第 7 章

# プログラミングの基本

R と Python のプログラミングの基本構文を学習します。実際にコードを書いてプログラムを実行し、プログラミングにより行われていることを理解します。

この章のゴール

R と Python の基本構文を理解する。

# R/PYTHON の基本構文

## 四則演算

ここでは R と Python の 2 つの言語について、四則演算の実行方法をそれぞれ解説します。

### ■R 言語　四則演算　演算子一覧

| 算術演算 | 例 | R の演算子 | R での書き方 | 実行結果 |
|---|---|---|---|---|
| 足し算(加算) | 3+5 | + | 3 + 5 | 8 |
| 引き算(減算) | 8 - 2 | - | 8 - 2 | 6 |
| 掛け算(乗算) | 5×4 | * | 5 * 4 | 20 |
| 割り算(除算) | 25÷5 | / | 25/5 | 5 |
| 割り算(除算)<br>(切り捨て) | 25÷3 | %/% | 25 %/% 3 | 8 |
| 割り算(剰余)<br>(除算の余り) | 10 ÷ 3 は<br>3 あまり 1 | %% | 10 %% 3 | 1 |
| べき乗 | 3^2 | ^ | 3^2 | 9 |

上記演算子以外に、括弧を使うことも出来ます。括弧の中に計算したい任意の数を指定することで計算結果を得ることができます。また、複数の関数が含まれた計算を一度に行ったり、セミコロン；で区切ることで、複数の式をまとめて一行に書くこともできます。

以下四則演算子の例を示します。

足し算(加算)　+演算子

```
> 3+5
[1] 8
```

引き算(減算)：-演算子

```
> 8-2
[1] 6
```

掛け算(乗算):*演算子

```
> 5*4
[1] 20
```

割り算(除算)：/演算子

```
> 25/5
[1] 5
```

割り算(除算)_切り捨て：//演算子

```
> 25%/%3
[1] 8
```

割り算(剰余)_除算の余り：%演算子

```
> 10%%3
[1] 1
```

べき乗：**演算子

```
> 3^2
[1] 9
```

## ■Python 四則演算　演算子一覧

| 算術演算 | 例 | Python の演算子 | Python での書き方 | 実行結果 |
|---|---|---|---|---|
| 足し算(加算) | 3 + 5 | + | 3 + 5 | 8 |
| 引き算(減算) | 8 - 2 | - | 8 - 2 | 6 |
| 掛け算(乗算) | 5×4 | * | 5 * 4 | 20 |
| 割り算(除算) | 25÷5 | / | 25/5 | 5 |
| 割り算(除算)<br>(切り捨て) | 25÷3 | // | 25 // 3 | 8 |
| 割り算(剰余)<br>(除算の余り) | 10 ÷ 3 は<br>3 あまり 1 | % | 10 % 3 | 1 |
| べき乗 | 3^2 | ** | 3 ** 2 | 9 |

データ型が、整数型 int、浮動小数点型 float などの数値に対しては四則演算やべき乗の処理が行えます。int と float は異なる型ですが、計算を行うことができ、int と float の演算結果の型は float になります。

演算子を使う際には、**記号の両側に値を記述します。**半角英数字入力が基本(全角は避ける)で、大文字と小文字は区別されます。**また、**文法的な意味はありませんが、演算子の両側にひとつずつ空白を空けるとコードが読みやすくなります。この空白は Python のコーディング規約で推奨されています。

＜参考 URL:PEP 8 -- Style Guide for Python Code＞
https://www.python.org/dev/peps/pep-0008/#should-a-line-break-before-or-after-a-binary-operator

以下四則演算子の例を示します。

足し算(加算)　+演算子

```
3 + 5
```

```
8
```

引き算(減算):-演算子

```
8 - 2
```

```
6
```

掛け算(乗算):*演算子

```
5 * 4
```

```
20
```

割り算(除算):/演算子

```
25 / 5
```

```
5.0
```

割り算(除算)_切り捨て://演算子

```
25 // 3
```

```
8
```

割り算(剰余)_除算の余り:%演算子

```
10 % 3
```

```
1
```

べき乗:**演算子

```
3 ** 2
```

```
9
```

## 代入

代入とは、数学で使用される、式または関数において、その中に含まれている文字または変数を他の値で置きかえることを言います。Python や R 言語も同様に、変数に値を代入することで、オブジェクトに名前をつけることができます。

変数名に使える文字は、小文字の英字、大文字の英字、数字、アンダースコア(_)です。また、数字は名前の先頭には使えません。例えば、「A4」は使用できますが、「4A」はエラーになるので使用することができません。

代入する値には、データや計算結果が入り、また代入によって名前を付けたオブジェクトはその名前で参照することが可能です。

### ■R 言語　代入

R 言語では、通常は変数に値を代入する場合以下の様に表記します。

**変数名 <- 代入するもの(値)**

[<- ]の左辺に変数を、右辺に値を記述することで、変数に値が代入されます。

そのまま変数名を入力して実行すると、変数が表示され確認することができます。

```
> x<-'機械学習'
> x
[1] "機械学習"
```

例 1:x に 2 を代入

```
> x<-2
> x
[1] 2
```

例 2:y に 5 を代入

```
> y<-5
> y
[1] 5
```

例 3:z に x と y を足したものを代入

```
> z<-x+y
> z
[1] 7
```

R 言語　演算子は、他に「　　->」「=」「assign(”変数名”, 値)」も使用できます。また、()で囲むと代入と表示を同時に行うことができます。

```
> (x <- '機械学習')
[1] "機械学習"
```

