

## 演習2

Python基礎 Pythonの変数とデータの型



統合教育機構 曹 日丹

# 医療とAI・ビッグデータ入門

## 演習2-7の構成

Python基礎を学びましょう

演習2 11/16 11:35-12:20

Pythonの変数とデータの型

演習3

プログラミング基礎

演習4

モジュール、パッケージ、ライブラリ

Pythonを使ってみましょう

演習5

患者の歯に関する病院のリアルワールドデータの説明

架空データ

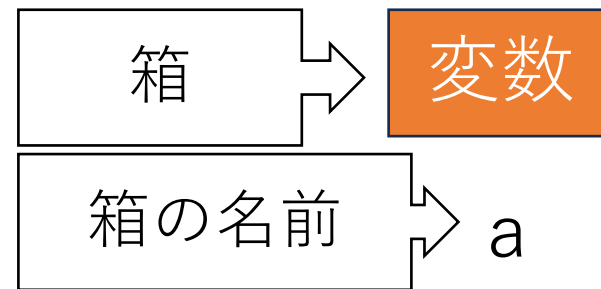
演習6

データクレンジングに必要なライブラリ（Pandas）の応用

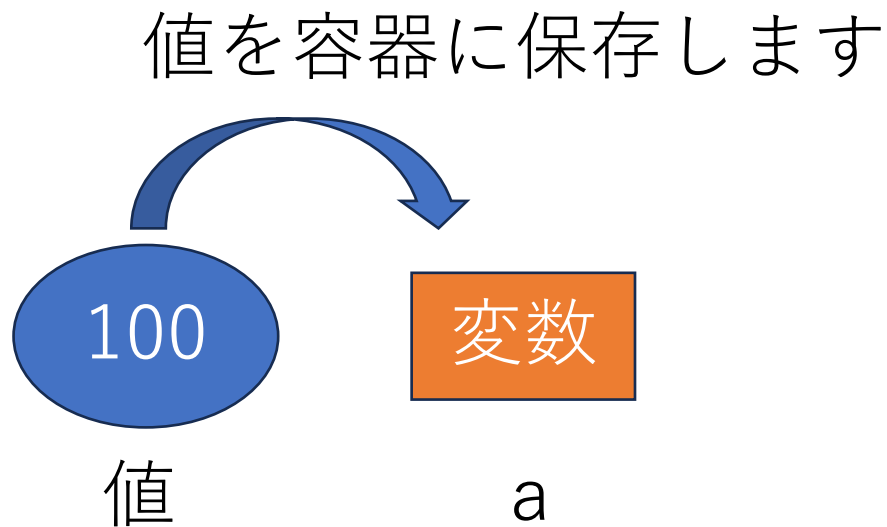
演習7

データクレンジングとデータの可視化

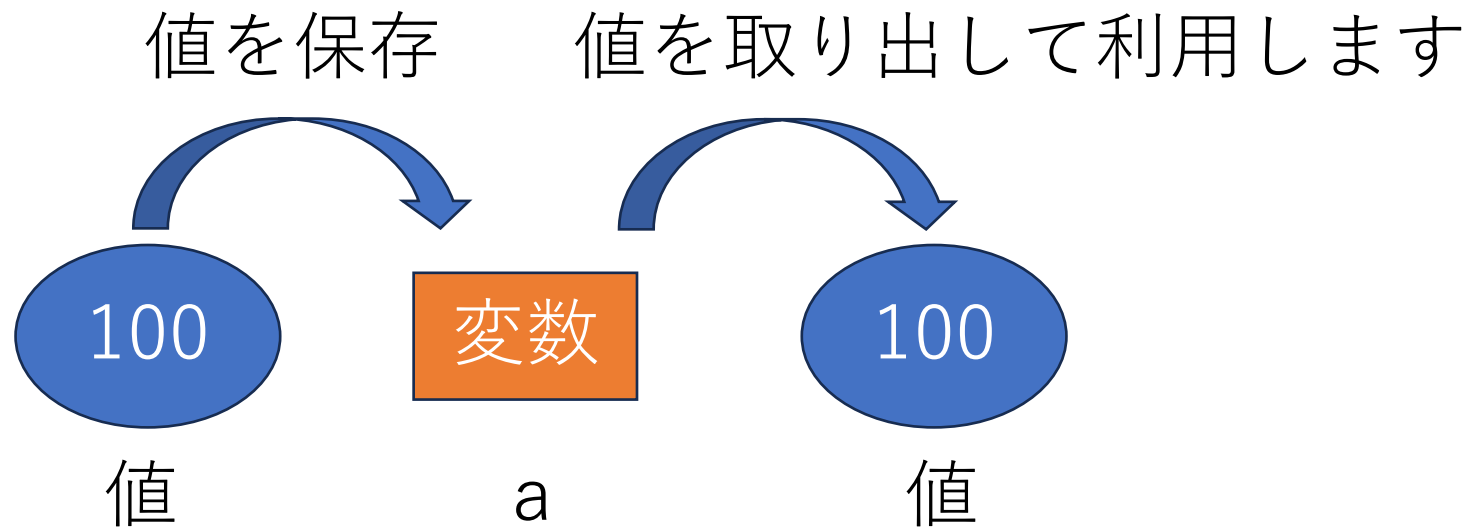
**変数**とは、データや値を一時的に保持するための名前付きの「箱」のようなものです。  
この変数を使用することで、プログラム中でその値を繰り返し利用することができます。



