊藤さん165cmであったとする

(1) 5人の身長を要素とする辞書をdic_highの変数名で作成して、要素数を出力しよう

In [ ]:

(2) 高橋さんの身長に間違いがあったため150cmに変更して、辞書を出力しよう

In [ ]:

(3) 新たに渡辺さん180cmを追加し、辞書を出力しよう

In [ ]:

(4) 身長の平均値を計算し、出力しよう。ただし、for文を使って辞書の要素を取り出すものとする

In [ ]:

解答例

In [ ]:

```
#(1)
dic_high = {"佐藤さん":170, "鈴木さん":175, "高橋さん":160, "田中さん":155, "伊藤さん"
print(dic_high)

#(2)
dic_high["高橋さん"] = 150
print(dic_high)

#(3)
dic_high["渡辺さん"] = 180
print(dic_high)

#(4)
sum = 0
for key in dic_high:
    sum = sum + dic_high[key]
ave = sum / len(dic_high)
print("平均値:", ave)
```

メソッドを用いた辞書の操作

- メソッド とはある処理のまとまりに名前をつけたもので、オブジェクト（変数や値のこと）の後にドット「.」を付けて使用する
  辞書.メソッド名()

全てのキーの集合を取得

- 使用メソッド

  辞書.keys()

In [ ]:
```
#テストの点数がAさん90点、Bさん85点、Cさん75点、Dさん95点、Eさん80点であったとする
dicA = {"Aさん":90, "Bさん":85, "Cさん":75, "Dさん":95, "Eさん":80} # 5人の点数

# 全てのキーの集合を出力
print(dicA.keys())
```

In [ ]:
```
# for文でキーの集合から個別のキーを取得
for key in dicA.keys():
    print(key, end=",")
```

全ての値の集合を取得

- 使用メソッド

  辞書.values()

In [ ]:
```
# 全ての値の集合を出力
print(dicA.values())
```

In [ ]:
```
# for文でキーの集合から個別の値を取得
for val in dicA.values():
    print(val, end=",")
```

全てのキーと値の集合を取得

- 使用メソッド

  辞書.items()

In [ ]:
```
# 全てのキーと値の集合を出力
print(dicA.items())
```

In [ ]:
```
# for文でキーと値の集合から個別のキーと値を取得
for key, val in dicA.items():
    print(key, val)
```

値の取り出し

- 使用メソッド

  辞書.get(キー)

- 辞書.get(キー) と 辞書[キー] の違いは、該当するkeyがなかったときの処理方法

- 辞書.get(キー) : 指定したキーがないと、None(=「ない」)を返す
- 辞書[キー] : 指定したキーがないと、エラーになる

In [ ]:
```
# Fさんの値を取り出し
key = "Fさん"
print(key, dicA.get(key))

# Eさんの値を取り出し
key = "Eさん"
print(key, dicA.get(key))
```

要素（キーと値のペア）の削除

- 使用メソッド
  辞書.pop(キー)

In [ ]:
```
# 削除前の辞書を出力
print("削除前: ", dicA)

# Aさんに該当する要素（キーと値のペア）を削除
dicA.pop('Aさん')

# 削除後の辞書を出力
print("削除後: ", dicA)
```

練習

学生Aから学生Hまでの10人のテストの点数が、80, 70, 70, 60, 60, 90, 80, 80, 70, 60 (点)とする

(1) 学生をキー、点数を値としてdicBという辞書を作成して、表示しよう。

In [ ]:
```

```

(2) 学生Bの点数を表示しよう

In [ ]:
```

```

(3) for文を用いて点数を取り出して足し合わせた後、平均点を求めて表示しよう

In [ ]:
```

```

(4) 新たに受験した学生Iの点数は50点であった。辞書に追加して表示しよう

In [ ]:
```

```

(5) for文を用いて最高点の学生をチェックし表示してみよう

In [ ]:
```

```

解答例

In [ ]:
```
#(1)
dicB = {"学生A":80, "学生B":70, "学生C":70, "学生D":60, "学生E":60, "学生F":90, "学生(
```

```
print(dicB)

#(2)
print(dicB["学生B"])

#(3)
sum = 0
for val in dicB.values():
    sum = sum + val # sum += valでも可
ave = sum / len(dicB)
print("平均値: ", ave)

# (4)
dicB["学生I"] = 50
print(dicB)

# (5)
max_val = 0 # 最高点を保持する変数（初期値0としておく）
for key, val in dicB.items():
    if(val > max_val): # もし、現在の最大点より大きな値が来たら最大点を更新する
        max_val = val  # 最大点の保存
        max_key = key  # 最大点のキーを保存
print("最高点", max_key, max_val)
```