## ■Python　代入

Python では、変数に値を代入する場合以下の様に表記します。

**変数名 = 代入するもの(値)**

[=(イコール)]の左辺に変数を。右辺に値を記述することで、変数に値が代入されます。

print 関数を使用すると変数が表示され確認することができます。

```
a='機械学習'
print(a)
```

```
機械学習
```

例 1:x に 2 を代入

```
x=2
print(x)
```

```
2
```

例 2:y に 5 を代入

```
y=5
print(y)
```

```
5
```

例 3:z に x と y を足したものを代入

```
z=x+y
print(z)
```

```
7
```

Python 代入演算子は以下のものも使用できる。

| 例 | Python の演算子 | Python での書き方 |
|---|---|---|
| a=a+b | += | a += b |
| a=a-b | -= | a -= b |
| a=a*b | *= | a *= b |
| a=a/b | /= | a /= b |

## 抽出

　データの中から条件を指定して必要な部分だけ抜き出すことができます。R 言語と Python での代表的な抽出のコード例を示します。

■R 言語　抽出
「オブジェクト:data」には「第4章　機械学習(教師なし学習)　主成分分析のデータ shuseibun.csv」を読み込んでいます。

●data より抽出したい行番号と列番号を指定して抽出

オブジェクト名[抽出したい行番号,抽出したい列番号]

例:1 行目の 5 列目のデータを抽出

```
> data[1,5]
[1] 66
>
```

●data より行番号が一致するデータを抽出

オブジェクト名[抽出したい行番号,]

例:1 行目のみを抽出

```
> data[1,]
  チャンネル名 ケーキ カフェ 食レポ スイーツ 食いしん坊 コンビニスイーツ
1  チャンネル1    55    4    66      66          3              60
```

※範囲を指定しても抽出できます

オブジェクト名[抽出したい行番号始まり:抽出したい行番号終わり,]

例:1 行目～3 行目を抽出

```
> data[1:3,]
  チャンネル名 ケーキ カフェ 食レポ スイーツ 食いしん坊 コンビニスイーツ
1  チャンネル1    55    4    66      66          3              60
2  チャンネル2     2    0    40      25         50              55
3  チャンネル3    30   60    40      30          0               0
```

●列番号で抽出

オブジェクト名[抽出したい列番号]

例:1 列目のみを抽出

```
> data[1]
    チャンネル名
1    チャンネル1
2    チャンネル2
3    チャンネル3
4    チャンネル4
5    チャンネル5
6    チャンネル6
7    チャンネル7
8    チャンネル8
9    チャンネル9
10 チャンネル10
11 チャンネル11
12 チャンネル12
13 チャンネル13
```

※範囲でも抽出できる

オブジェクト名[抽出したい列番号始まり:抽出したい列番号終わり]

例:1 列目～3 列目を抽出

```
> data[1:3]
    チャンネル名 ケーキ カフェ
1    チャンネル1    55    4
2    チャンネル2     2    0
3    チャンネル3    30   60
4    チャンネル4    30   59
5    チャンネル5     0    0
6    チャンネル6     2    0
7    チャンネル7    80   30
8    チャンネル8    10    0
9    チャンネル9     0    0
10 チャンネル10     0    4
11 チャンネル11    44   52
12 チャンネル12    25   40
13 チャンネル13    45   20
```

●ヘッダー名をキーに、比較演算子が一致するデータを抽出

文字列一致:オブジェクト名[オブジェクト名$ヘッダー名=="抽出したい条件",]

例:ヘッダー名「チャンネル名」が「チャンネル 13」の行を抽出

```
> data[data$チャンネル名 =="チャンネル13",]
   チャンネル名 ケーキ カフェ 食レポ スイーツ 食いしん坊 コンビニスイーツ
13 チャンネル13    45    20    66      60        20              50
```

※データフレームの変数とヘッダー名の区切りには「$」を使います。ヘッダー名は、クオートする必要はありません。数値比較の場合は比較演算子を使用します。(代表的な比較演算子は以下に記載しています。)キーに指定している「チャンネル 13」は文字列なので、ダブルクオート「"」で囲ってください。行を抽出したいので、最後にカンマ「,」が必要です。

●数値を比較して抽出　オブジェクト名[オブジェクト名$ヘッダー名＞比較したい数値,]

例:ヘッダー名[ケーキ]の値が 0 の行を抽出

```
> data[data$ケーキ  ==  0, ]
   チャンネル名 ケーキ カフェ 食レポ スイーツ 食いしん坊 コンビニスイーツ
5   チャンネル5     0     0     0      0        0               0
9   チャンネル9     0     0     0      0        0               0
10 チャンネル10     0     4    50     55       30              53
```