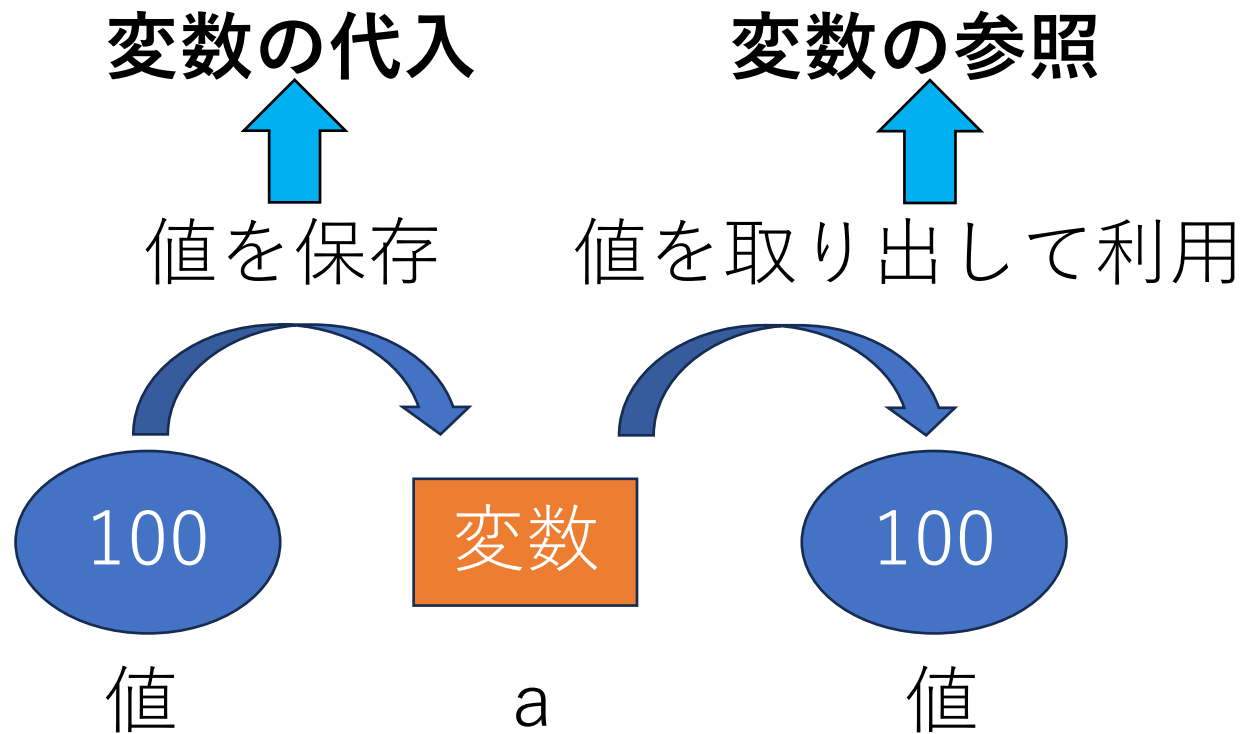
**変数**とは、データや値を一時的に保持するための名前付きの「容器」または「箱」のようなものです。この変数を使用することで、プログラム中でその値を繰り返し利用することができます。



