

第6回

関数 & 関数定義

早大本庄 情報科 飯島 涼



今日の内容

- 関数とは
 - 関数の利用 (p.74-p.82)
 - 関数定義 / 関数の作成 (p.232-p.254)
 - 関数の利点

関数とは

入力した数字や文字に対して、処理をした結果を出力するもの



処理ごとに、関数名が割り当てられる.

関数の使い方

関数名(引数1, 引数2,, 引数n)
のように記述する.

例1: 小さい数を入力する関数

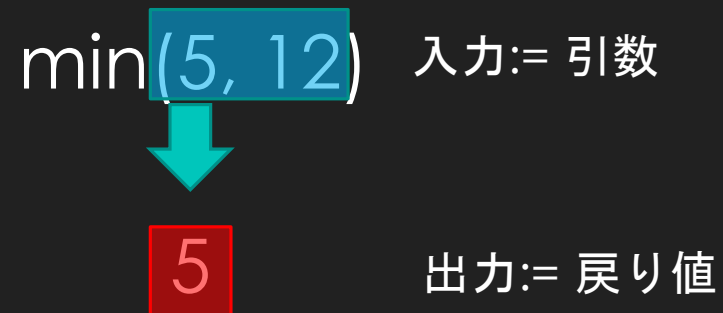
```
print(min(5, 12))
```



プログラム内で下のように置き換えられる

```
print(5)
```

=> 5が出力される.



関数の使い方

例2: abs

```
1 a = -5
2
3 print(abs(a))
4 # print(abs(-5))
5 # print(5)
6
7 #のようにプログラムが置き換えられる.
8
9 b = 5-12j
10 print(abs(b))
```

関数の使い方

例3: sin()

sin関数は, math **モジュール**の中に入っている.

いろんな関数が入ったフォルダ

```
1 import math
2
3 x = math.pi/2
4
5 print(math.sin(x))
```



1.0

モジュール名.関数名

math.sin()

math
モジュール
の中の

sin関数を使います.

文字列 / リスト に関数を適用する場合

文字列 / リストの入った変数の直後にピリオドを書き、その後に関数を記述する。

変数.関数名()

文字列を指定した区切り文字で区切ってリストにする例: `split()` 関数



```
1 # , (カンマ)区切りの文字列を分ける
2
3 month = "Jan,Feb,Mar,Apr,May,Jun,Jul,Aug,Sep,Oct,Nov,Dec"
4
5 print(month.split(","))
```



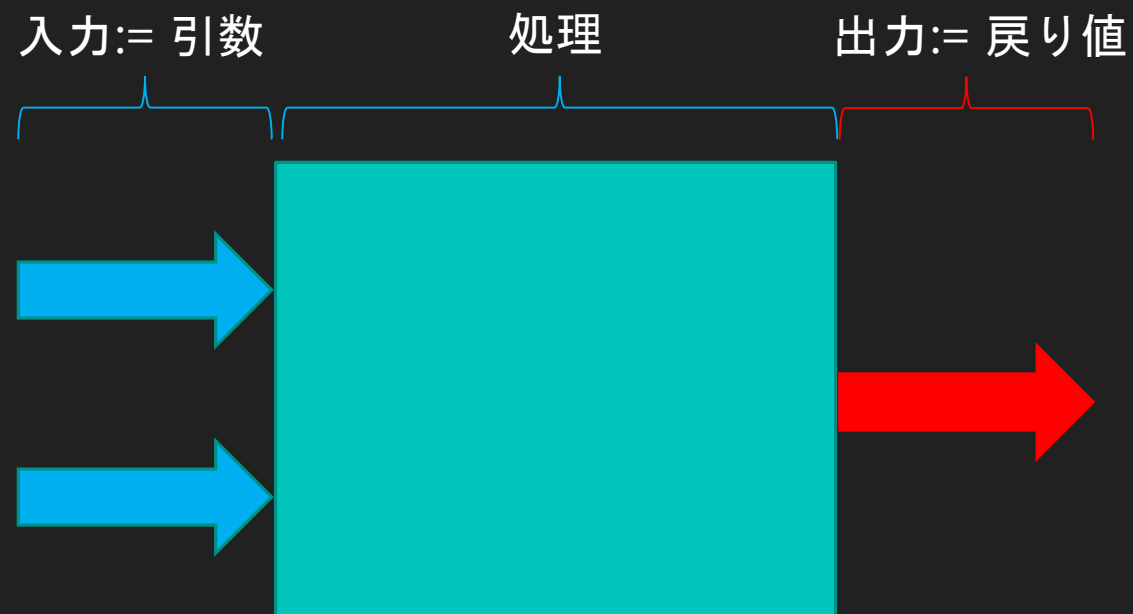
```
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

こういう機能がないかな?と思ったら

- 本で探す
- Python + document で検索する.
- Python + モジュールで検索する.
- Python + (実現したいことの単語) やり方 など調べ.
- 自分で関数を作ってしまう.

関数定義

- 関数を作成する方法
 - 自分で引数, 戻り値, 処理の内容も決める.



関数定義の方法