●複数の条件が一致するデータを抽出

比較演算を[ & ]記号で繋げると抽出できます。

例:ケーキが 20 以上、食レポが 30 より小さいデータ

```
> data[ data$ケーキ > 20 & data$食レポ < 30, ]
   チャンネル名 ケーキ カフェ 食レポ スイーツ 食いしん坊 コンビニスイーツ
4   チャンネル4    30    59     3      44        0               3
12 チャンネル12    25    40     0      30        2               2
```

●ソートして表示する

例:ヘッダー名「スイーツ」を昇順にソートして抽出

```
> data[ order( data$スイーツ ), ]
   チャンネル名 ケーキ カフェ 食レポ スイーツ 食いしん坊 コンビニスイーツ
5   チャンネル5     0     0     0      0        0               0
9   チャンネル9     0     0     0      0        0               0
6   チャンネル6     2     0     5     10       10              10
2   チャンネル2     2     0    40     25       50              55
3   チャンネル3    30    60    40     30        0               0
8   チャンネル8    10     0    50     30       52              60
12 チャンネル12    25    40     0     30        2               2
4   チャンネル4    30    59     3     44        0               3
11 チャンネル11    44    52    52     50        0               0
10 チャンネル10     0     4    50     55       30              53
13 チャンネル13    45    20    66     60       20              50
1   チャンネル1    55     4    66     66        3              60
7   チャンネル7    80    30    90     92        2               4
```

※降順にしたい場合は、「> data[ order( -data$スイーツ ), ]」とヘッダー名を指定する際にオブジェクトの前に「-」をつける。

R で使える論理演算子

| | |
|---|---|
| a=b 等しい(イコール) | a == b |
| a≠b ノットイコール(異なる) | a != b |
| a<b 未満(小なり) | a < b |
| a≦b 以下(小なりイコール) | a <= b |
| a>b 大きい(大なり) | a > b |
| a≧b 以上(大なりイコール) | a >= b |

R で使える論理演算子

| | |
|---|---|
| a かつ b（論理積） | a & b |
| a もしくは b（論理和） | a｜b |
| a 以外（否定） | ! a |

■Python 抽出

dfには「第4章 機械学習(教師なし学習) 主成分分析のデータ shuseibun.csv」を使用しています。

●df より抽出したい行番号と列番号を指定して抽出(pandas の iloc アトリビュートを使用し抽出します。)行や列は 0 行目・0 列目から始まります。

例:1 行目の 5 列目のデータを抽出

```
print(df.iloc[[0],[4]])
```

```
   スイーツ
0    66
```

●df より行番号が一致するデータを抽出(コロンを用いて抽出できます)

例:1 行目のみを抽出

```
print(df[0:1])
```

```
   チャンネル名  ケーキ  カフェ  食レポ  スイーツ  食いしん坊  コンビニスイーツ  ファッション  スクールメイク  プチプラ  コスメ
0  チャンネル1   55    4   66    66      3        60       44        30     40    45
```

※範囲を指定しても抽出できます

例:1 行目～3 行目を抽出

```
print(df[0:3])
```

```
   チャンネル名  ケーキ  カフェ  食レポ  スイーツ  食いしん坊  コンビニスイーツ  ファッション  スクールメイク  ブチブラ  コスメ
0  チャンネル1   55    4   66    66      3        60       44        30     40    45
1  チャンネル2    2    0   40    25     50        55        0         0      0     0
2  チャンネル3   30   60   40    30      0         0        0         0      0     0
```

●数値を比較して抽出

例:ヘッダー名[ケーキ]の値が 0 の行を抽出

```
print(df[df.ケーキ == 0])
```

```
     チャンネル名  ケーキ  カフェ  食レポ  スイーツ  食いしん坊  コンビニスイーツ  ファッション  スクールメイク  ブチブラ  コスメ
4    チャンネル5    0    0    0     0       0         0        2        20     40    60
8    チャンネル9    0    0    0     0       0         0      104        40     82    50
9   チャンネル10    0    4   50    55      30        53        0         0      0     0
17  チャンネル18    0    0    0     0      11        33       17         1     40    17
18  チャンネル19    0    0    0     0       0         0       60         5     50    44
25  チャンネル26    0    0    0     0       0         0       55         2     80    44
26  チャンネル27    0    0    0     0       0         0       40        13     61    90
28  チャンネル29    0   18   70    55      50        60        0         0      0     0
29  チャンネル30    0    0    0     0       0         0       80         3     90    40
37  チャンネル38    0    0    0     0       0         0       30        14     50    82
```

●複数の条件が一致するデータを抽出

例:ケーキが 20 以上、食レポが 30 より小さいデータ

```
print(df[(df.ケーキ > 20) & (df.食レポ < 30)])
```

```
     チャンネル名  ケーキ  カフェ  食レポ  スイーツ  食いしん坊  コンビニスイーツ  ファッション  スクールメイク  ブチブラ  コスメ
3    チャンネル4   30   59    3    44       0         3       20         1      2    44
11  チャンネル12   25   40    0    30       2         2       10         0      3    33
44  チャンネル45   30   47    2    40       1         2        5         1      4    30
```