**変数**とは、データや値を一時的に保持するための名前付きの「容器」または「箱」のようなものです。この変数を使用することで、プログラム中でその値を繰り返し利用することができます。



**変数**とは、データや値を一時的に保持するための名前付きの「容器」または「箱」のようなものです。この変数を使用することで、プログラム中でその値を繰り返し利用することができます。



**変数**とは、データや値を一時的に保持するための名前付きの「容器」または「箱」のようなものです。  
この変数を使用することで、プログラム中でその値を繰り返し利用することができます。

## コード01

プログラミング言語  
で表現します

### 変数の代入

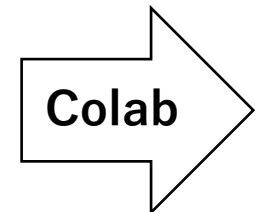
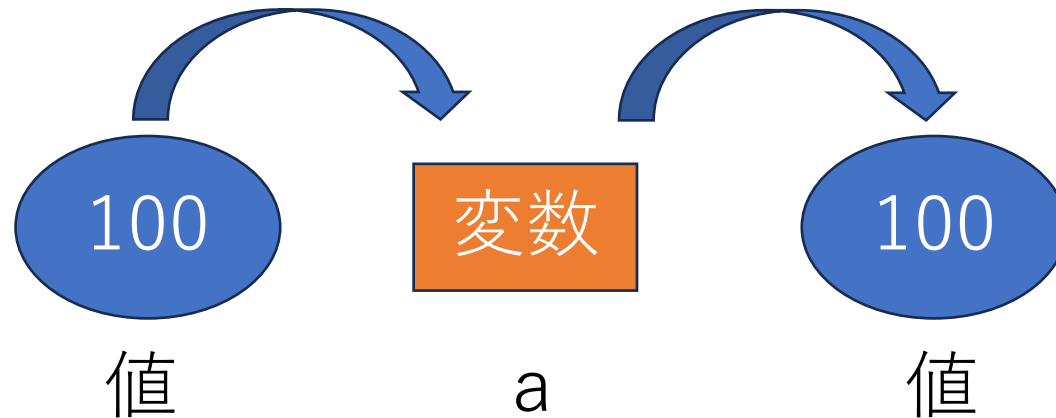
```
a = 100
```