```
def 関数名 (引数1, 引数2, 引数3, ... .. 引数n):  
    処理1  
    処理2  
    処理3  
    ... ..  
    return 戻り値
```

関数定義の方法

```
def 関数名 (引数1, 引数2, 引数3, ... .. 引数n):
```

```
    処理1
```

```
    処理2
```

```
    処理3
```

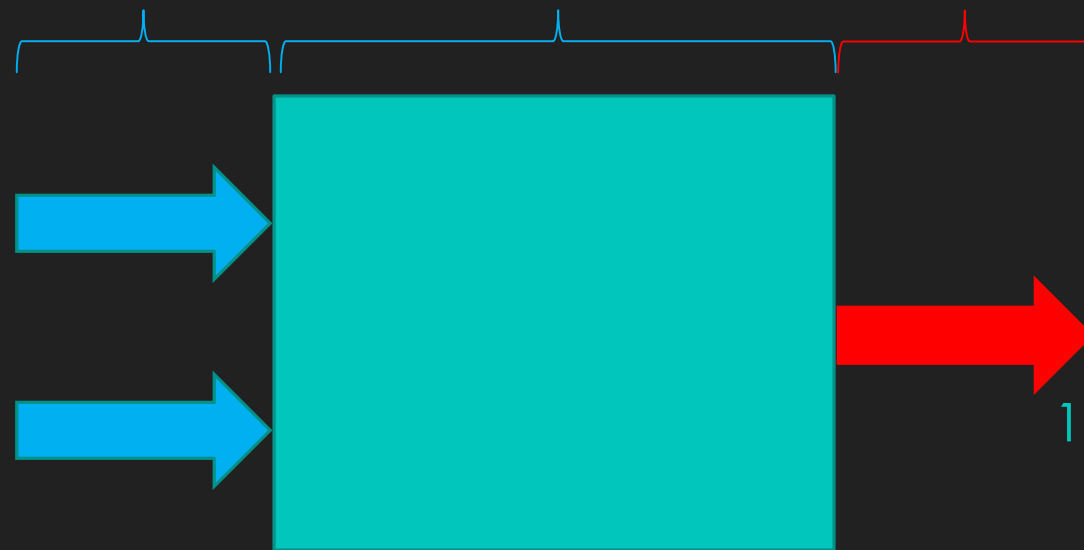
```
    ... ..
```

```
    return 戻り値
```

入力:= 引数

処理

出力:= 戻り値



関数定義

処理の部分には何を書くのか？

- 計算式
- 条件分岐 / 繰り返し
- 今まで使ってきた関数(機能)

今まで習ってきたことを組み合わせて、
自分の作りたい関数を作る。

関数定義の例 / 定義した関数の利用例

時間表記を，秒数表記に直す関数