●ソートして表示する

例:ヘッダー名「スイーツ」を昇順にソートして抽出

```
df = df.sort_values('スイーツ')
print(df)
```

| | チャンネル名 | ケーキ | カフェ | 食レポ | スイーツ | 食いしん坊 | コンビニスイーツ | ファッション | スクールメイク | プチプラ | コスメ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 18 | チャンネル19 | 0 | 0 | 0 | 0 | 0 | 60 | 5 | 50 | 44 | |
| 4 | チャンネル5 | 0 | 0 | 0 | 0 | 0 | 2 | 20 | 40 | 60 | |
| 29 | チャンネル30 | 0 | 0 | 0 | 0 | 0 | 80 | 3 | 90 | 40 | |
| 8 | チャンネル9 | 0 | 0 | 0 | 0 | 0 | 104 | 40 | 82 | 50 | |
| 26 | チャンネル27 | 0 | 0 | 0 | 0 | 0 | 40 | 13 | 61 | 90 | |
| 25 | チャンネル26 | 0 | 0 | 0 | 0 | 0 | 55 | 2 | 80 | 44 | |
| 37 | チャンネル38 | 0 | 0 | 0 | 0 | 0 | 30 | 14 | 50 | 82 | |
| 17 | チャンネル18 | 0 | 0 | 0 | 0 | 11 | 33 | 17 | 1 | 40 | 17 |
| 39 | チャンネル40 | 3 | 20 | 18 | 5 | 0 | 0 | 25 | 1 | 20 | 30 |
| 19 | チャンネル20 | 5 | 30 | 40 | 6 | 0 | 0 | 0 | 0 | 0 | 0 |
| 32 | チャンネル33 | 4 | 40 | 30 | 10 | 5 | 5 | 30 | 4 | 40 | 50 |
| 5 | チャンネル6 | 2 | 0 | 5 | 10 | 10 | 10 | 1 | 30 | 33 | 37 |
| 43 | チャンネル44 | 6 | 40 | 30 | 10 | 2 | 3 | 0 | 0 | 0 | 0 |
| 38 | チャンネル39 | 4 | 40 | 30 | 12 | 2 | 4 | 22 | 3 | 30 | 62 |
| 23 | チャンネル24 | 2 | 40 | 12 | 16 | 20 | 22 | 2 | 1 | 55 | 20 |
| 33 | チャンネル34 | 15 | 50 | 45 | 20 | 0 | 3 | 0 | 0 | 0 | 0 |
| 36 | チャンネル37 | 11 | 0 | 0 | 20 | 20 | 20 | 5 | 33 | 55 | 55 |
| 24 | チャンネル25 | 20 | 27 | 25 | 25 | 5 | 10 | 13 | 11 | 18 | 24 |
| 1 | チャンネル2 | 2 | 0 | 40 | 25 | 50 | 55 | 0 | 0 | 0 | 0 |
| 15 | チャンネル16 | 8 | 6 | 33 | 26 | 28 | 28 | 4 | 1 | 50 | 44 |
| 11 | チャンネル12 | 25 | 40 | 0 | 30 | 2 | 2 | 10 | 0 | 3 | 33 |
| 7 | チャンネル8 | 10 | 0 | 50 | 30 | 52 | 60 | 0 | 0 | 0 | 0 |
| 2 | チャンネル3 | 30 | 60 | 40 | 30 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | チャンネル23 | 12 | 0 | 30 | 33 | 40 | 32 | 0 | 0 | 0 | 0 |
| 44 | チャンネル45 | 30 | 47 | 2 | 40 | 1 | 2 | 5 | 1 | 4 | 30 |
| 3 | チャンネル4 | 30 | 59 | 3 | 44 | 0 | 3 | 20 | 1 | 2 | 44 |
| 31 | チャンネル32 | 40 | 4 | 70 | 46 | 33 | 30 | 0 | 0 | 0 | 0 |
| 14 | チャンネル15 | 20 | 0 | 53 | 50 | 60 | 45 | 0 | 0 | 0 | 0 |
| 10 | チャンネル11 | 44 | 52 | 52 | 50 | 0 | 0 | 0 | 0 | 0 | 0 |
| 47 | チャンネル48 | 30 | 15 | 40 | 50 | 22 | 14 | 24 | 2 | 13 | 20 |
| 21 | チャンネル22 | 60 | 17 | 70 | 53 | 10 | 20 | 22 | 5 | 43 | 30 |
| 28 | チャンネル29 | 0 | 18 | 70 | 55 | 50 | 60 | 0 | 0 | 0 | 0 |
| 9 | チャンネル10 | 0 | 4 | 50 | 55 | 30 | 53 | 0 | 0 | 0 | 0 |
| 30 | チャンネル31 | 40 | 13 | 50 | 60 | 9 | 30 | 33 | 10 | 30 | 34 |
| 12 | チャンネル13 | 45 | 20 | 66 | 60 | 20 | 50 | 30 | 15 | 33 | 40 |
| 48 | チャンネル49 | 50 | 10 | 60 | 62 | 2 | 11 | 0 | 0 | 0 | 0 |
| 46 | チャンネル47 | 60 | 40 | 72 | 65 | 7 | 5 | 0 | 0 | 0 | 0 |
| 0 | チャンネル1 | 55 | 4 | 66 | 66 | 3 | 60 | 44 | 30 | 40 | 45 |
| 34 | チャンネル35 | 50 | 12 | 60 | 70 | 4 | 12 | 0 | 0 | 0 | 0 |
| 40 | チャンネル41 | 60 | 12 | 80 | 70 | 39 | 65 | 0 | 0 | 0 | 0 |
| 41 | チャンネル42 | 54 | 40 | 50 | 70 | 20 | 20 | 30 | 5 | 25 | 30 |
| 42 | チャンネル43 | 64 | 22 | 70 | 77 | 4 | 16 | 0 | 0 | 0 | 0 |
| 20 | チャンネル21 | 70 | 10 | 80 | 90 | 11 | 8 | 0 | 0 | 0 | 0 |
| 16 | チャンネル17 | 88 | 36 | 90 | 90 | 0 | 0 | 0 | 0 | 15 | 18 |
| 49 | チャンネル50 | 88 | 22 | 80 | 90 | 0 | 4 | 0 | 0 | 0 | 0 |
| 6 | チャンネル7 | 80 | 30 | 90 | 92 | 2 | 4 | 0 | 0 | 0 | 0 |
| 13 | チャンネル14 | 83 | 14 | 100 | 95 | 10 | 6 | 0 | 0 | 0 | 0 |
| 27 | チャンネル28 | 70 | 40 | 102 | 110 | 30 | 20 | 0 | 0 | 0 | 0 |
| 35 | チャンネル36 | 100 | 0 | 92 | 110 | 0 | 0 | 0 | 0 | 0 | 0 |
| 45 | チャンネル46 | 113 | 40 | 120 | 114 | 5 | 11 | 0 | 0 | 0 | 0 |