値を保存

### 変数の参照

```
print(a)
```

値を取り出して利用



# 検索google colab Colaboratory へようこそ - Colaboratory - Google

 Colaboratory へようこそ  
ファイル 編集 表示

目次

はじめに

データサイエンス

機械学習

その他のリソース

使用例

セクション

ノートブックを開く

例

最近

Google ドライブ

GitHub

アップロード

ノートブックを検索

タイトル

最終閲覧

最初に開いた日時

例

Colaboratory へようこそ

ファイル名

ファイル名

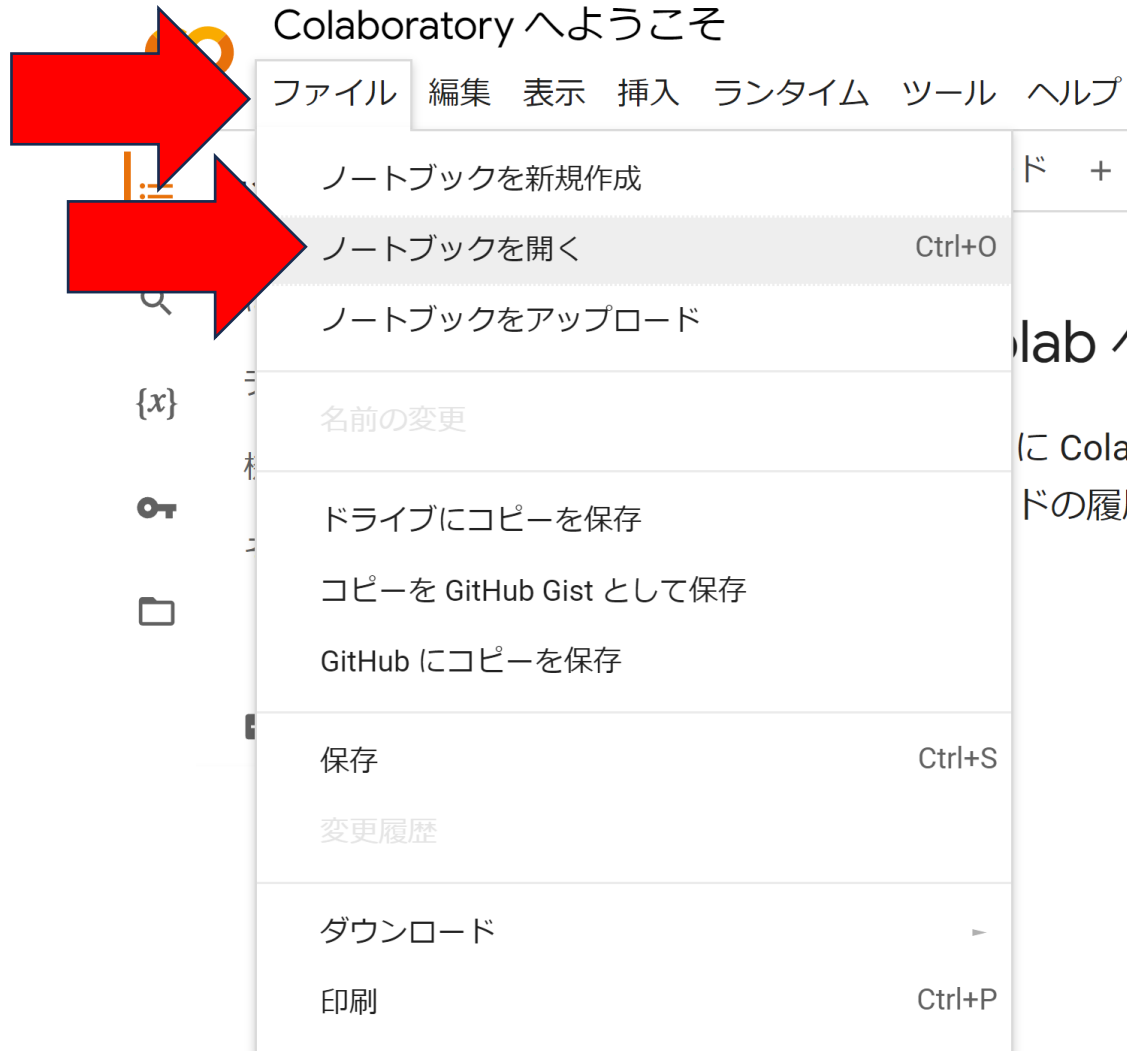
ファイル名

+ ノートブックを新規作成

キャンセル



# 検索google colab Colaboratory へようこそ - Colaboratory - Google



## Colab へようこそ

に Colab をよくご存じの場合は、この動画でインタラクティブなラドの履歴表示、コマンド パレットについてご覧ください。



Colab とは

# 検索google colab Colaboratory へようこそ - Colaboratory - Google

ノートブックを開く













例 > 🔍 ノートブックを検索

最近 >

Google ドライブ > **演習2コード.ipynb**

GitHub >

アップロード >

タイトル	所有者	最終閲覧 ▲	最終更新 ▼		
 演習準備資料.ipynb	曹日丹	11月1日	11月1日		
 演習1116確認.ipynb のコピー	曹日丹	10月31日	10月27日		
 2023入門dataframe.ipynb	曹日丹	10月31日	10月27日		
 演習1116確認.ipynb	曹日丹	10月25日	10月25日		

+ ノートブックを新規作成

キャンセル

# 検索google colab Colaboratory へようこそ - Colaboratory - Google

ノートブックを開く

例 >

最近 >

Google ドラ  
イブ >

GitHub >

アップロード >



または、ここにファイルをドラッグしてください

演習：コード01を書いてみましょう

GoogleColabでノートブックを開きましょう



ここにコードを入力します

[ ]

