

The Kyoto College of
Graduate Studies
for Informatics

kcg.edu

コンピュータプログラミング概論

2021年秋第3回 eラーニング資料

安 平勲

h_an@kcg.ac.jp

Pythonプログラムの構成要素

- 関数は複数の命令文からなる一連の処理。段落に対応

プログラミング言語 ; Python

自然言語

数値, 文字列, 演算子, 変数

名詞, 動詞などの品詞

文(Code), 命令文 (statement)

文, 文章

関数, メソッド, クラス

段落

モジュール

○○.py

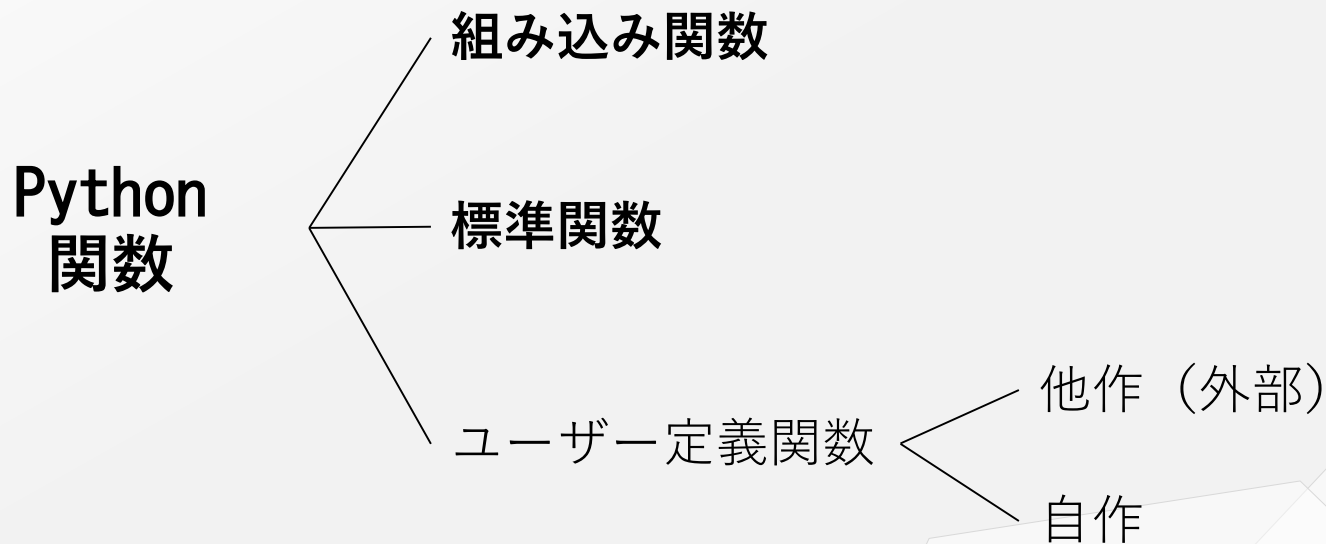
節, 章

パッケージ

論文, 本



- Pythonの関数には色々な種類ある



組み込み (built-in) 関数

- 組み込み関数はいつでも呼び出せるPythonの関数
- `print()`, `input()`, `exit()`, `type()`, `int()`, `float()`, `str()`, . . .

関数名 (引数, 引数, . . .)



引数argumentがなくても丸括弧 **()** は**省略不可!**

入出力に使う組み込み関数の例

組み込み関数	説明
<code>print(値, 値, ..., sep="文字列", end="文字列")</code>	引数の値を順に出力する。 sep の文字列を区切りに、 end の文字列を文末に追加して表示
<code>input("文字列")</code>	コンソールからの入力を受け取る。引数の文字列はプロンプトとしてコンソールに表示
<code>type(引数)</code>	引数の型を返す
<code>open()</code>	テキストファイルを開く
...	