Python で使える論理演算子

| | |
|---|---|
| a=b 等しい(イコール) | a == b |
| a≠b ノットイコール(異なる) | a != b |
| a<b 未満(小なり) | a < b |
| a≦b 以下(小なりイコール) | a <= b |
| a>b 大きい(大なり) | a > b |
| a≧b 以上(大なりイコール) | a >= b |

Python で使える論理演算子

| | |
|---|---|
| a かつ b（論理積） | a & b |
| a もしくは b（論理和） | a \| b |
| a 以外（否定） | ~a |

## 代表値の算出

代表値とは、グループを代表する値(グループの中心の値)です。主に平均値、中央値、最頻値の 3 つを指します。

平均値： 全てのデータを足しあげて、その合計をデータの個数で割り算した値

最頻値： 一番多く出現している値(データの中で最も頻度が高い値)

中央値： データを大きい順に並べて、ちょうど真ん中にくる値。

左右対称のきれいな正規分布ではこれら 3 つの値は等しくなり、歪んだ分布だと数値にも偏りが生じ等しくなりません。分析や解析を行う場合は、取得したデータがどのような分布なのかをチェックする必要があります。

R 言語と Python それぞれの代表値を確認する演算子を次に示します。
「第4章 機械学習(教師なし学習) 主成分分析のデータ shuseibun.csv」のデータを使用します。

## ■R 言語　代表値の算出

R 言語では、次の関数を使用することで、データの代表値を求めることができます。

| | |
|---|---|
| 平均値 | mean() |
| 中央値 | median() |
| 最頻値 | table() ・mfv() |

※「オブジェクト:data」には先の sample データを読み込みます。

例:スィーツの平均値を表示

```
> mean(data$スイーツ)
[1] 42.84
```

例:スイーツの中央値を表示

```
> median(data$スイーツ)
[1] 42
```

例:スイーツの最頻値を表示

Table 関数を使用して、最頻値を求めることができます。
最頻値だけを表示するには、操作が必要です。

```
> # 頻度を数える
> table(data$スイーツ)

  0   5   6  10  12  16  20  25  26  30  33  40  44  46  50  53  55  60  62  65
  8   1   1   3   1   1   2   2   1   3   1   1   1   1   3   1   2   2   1   1
 66  70  77  90  92  95 110 114
  1   3   1   3   1   1   2   1
>
> # 頻度に基づいて表を並べ替える。頻度の高い順にするため並べ替えを逆転させる。
> rev(sort(table(data$スイーツ)))

  0  90  70  50  30  10 110  60  55  25  20 114  95  92  77  66  65  62  53  46
  8   3   3   3   3   3   2   2   2   2   2   1   1   1   1   1   1   1   1   1
 44  40  33  26  16  12   6   5
  1   1   1   1   1   1   1   1
>
> # 最頻値はこの並べ替えた表の最初の値
> rev(sort(table(data$スイーツ)))[1]
0
8
>
> # 最頻値の値のみ
> names(rev(sort(table(data$スイーツ))))[1]
[1] "0"
```

Modeest パッケージをインストールし mfv 関数を使用すると一回で求めることができます。

```
> library("modeest")
> mfv(data$スイーツ)
[1] 0
```