[ ]

[ ]

演習：コード01を書いてみましょう

コード01

```
a = 100
```

```
print(a)
```



```
a = 100
```

```
print(a)
```

実行ボタンを押す、またはShift+Enterを押すと現在のセルのコードを実行できます。

```
[ ]
```

演習：コード01を書いてみましょう

コード01

```
a = 100
```

```
print(a)
```



```
a = 100
```

```
[ ]
```

```
print(a)
```



```
100
```

```
[ ]
```

```
[ ]
```

Colab

演習：コード01を書いてみましょう

コード01

```
a = 100  
print(a)
```



```
1 a = 100  
2 print(a)  
3 a
```



```
100  
100
```

+ コード + テキスト



```
1 a = 100  
2 print(a)
```

変数の代入

変数の参照



```
100
```

参照した結果

```
[ ] 1
```

演習：コード02を書いてみましょう

コード02

a= "hello world " → 文字を入力するため""が必要です。

print(a)

a= [11, 16] → リストを作成するため[]が必要です。

print(a)

▶ a= "hello world"

[ ] print(a)

[ ] a= [11, 16]

[ ] print(a)



演習：コード02を書いてみましょう



```
a= "hello world" # "" (ダブルクォーテーション) の中に文字を入れてみましょう
```

```
print(a)
```

```
a= [11, 16]
```

```
print(a)
```

# 記号は、コメントを示すために使われます。  
コメントは、コードを実行する際には無視されるテキスト

# []角括弧の中にリストを作きましょう

hello world  
[11, 16]

1回目に実行した結果

文字

2回目に実行した結果

リスト

Colab

## データの型：

Pythonにはさまざまなデータ型が存在します。以下はPythonの基本的なデータ型です：

### 基本データ型

### コンテナデータ型

## データの型：

Pythonにはさまざまなデータ型が存在します。以下はPythonの基本的なデータ型です：

### 基本データ型

int(整数)

float(浮動小数点数)

### コンテナデータ型

str (文字列)

list (リスト)

tuple (タプル)

set (セット)

dict (辞書)

## データの型：

Pythonにはさまざまなデータ型が存在します。以下はPythonの基本的なデータ型です：

### 基本データ型

**int**(整数)

a = 5

**float**(浮動小数点数)

b = 5.0

### コンテナデータ型

**str** (文字列)

greeting = "Hello, World!"

**list** (リスト)

numbers = [1, 2, 3, 4, 5]

**tuple** (タプル)

colors = ("red", "green", "blue")

**set** (セット)

fruits = {"apple", "banana", "cherry"}

**dict** (辞書)

person = {"name": "John", "age": 30, "city": "New York"}

## デモ：データの型を確認しましょう

```
[ ] a = 5  
    b = 5.0  
    c = 3+4*b
```

### 基本データ型

```
greeting = "Hello, World!"  
numbers = [1, 2, 3, 4, 5]  
colors = ("red", "green", "blue")  
person = {"name": "John", "age": 30, "city": "New York"}  
fruits = {"apple", "banana", "cherry"}
```

### コンテナデータ型

## デモ：データの型を確認しましょう

type関数で型を確認できます



```
type(a)
```

```
[ ]
```

```
type(c)
```

```
[ ]
```

```
type(numbers)
```

```
[ ]
```

変数エクスペローラで型を確認できます

変数



名前	型	形状	値
a	int		5
b	float		5.0
c	float		23.0
colors	tuple	3 items	('red', 'green', 'blue')
fruits	set	3 items	{'apple', 'banana', 'cherry'}
greeting	str	13 chars	'Hello, World!'
numbers	list	5 items	[1, 2, 3, 4, 5]
person	dict		{'name': 'John', 'age': 30, 'c