関数と戻り値


- 関数には**戻り値**return valueがあり／なしの2タイプがある

戻り値	関数の例	説明
あり	<code>strings = input("文字列")</code>	コンソールからの入力文字列が 戻り値として返る 。戻り値を変数で受け取る
なし	<code>print(値)</code>	引数の値（文字列または数値など）をコンソールに出力する * 戻り値はない

`print(値, 値, . . . , sep="文字列", end="文字列")`

- Spyderを起動して、`print`関数の引数を確認しよう

 `print_func.py` ×

C: > Users > KCG > Desktop >  `print_func.py` > [e] a

1 `a = 100`

2 `b = 200`

3 `c = 300`

4 `sum = a + b + c`

5 `print(a,b,c,sep=', ',end=' . . . 合計')` # 区切りは、文末は . . .

6 `print(sum)`



`'sep='`, `'end='`をキーワード引数, `'a'`, `'b'`, `'c'`を位置引数と呼ぶ

input("文字列")

- input関数の戻り値を確認しよう

input_func.py ×

C: > Users > KCG > Desktop > input_func.py > ...

```
1 free_word = input('好きな言葉を入力: ') # '好きな言葉を入力'がプロンプト
2 print(free_word)                        # 入力文字列を出力
```


演習：input関数とprint関数

- 前回Jupyterで作成したBMIを計算するプログラム（下記）を，身長と体重を**input**関数でキーボードから入力して計算するように改修しよう

※入力値の型変換（文字列⇒数値）を忘れないように！

自分のBMIを計算してみよう

```
▶ # 自分のBMIを計算
height = 1.70          # 身長
weight = 68            # 体重
bmi = weight / height ** 2 # 計算式
print(bmi)             # 25以上は肥満
```

23.529411764705884

算術計算に使う組み込み関数の例

関数	説明
<code>max(数値1,数値2,・・・)</code>	引数のうち、最大の数値を返す
<code>min(数値1,数値2,・・・)</code>	引数のうち、最小の数値を返す
<code>abs(数値)</code>	引数の数値の絶対値を返す
<code>round(数値, 桁数)</code>	引数の数値を、指定された桁数に丸める。 桁数を省略すると、整数に丸める ※切り捨てと切り上げが同値の場合、偶数側に丸める

Python標準関数

- 標準関数は、Python**標準ライブラリ**から**モジュール**を読み込み（**import**）後に使える関数

※標準ライブラリ自体はPythonと一緒にインストールされている

読み込み方法（その1）

import **モジュール名**

使用方法（その1）

モジュール名.関数名（引数, 引数, . . .）

dot

mathモジュールの標準関数の例

関数	説明
<code>factorial(<i>n</i>)</code>	整数 <i>n</i> の階乗（数学の <i>n</i> ！）を返す
<code>exp(<i>x</i>)</code>	指数関数。 <i>e</i> の <i>x</i> 乗を返す
<code>log2(<i>x</i>)</code>	2を底とする <i>x</i> の対数を返す
<code>sqrt(<i>x</i>)</code>	<i>x</i> の平方根を返す
参考) <code>pi</code>	円周率（3.1415…） ※定数なので（）がつかない

mathモジュールの標準関数を使う（その1）

factorial_func.py ×

C: > Users > KCG > Desktop > factorial_func.py > ...

```
1 import math
2 print(math.factorial(1))    # 1の階乗
3 print(math.factorial(3))    # 3の階乗
4 print(math.factorial(5))    # 5の階乗
5 print(math.factorial(10))   # 10の階乗
6 print(math.factorial(50))   # 50の階乗
```

- **math**モジュールを**import**
- 関数**factorial**の前に'**math.**'

```
(base) C:¥Users¥KCG>python C:¥Users¥KCG¥Desktop¥factorial_func.py
```

```
1
6
120
3628800
304140932017133780436126081660647688443776415689605120000000000000
```

関数だけを読み込む方法（その2）

- モジュールから使う関数だけを読み込む

読み込み方法その2

from モジュール名 **import** 関数名 (**as** 別名)

使用方法その2

関数名 (引数, 引数, . .)