## ■Python　代表値の算出

Python では、次の関数を使用することでデータの代表値を求めることができます。

| | |
|---|---|
| 平均値 | mean() |
| 中央値 | median() |
| 最頻値 | mode() |

※「df」には先の sample データを読み込みます。

例：スィーツの平均値を表示

```
#平均値
print(df.mean()["スイーツ"])
```

```
42.84
```

例：スイーツの中央値を表示

```
#中央値
print(df.median().round(2)["スイーツ"])
```

```
42.0
```

例：スイーツの最頻値を表示

```
#最頻値
print(df.mode().min()["スイーツ"])
```

```
0.0
```

numpy パッケージの「describe」関数を使用すると、データの基本統計量を確認できます。 以下の順で基本統計量が出力されます。

1.データの個数（count）
2.平均（mean）
3.標準偏差（std）
4.最小値（min）
5.1/4 位値（25%）
6.中央値（50%）
7.3/4 位値（75%）
8.最大値（max）

```
#dfの基本統計量を表示
df.describe() .round(2)
```

| | ケーキ | カフェ | 食レポ | スイーツ | 食いしん坊 | コンビニスイーツ | ファッション | スクールメイク | プチプラ | コスメ |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 |
| mean | 31.66 | 19.48 | 43.32 | 42.84 | 12.38 | 16.72 | 14.16 | 5.02 | 20.04 | 21.46 |
| std | 31.95 | 18.78 | 33.59 | 34.04 | 16.49 | 19.93 | 22.70 | 9.65 | 25.72 | 24.51 |
| min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 25% | 2.25 | 0.00 | 6.75 | 10.50 | 0.00 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 |
| 50% | 22.50 | 14.50 | 42.50 | 42.00 | 4.50 | 9.00 | 0.50 | 0.00 | 2.50 | 17.50 |
| 75% | 54.75 | 40.00 | 70.00 | 65.75 | 20.00 | 26.50 | 23.50 | 4.75 | 40.00 | 40.00 |
| max | 113.00 | 60.00 | 120.00 | 114.00 | 60.00 | 65.00 | 104.00 | 40.00 | 90.00 | 90.00 |

## 集計

R 言語と Python での代表的なデータ集計のコード例を示します。

### ■R 言語　集計

R 言語では、以下の演算子を使用することで、データ集計が行えます。

| | |
|---|---|
| 平均(因子) | ave(x) |
| 5 数要約 | fivenum(x) |
| 4 分位偏差 | IQR(x) |
| 最大値 | max(x) |
| 平均 | mean(x) |
| 中央値 | median(x) |
| 最小値 | min(x) |
| クォンタイル点 | quantile(x) |
| 範囲 | range(x) |
| 不偏標準偏差 | sd(x) |
| 合計値 | sum(x) |
| 不偏分散 | var(x,y) |
| 重み付け平均 | weighted.mean(x) |

また、summary 関数を使用することで、【最小値・第1四分位・中央値・平均値・第3四分位・最大値】については、指定したデータについてデータの概要を個別に演算子を実行せずとも一気に求めることができます。

例：summary 関数をしようして、データの概要を確認する。

```
> summary(data)
     ケーキ            カフェ            食レポ          スイーツ
 Min.   :  0.00   Min.   : 0.00   Min.   :  0.00   Min.   :  0.00
 1st Qu.:  2.25   1st Qu.: 0.00   1st Qu.:  6.75   1st Qu.: 10.50
 Median : 22.50   Median :14.50   Median : 42.50   Median : 42.00
 Mean   : 31.66   Mean   :19.48   Mean   : 43.32   Mean   : 42.84
 3rd Qu.: 54.75   3rd Qu.:40.00   3rd Qu.: 70.00   3rd Qu.: 65.75
 Max.   :113.00   Max.   :60.00   Max.   :120.00   Max.   :114.00
    食いしん坊        コンビニスイーツ
 Min.   : 0.00     Min.   : 0.00
 1st Qu.: 0.00     1st Qu.: 0.50
 Median : 4.50     Median : 9.00
 Mean   :12.38     Mean   :16.72
 3rd Qu.:20.00     3rd Qu.:26.50
 Max.   :60.00     Max.   :65.00
```

## ■Python　集計

Python では、次の演算子を使用することで、データ集計を行います。

| | |
|---|---|
| データ数 | count() |
| 平均値 | mean() |
| 標準偏差 | std() |
| 最小値 | min() |
| 第 1 四分位(25%点) | quantile(0.25) |
| 中央値(50%点) | median() |
| 第 3 四分位(75%点) | quantile(0.75) |
| 最大値 | max() |
| 合計値 | sum() |
| 分散値 | var() |
| 最頻値 | mode() |
| 歪度 | skew() |
| 尖度 | kurt() |