Colab

Pythonにおけるset、dict（辞書）、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

### リスト（list）の特徴：

順序付けられた要素の集まり

要素の重複が可能

異なるデータ型の要素を含むことができる

要素の追加、削除、変更が可能

各要素はインデックス（位置）によってアクセスされる。

Pythonにおけるset、dict（辞書）、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

### リスト（list）の特徴：

順序付けられた要素の集まり

要素の重複が可能

異なるデータ型の要素を含むことができる

要素の追加、削除、変更が可能

各要素はインデックス（位置）によってアクセスされる。

### セット（set）特徴:

重複しない要素の集まり。

要素の追加や削除は可能。

順序付けられない。

### タプル（Tuple）特徴:

順序付けられた要素の集まり。

変更不可能です。

### 辞書 (Dictionary)特徴:

順序付けられたキーと値の集まり。

要素の追加、削除、変更することが可能。



## Python基礎 Pythonの変数とデータの型

Pythonにおける**set**、**dict**（辞書）、**tuple**、**list**は、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード03を書いてみましょう

リストを作成します。

```
▶ fruits = [ , , , , ]
```

文字の要素で構成されています。

```
[ ] numbers = [ , , , , ]
```

数字の要素で構成されています。

```
[ ] mixed = [数字, 文字, リスト]
```

複数の要素で構成されています。

```
[ ]
```

角括弧で定義されます、要素の間にカンマで区切られます。

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード03を書いてみましょう

リストを作成します。

```
▶ fruits = ["apple", "banana", "cherry", "melon"]
```

文字の要素で構成されています。

```
[ ] numbers = [1, 2, 3, 4, 5]
```

数字の要素で構成されています。

```
[ ] mixed = [1, "apple", 3.14, [5, 6, 7]]
```

複数の要素で構成されています。

```
[ ]
```

リストでは、異なるデータ型の要素を含むことができます。

Colab

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード03を書いてみましょう

```
▶ fruits = ["apple" , "banana" , "cherry", "melon"]
```

```
[ ] numbers = [1, 2, 3, 4, 5]
```

```
[ ] mixed = [1, "apple", 3.14, [5, 6, 7]]
```

```
[ ] type(mixed)
```

type()関数を使って型を確認できます。

Colab

Pythonにおけるset、dict（辞書）、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

### リスト（list）の特徴：

順序付けられた要素の集まり

要素の重複が可能

異なるデータ型の要素を含むことができる

要素の追加、削除、変更が可能

各要素はインデックス（位置）によってアクセスされる。

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード04を書いてみましょう

インデックスを使用して要素にアクセスします



```
fruits = ["apple", "banana", "cherry", "melon"]
```

```
[ ]
```

```
print(fruits[ ])
```

角括弧にインデックス数字を入力します。

```
[ ]
```

```
print(fruits[ ])
```

角括弧にインデックス数字を入力します。

```
[ ]
```

## Python基礎 Pythonの変数とデータの型

Pythonにおける**set**、**dict**（辞書）、**tuple**、**list**は、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード04を書いてみましょう

インデックスを使用して要素にアクセスします



```
fruits = ["apple", "banana", "cherry", "melon"]
```

```
[ ]
```

```
print(fruits[0])
```

```
[→] apple
```

“apple”を取り出しました

```
[ ]
```

```
print(fruits[-1])
```

```
[→] melon
```

"melon"を取り出しました

```
[ ]
```

```
fruits = ["apple", "banana", "cherry", "melon"]
```

fruits[0]

[1]

[2]

[3]

[-3]

[-2]

[-1]

インデックスを使用してリストの要素にアクセスします

```
fruits = ["apple", "banana", "cherry", "melon"]
```

fruits[0]

fruits[1]

fruits[2]

fruits[3]

fruits[-4]

fruits[-3]

fruits[-2]

fruits[-1]

リスト内の各要素は、0から始まるインデックス番号を持ちます。最初の要素はインデックス0にあり、次の要素は1、2、3と続きます。このインデックスを使用して、特定の要素にアクセスできます。

コード04 Colab

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict（辞書）、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード05を書いてみましょう

スライスを使用して部分リストを取得する：



```
fruits = ["apple", "banana", "cherry", "melon"]
```

[ ]

```
print(fruits[ : ]) 角括弧に範囲を表す数字を入力します。
```

[ ]

```
print(fruits[ : ]) 角括弧に範囲を表す数字を入力します。
```

[ ]



## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード05を書いてみましょう

スライスを使用して部分リストを取得する:



```
fruits = ["apple", "banana", "cherry", "melon"]
```

```
[ ]
```

```
print(fruits[1:3])
```

```
[→] ["banana", "cherry"]
```

```
[ ]
```

```
print(fruits[:2])
```

```
[→] ["apple", "banana"]
```

```
[ ]
```

```
fruits = ["apple", "banana", "cherry", "melon"]
```

```
fruits[0]
```

```
[1]
```

```
[2]
```

```
[3]
```

```
fruits[0]
```

```
[1]
```

```
[2]
```

```
[3]
```

スライスを使用することで、リスト内の特定の範囲の要素を取り出すことができます。

```
fruits = ["apple", "banana", "cherry", "melon"]
```

```
fruits[0]
```

```
fruits[1]
```

```
fruits[2]
```

```
fruits[3]
```

```
fruits[1]
```

```
fruits[2]
```

```
fruits[1:3]
```

```
fruits[0]
```

```
fruits[1]
```

```
fruits[0:2]
```

スライスは次の形式を取ります：start:end:step。  
startはスライスの開始位置を示します、  
endはスライスの終了位置を示します（この位置は含まれません）  
stepはステップ数を指定します（指定しない場合はデフォルトで1となります）

コード05 Colab

Pythonにおけるset、dict（辞書）、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

### リスト（list）の特徴：

順序付けられた要素の集まり

要素の重複が可能

異なるデータ型の要素を含むことができる

要素の追加、削除、変更が可能

各要素はインデックス（位置）によってアクセスされる。

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード06を書いてみましょう

リストの要素を変更します



```
fruits = ["apple", "banana", "cherry", "melon"]
```

```
[ ]
```

```
fruits[ ] = "blueberry"
```

変更したい要素のインデックス数字を入力します。

```
[ ]
```

```
print(fruits)
```

```
[ ]
```

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード06を書いてみましょう

リストの要素を変更します

```
1 fruits = ["apple", "banana", "cherry", "melon"]
```

```
[ ] 1 fruits[1] = "blueberry"
```

```
[ ] 1 print(fruits)
```

[→] ["apple", "blueberry", "cherry", "melon"]

```
fruits = ["apple", "banana", "cherry", "melon"]
```

fruits[0]

[1]

[2]

[3]

コード06 Colab

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード07を書いてみましょう

リストの要素を追加、削除します：



```
fruits.append("○○")
```

追加したい要素を入力します。

```
[ ]
```

```
print(fruits)
```

新しいリストを表示します。

```
[ ]
```

```
fruits.remove("○○")
```

削除したい要素を入力します。

```
[ ]
```

```
print(fruits)
```

新しいリストを表示します。

## Python基礎 Pythonの変数とデータの型

Pythonにおけるset、dict (辞書)、tuple、listは、いずれも複数の要素を格納できるデータ構造ですが、それぞれ異なる特性や用途を持っています。

演習：コード07を書いてみましょう

リストの要素を追加、削除します：



```
fruits.append("strawberry")
```

追加したい要素を入力します。

```
[ ] print(fruits)
```

```
[→] ["apple" , "banana" , "cherry", "melon", "strawberry"]
```

```
[ ] fruits.remove("cherry")
```

削除したい要素を入力します。

```
[ ] print(fruits)
```

```
[→] ["apple" , "banana" , "melon", "strawberry"]
```

コード07 Colab

さまざまな関数を使用してリストの計算ができます。

**sum 関数**：数値を含むリストの全要素の合計を計算する関数です。

① `total = sum([1,3,5,7,11])`  
`print(total)`

括弧の中数字を入れます。

② リスト = [ , , , ]

`total = sum(リスト名)`  
`print(total)`

括弧の中数字から構成されている  
リスト名を入れます。



さまざまな関数を使用してリストの計算ができます。

**sum 関数**：数値を含むリストの全要素の合計を計算する関数です。

① `total = sum([1,3,5,7,11])`  
`print(total)`

27

括弧の中数字を入れます。

② リスト = `[a1, a2, a3, a4]`

`total = sum(リスト名)`  
`print(total)`

`a1+ a2+ a3+ a4`

括弧の中数字から構成されている  
リスト名を入れます。

さまざまな関数を使用してリストの計算ができます。

演習：コード08を書いてみましょう

リストの合計を計算します：

▶ numbers = [10, 20, 30, 40, 50, 60]