```
[ ] 1 # 関数定義の部分
    2 def hm2s(hour, minute):
    3     second = hour * 3600 + minute * 60
    4     return second
    5
    6 # 関数の利用
    7 print(hm2s(2, 30))
    8
```

📄 9000

引数は，変数の形で書く．

関数を利用することを，
「関数呼び出し」と表記することもある．

関数定義の例 / 使用例

- 税抜き価格を，消費税込みの値段に変換する(去年の中間テストに出題)


```
1 def add_tax(price, rate):  
2     added = int( price * (1 + rate) )  
3     return added  
4  
5 print(add_tax(100, 0.1))
```

110

関数の処理の流れ

① add_tax関数を呼び出す (5行目)と、関数定義の先頭に処理が移動する.

```
1 def add_tax(price, rate):  
2     added = int( price * (1 + rate) )  
3     return added  
4  
5 print(add_tax(100, 0.1))
```



②引数の部分に書かれた変数に、5行目で指定した引数の値が代入される.

```
1 def add_tax(price=100, rate=0.1):  
2     added = int( price * (1 + rate) )  
3     return added  
4  
5 print(add_tax(100, 0.1))
```

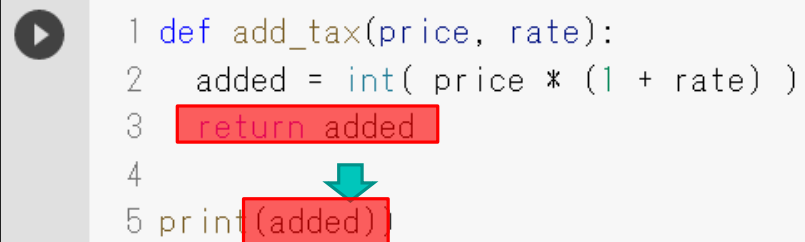
```
price = 100  
rate = 0.1
```

関数の処理の流れ

③ 代入されたprice, rateを使って処理が行われる (2行目).

```
1 def add_tax(price, rate):  
2     added = int( price * (1 + rate) )  
3     return added  
4  
5 print(add_tax(100, 0.1))
```

④関数を呼び出した場所を, 戻り値の値で置き換える.



The diagram shows the execution flow of the code. A play button icon is on the left. The code is:
1 def add_tax(price, rate):
2 added = int(price * (1 + rate))
3 return added
4
5 print(add_tax(100, 0.1))
A red box highlights the 'return added' statement on line 3. A blue arrow points from this box down to a red box on line 5, which highlights 'added' inside the 'print' function call. This illustrates how the return value is substituted into the call site.

```
1 def add_tax(price, rate):  
2     added = int( price * (1 + rate) )  
3     return added  
4  
5 print(add_tax(100, 0.1))
```

110

⑤ 置き換えられた値が実行される.

関数定義の例 / 定義した関数の利用例

コインを投げた結果を出す関数
(去年のテストで類題を出題)

```
1 import random
2 # コインを投げる関数
3 def coin():
4     x = random.random()
5     if (x > 0.5):
6         return "H"
7     else:
8         return "T"
9
10 # コインを投げる
11 print(coin())
12
```

T

表が出る確率を操作できる関数

```
1 def coin2(rate):
2     x = random.random()
3     if (x > rate):
4         return "H"
5     else:
6         return "T"
7
8 print(coin2(0.2))
```

H

実用的な例

メールのテンプレート文