| | |
|---|---|
| 最大値を取る項目名 | idxmax() |
| 最小値を取る項目名 | idxmin() |

Python のパッケージには、「decsribe」という便利な関数があり、上記の集計内容は「decsribe」関数を使うことで、一気に求めることができます。

例：decsribe 関数を使用して、データの概要を確認する。

```
#dfの基本統計量を表示
df.describe() .round(2)
```

| | ケーキ | カフェ | 食レポ | スイーツ | 食いしん坊 | コンビニスイーツ | ファッション | スクールメイク | プチプラ | コスメ |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 | 50.00 |
| mean | 31.66 | 19.48 | 43.32 | 42.84 | 12.38 | 16.72 | 14.16 | 5.02 | 20.04 | 21.46 |
| std | 31.95 | 18.78 | 33.59 | 34.04 | 16.49 | 19.93 | 22.70 | 9.65 | 25.72 | 24.51 |
| min | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 25% | 2.25 | 0.00 | 6.75 | 10.50 | 0.00 | 0.50 | 0.00 | 0.00 | 0.00 | 0.00 |
| 50% | 22.50 | 14.50 | 42.50 | 42.00 | 4.50 | 9.00 | 0.50 | 0.00 | 2.50 | 17.50 |
| 75% | 54.75 | 40.00 | 70.00 | 65.75 | 20.00 | 26.50 | 23.50 | 4.75 | 40.00 | 40.00 |
| max | 113.00 | 60.00 | 120.00 | 114.00 | 60.00 | 65.00 | 104.00 | 40.00 | 90.00 | 90.00 |

他に集計時には、以下の演算子も使用する場合があります。
（これらは、「decsribe」の中には入っていません。）

| | |
|---|---|
| 合計値 | sum() |
| 分散値 | var() |
| 最頻値 | mode() |
| 歪度 | skew() |
| 尖度 | kurt() |
| 最大値を取る項目名 | idxmax() |
| 最小値を取る項目名 | idxmin() |

## 可視化およびグラフ作成

■Python　可視化およびグラフ作成

Python での可視化に使われるパッケージは以下の通りです。

-Matplotlib（Python の可視化に使われる標準的なパッケージ）

-Pandas の plot 関数

（Matplotlib の wrapper、シンプルなコードで pandas.DataFrame を可視化）

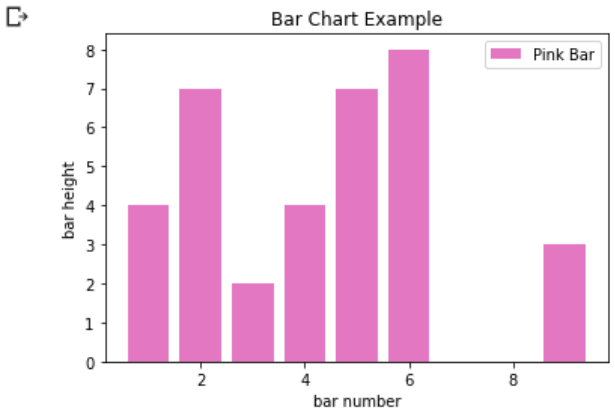
-seaborn

（Matplotlib の wrapper、統計的なデータを可視化しやすくしたパッケージでグラフを作成）

ここでは、標準的なパッケージの Matplotlib を使用してのコードの例を記載します。

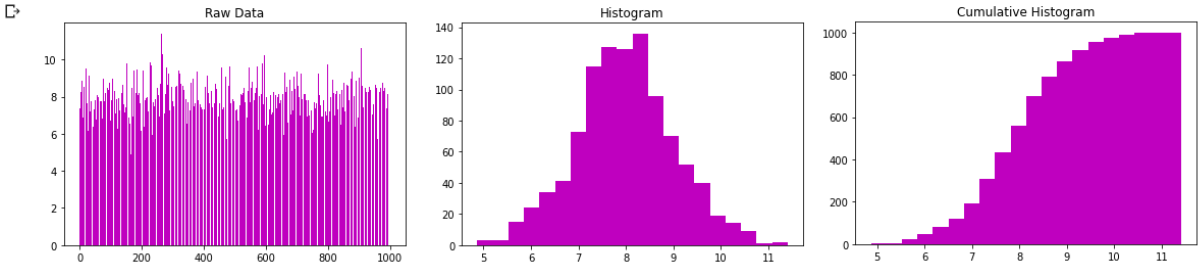
例:棒グラフ

```
import matplotlib.pyplot as plt

x1 = [1, 2, 3, 4, 5, 6, 9] #x1の値 (X軸の値
y1 = [4, 7, 2, 4, 7, 8, 3] #y1の値 (Y軸の値


# Colors: https://matplotlib.org/api/colors_api.html

plt.bar(x1, y1, label="Pink Bar", color='tab:pink') #棒グラフにする要素を指定

plt.plot() #棒グラフを表示

plt.xlabel("bar number") #x軸ラベル名の指定
plt.ylabel("bar height") #y軸ラベル名の指定
plt.title("Bar Chart Example")  #グラフのタイトルを指定
plt.legend() #凡例を表示させる
plt.show() #表示する
```



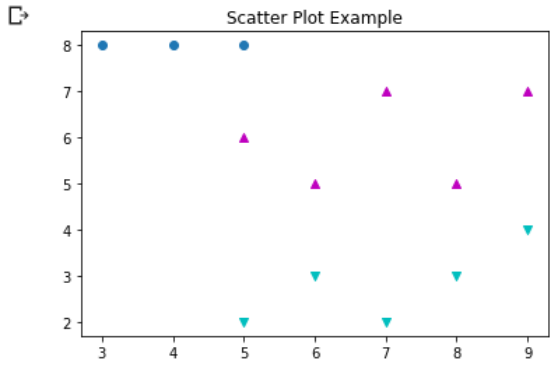
例:ヒストグラム

```
import matplotlib.pyplot as plt
import numpy as np

# Use numpy to generate a bunch of random data in a bell curve around 8.
n = 8 + np.random.randn(1000)

m = [m for m in range(len(n))]
plt.bar(m, n, color='m')
plt.title("Raw Data")
plt.show()

plt.hist(n, bins=20, color='m')
plt.title("Histogram")
plt.show()

plt.hist(n, cumulative=True, bins=20, color='m')
plt.title("Cumulative Histogram")
plt.show()
```