[ ] total = sum(numbers)

[ ] print( total )

括弧の中数字から構成されている  
リスト名を入れます。

[ ] [→] 210

コード08 Colab

さまざまな関数を使用してリストの計算ができます。

**len関数**：あるオブジェクトが持つ要素の数を返す組み込み関数です。

① 文字数が集計されます。

② リストの要素の数が集計されます。

さまざまな関数を使用してリストの計算ができます。

**len関数**：あるオブジェクトが持つ要素の数を返す組み込み関数です。

① 文字数が集計されます。

```
greeting = " Hello World"
```

② リストの要素の数が集計されます。

```
リスト = [ , , , ]
```

さまざまな関数を使用してリストの計算ができます。

**len関数**：あるオブジェクトが持つ要素の数を返す組み込み関数です。

① `greeting = "Hello World"`

```
count = len(greeting)
print(count)
```

括弧の中変数名を入れます。

② `リスト = [ , , , ]`

```
total = len(リスト名)
print(total)
```

括弧の中リスト名を入れます。

さまざまな関数を使用してリストの計算ができます。

演習：コード09を書いてみましょう

リストの要素の数を取得します:

```
▶ numbers = [10, 20, 30, 40, 50, 60]
```

```
[ ] count = len(numbers)
```

```
[ ] print( count )
```

```
[→] 6
```

コード09 Colab

さまざまな関数を使用してリストの計算ができます。

**max** 関数：与えられたリストの中から最大値を返す組み込み関数です。

**min** 関数：与えられたリストの中から最小値を返す組み込み関数です。

```
▶ min_value = min(リスト名)
```

```
[ ] print(min_value)
```

```
[ ] max_value = max(リスト名)
```

```
[ ] print(max_value)
```

さまざまな関数を使用してリストの計算ができます。

演習：コード10を書いてみましょう

リスト内の最小値、最大値を取得します：

```
▶ min_value = min(numbers)
```

```
[ ] print(min_value)
```

[→] 10

```
[ ] max_value = max(numbers)
```

```
[ ] print(max_value)
```

[→] 60

コード10 Colab



さまざまな関数を使用してリストの計算ができます。

演習：コード11を書いてみましょう

リスト内の要素の平均値を計算します：



```
numbers = [10, 20, 30, 40, 50, 60]
```

```
[ ] average = sum(numbers) / len(numbers)
```

```
[ ] print(average)
```

```
[ ] [→] 35.0
```

Pythonは整数同士の割り算でも小数（浮動小数点数）の結果を返すように変更されました。

コード11Colab

演習：コード12を書いてみましょう

リストとリストを結合することもできます



```
newlist = リスト1名 + リスト2名
```

```
[ ] print(newlist)
```

```
[ ]
```

演習：コード12を書いてみましょう

リストとリストを結合することもできます

 `newlist = fruits + numbers`

`[ ] numbers = [10, 20, 30, 40, 50, 60]`

`[ ] fruits = ["apple", "banana", "cherry", "melon"]`

`[ ] print(newlist)`

`[→] ['apple', 'banana', 'cherry', 'melon',  
10, 20, 30, 40, 50, 60]`

コード12 Colab 

**課題：**11月30日23:59までWebClassで提出してください。

### 課題1

下記5人の成績表リストを作成し、`print`一つを使って3番目と4番目の人の成績を表示するようにコードを書いてください。

成績表：97, 67, 89, 100, 87

### 課題2

`a = "A Python list is an ordered collection of items. This means that each item in the list has a definite order, and that order will not change unless the list is explicitly modified."`

Pythonの関数を使って変数`a`の文字数を答えてください。

# 医療とAI・ビッグデータ入門

## 演習2-7の構成

Python基礎を学びましょう

演習2 11/16 11:35-12:20

Pythonの変数とデータの型

Pythonを使ってみましょう

演習5

患者の歯に関する病院のリアルデータの説明

演習3 11/30 11:35-12:20

プログラミング基礎

演習6

データクレンジングに必要なライブラリ（Pandas）の応用

演習4

モジュール、パッケージ、ライブラリ

演習7

データクレンジングとデータの可視化