- 👉 関数名の前に付ける **‘モジュール名.’** が **不要**（その1と比較）

mathモジュールの標準関数を使う（その2）

factorial_func2.py ●

C: > Users > KCG > Desktop > factorial_func2.py > ...

```
1  from math import factorial as fac
2  print(fac(1))    # 1の階乗
3  print(fac(3))    # 3の階乗
4  print(fac(5))    # 5の階乗
5  print(fac(10))   # 10の階乗
6  print(fac(50))   # 50の階乗
```

- mathモジュールから、factorial 関数だけを読み込む
- factorialの名前が長いので、**as**で別名（**fac**）に変更

```
(base) C:¥Users¥KCG>
(base) C:¥Users¥KCG>python C:¥Users¥KCG¥Desktop¥factorial_func2.py
1
6
120
3628800
30414093201713378043612608166064768844377641568960512000000000000
```

- 結果はその1と同じ

標準ライブラリ

- os, sys – オペレーティングシステムのインターフェース
- **math**, cmath – 数学的な関数と演算
- itertools – イテレーター, ジェネレーター
- functools – 関数型プログラミング
- random – 疑似乱数の生成
- pickle – ディスクへのオブジェクトを保存, ディスクからオブジェクトを読み込む
- json, csv – JSON と CSV ファイルの読み込み
- urllib – HTTP と ウェブリクエストの処理



詳細は, [ここで確認](#)

randomモジュール；疑似乱数を生成する

- randomモジュールのrandint関数で1から10の乱数を作る

```
9 import random  
10  
11 print(random.randint(1,10))
```

または

```
8 from random import randint  
9  
10 print(randint(1,10))
```

関数の使い方の比較

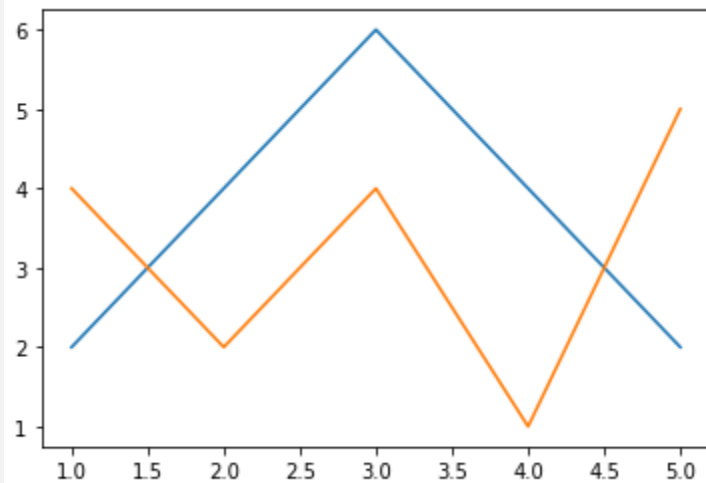
		<u>インストール</u>	<u>import</u>
Python 関数	組み込み関数	不要	不要
	標準関数	不要	必要
	ユーザー定義 関数	他作／外部	必要
		自作	(別の回でやります)

ユーザー定義関数（他作/外部）

- AIやデータ分析の分野で使われる、**matplotlib**や**numpy**は**外部**の**ユーザー定義関数**。なのでインストール要
※anacondaを使っていればmatplotlib・numpyはインストール済

```
import matplotlib.pyplot as plt
```

```
dx = (1, 2, 3, 4, 5)  
dy1 = (2, 4, 6, 4, 2)  
dy2 = (4, 2, 4, 1, 5)  
plt.plot(dx, dy1)  
plt.plot(dx, dy2)
```



ユーザー定義関数（他作/外部）

```
▶ import numpy as np          # numpyの別名はnpが慣習
A = np.array([[1,2],[3,4]])   # 行列Aの定義
B = np.array([[5,6],[7,8]])   # 行列Bの定義
print('行列の和')
print(A+B)
print('行列の積')
print(np.dot(A,B))
```

```
行列の和
[[ 6  8]
 [10 12]]
行列の積
[[19 22]
 [43 50]]
```