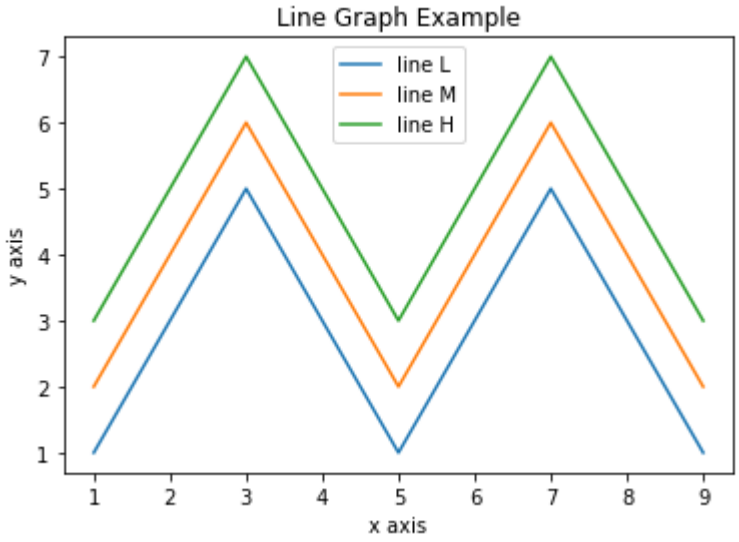

例:散布図

```
import matplotlib.pyplot as plt

x1 = [3, 4, 5]
y1 = [8, 8, 8]

x2 = [5, 6, 7, 8, 9]
y2 = [2, 3, 2, 3, 4]
y3 = [6, 5, 7, 5, 7]

# Markers: https://matplotlib.org/api/markers_api.html

plt.scatter(x1, y1)
plt.scatter(x2, y2, marker='v', color='c')
plt.scatter(x2, y3, marker='^', color='m')
plt.title('Scatter Plot Example')
plt.show()
```

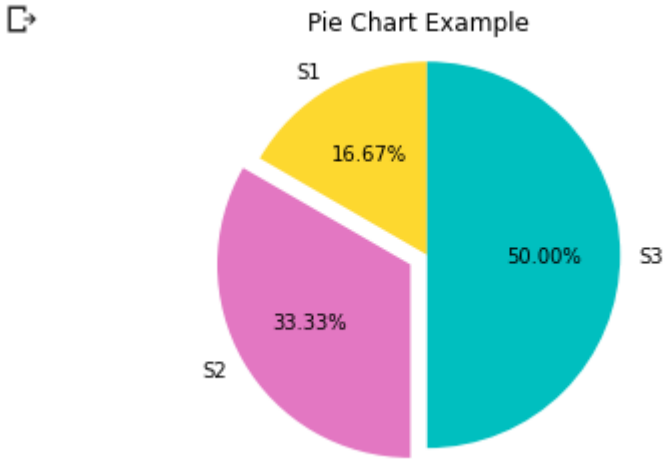



例:折れ線グラフ

```
import matplotlib.pyplot as plt

x  = [1, 2, 3, 4, 5, 6, 7, 8, 9]   # xの値   (X軸の値
y1 = [1, 3, 5, 3, 1, 3, 5, 3, 1]   # y1の値 (Y軸の値
y2 = [2, 4, 6, 4, 2, 4, 6, 4, 2]   # y2の値 (Y軸の値
y3 = [3, 5, 7, 5, 3, 5, 7, 5, 3]   # y3の値 (Y軸の値
plt.plot(x, y1, label="line L")    # xとy1を凡例:「line L」としてプロットする
plt.plot(x, y2, label="line M")    # xとy2を凡例:「line M」としてプロットする
plt.plot(x, y3, label="line H")    # xとy3を凡例:「line H」としてプロットする

plt.xlabel("x axis")   # x軸ラベル名の指定
plt.ylabel("y axis")   # y軸ラベル名の指定
plt.title("Line Graph Example")   # グラフのタイトルを指定
plt.legend() #  凡例を表示させる
plt.show() # 表示する
```



```
import matplotlib.pyplot as plt

labels = 'S1', 'S2', 'S3'
sections = [10, 20, 30]
colors = ['#FDD82F', 'tab:pink', 'c']

plt.pie(sections, labels=labels, colors=colors,
        startangle=90,
        explode = (0, 0.1, 0),
        autopct = '%1.2f%%')

plt.axis('equal') # Try commenting this out.
plt.title('Pie Chart Example')
plt.show()
```




Matplotlib のグラフを指定する主な関数

| 描画の種類 | 関数名 |
|---|---|
| 棒グラフ | plt.bar |
| ヒストグラム | plt.hist |
| 散布図 | plt.scatter |
| 折れ線グラフ | plt.plot |
| 円グラフ | plt.pie |