組み込みbuilt-inメソッドmethod

- メソッドは関数とほぼ同じ。作られ方/使い方が関数と少し違う
- メソッドは**オブジェクト**に定義された関数

オブジェクト obj

メソッド a(引数)

メソッド b(引数)

メソッド c(引数)

使用方法

dot

オブジェクト名.メソッド名 (引数, 引数, ,)
⇒ obj.a(引数) obj.b(引数) obj.c(引数)



文字列も**オブジェクト**なので多くのメソッドがbuilt_inされている

([公式ドキュメントをクリック](#))

文字列の操作メソッド

■ 大文字と小文字

```
In [7]: ▶ string = 'we think, therefore we are'
```

```
In [8]: ▶ string.capitalize() # 先頭を大文字
```

```
Out[8]: 'We think, therefore we are'
```

```
In [9]: ▶ string.title() # すべての単語の頭を大文字
```

```
Out[9]: 'We Think, Therefore We Are'
```

```
In [10]: ▶ string.upper() # すべての文字を大文字
```

```
Out[10]: 'WE THINK, THEREFORE WE ARE'
```

```
In [11]: ▶ string.lower() # すべての文字を小文字
```

```
Out[11]: 'we think, therefore we are'
```

■ 文字列.capitalize()

■ 文字列.title()

■ 文字列.upper()

■ 文字列.lower()

文字列の操作メソッド

```
In [1]: ▶ string = 'apple pie'  
        string.count('p')
```

```
Out[1]: 3
```

```
In [2]: ▶ string.find('p')
```

```
Out[2]: 1
```

```
In [3]: ▶ string.find('i')
```

```
Out[3]: 7
```

```
In [4]: ▶ string.find('y')
```

```
Out[4]: -1
```

```
In [5]: ▶ string.replace('pie','juice')
```

```
Out[5]: 'apple juice'
```

■ 文字を数える：文字列.count(文字)

■ 文字を探す：文字列.find(文字)

- ・ 見つければ語順（始めは0），
 複数あれば最初の語順
- ・ 見つからなければ-1

■ 文字を置換する：
 文字列.replace(文字, 文字)

文字列の操作メソッド

■ 文字列の長さ，分割と結合

```
In [14]: ▶ long_string = 'supercalifragilisticexpialidocious'  
len(long_string)
```

```
Out[14]: 34
```

```
In [15]: ▶ long_string1 = 'supercali,fragilisticexpiali,docious'  
long_string1.split(',')
```

```
Out[15]: ['supercali', 'fragilisticexpiali', 'docious']
```

```
In [16]: ▶ long_string_split = long_string1.split(',')  
'.'.join(long_string_split)
```

```
Out[16]: 'supercali,fragilisticexpiali,docious'
```

➤ カンマ区切りの文字列を配列に変換

➤ 配列をカンマ区切りの文字列に変換



len()は組み込み関数，split()とjoin() はメソッド

文字列の操作メソッド

■ 文字列に変数の値を埋め込む：format()

```
In [12]: ▶ name = '安 平勲'
          number = 'st100100'
          score = 88
          string = '氏名：[], 学籍番号：[], 点数：[]'
          print_text = string.format(name,number,score)
          print(print_text)
```

氏名：安 平勲、学籍番号：st100100、点数：88

但し、python3.6から以下の書き方が出来るようになった

```
In [13]: ▶ name = '安 平勲'
          number = 'st100100'
          score = 88
          print_text = f'氏名：{name}、学籍番号：{number}、点数：{score}'
          print(print_text)
```

氏名：安 平勲、学籍番号：st100100、点数：88

➤ 文字列の波括弧 {} に変数を埋め込むメソッド

➤ f文で波括弧 {} に変数を書ける