```
▶ 1 def mail_temp(name, txt):  
2     return name + txt # 文字の足し算は単純に2つを連結させる  
3  
4 name_list = ["aoki", "iiijima", "ueda", "endo"]  
5 txt = "様, いつもご利用いただきありがとうございます."  
6  
7 for name in name_list:  
8     print(mail_temp(name, txt))  
9  
10
```

↪ aoki様, いつもご利用いただきありがとうございます.
iiijima様, いつもご利用いただきありがとうございます.
ueda様, いつもご利用いただきありがとうございます.
endo様, いつもご利用いただきありがとうございます.

関数定義を利用するメリット

- コードの再利用
 - 同じようなコードを何度も書かなくていい.
 - 繰り返し文よりも自由度の高い繰り返し利用ができる.
- コードの視認性
 - わかりやすいコードになる
 - 機能ごとに名前を付けることで, どの行で何をしているのかがわかりやすくなる.
 - うまく名前を決めて処理を書くと, 時間が経ってから読み返しても内容がわかる.

自由課題

- 自分の持っているテキストの、関数定義の例を、意味を理解しながらcolaboratory上で実装してみてください。(詳細！Python 3入門ノートでは、p.232-242, 入門Python3では、p.109-115)
- さいころを投げる関数dice(N)を実装してください。引数Nは、さいころの面の数を表します。
- 整数xの絶対値を求める関数myabs(x)を作成してください。
 - 条件: 関数定義の中で、absを使用してはいけない
- mathモジュール内に含まれる関数を、授業で取り上げたもの以外で3つ、プログラムで例を挙げながら説明してください。

成績の付け方

貢献
ポイント

ポイントの平均が
112だった場合

112 pt → 成績の 74-75点に換算
最高pt → 成績の95点前後に加算
最低pt → 未定（貢献ポイントの割合に応じて）

基礎
ポイント

特に貢献ポイントに対して動きがない場合は、テストの素点で付ける形になります。

ポイントの種類例

基礎ポイント

- 中間テスト 最高点を20-25 ptとする。(参加必須, 中止の場合配分変更あり)
- 期末テスト 最高点を40-50 ptとする。(参加必須, 中止の場合配分変更あり)
- 自由課題 1つの提出につき0.5-2 pt(出来具合に応じて, 自由)

ポイントの種類例

貢献ポイント (多すぎる場合は上限あり, 自由)

- 授業・教科書で示された例以外のプログラムの共有 (slackの#programming_2020へ) 各週ごとに+1-3pt
- 自由課題の解説pdf or 動画 各週ごとに+1-3pt (原則授業期間がすぎてから)
- プログラミング周辺で勉強した内容について解説した テキストをSlackなどで配布 4-6pt (ブログ公開したURLの送信で+2pt, 授業で扱っていない内容でも可, 先生の説明で簡潔過ぎた部分の説明など)

ポイントの種類例

- プログラミング周辺で勉強した内容について解説した音声・動画のクラスへの共有 (YouTubeなどの限定公開設定を利用, 顔は写ってなくても可. 授業で扱っていない内容でも可, 先生の説明で簡潔過ぎた部分の説明など) => 内容に応じて +5-15pt
- AtCoder のアカウント名公開 +1pt
- AtCoderへの参加 1大会につき+1pt
- 解いたAtCoder過去問の解説の共有 問題のレベル・説明の充実度に応じて +0.2-0.5点 (A問題を除く)
- 授業に参加する人に内容や問題について質問 => 議論した内容をスクショして先生にslackのDMなどで送る (質問した人, された人にそれぞれ0.5 pt, 質問した人, された人の組み合わせが異なるごとに新しく加算される.)

中間テストについて

- 日時: 6/5 Fri 11:20-13:10 (本来の授業時間+提出用で数分伸ばしています.)
- 内容: 5/28までに配信する動画の内容
- 方法: Course Navi (Moodle のテストの機能にバグが多いため)
- 許可: 任意の書籍の閲覧, 任意のWebページの閲覧, 自分で作ったまとめノートや過去に書いたプログラムの閲覧, 作成した目次の閲覧
- 禁止: 他人と相談できる一切の手段 (カンニングとみなされる行為. 相談して書いたことが発覚した場合は, 相談を受けて教えた側も0点となる. 教えずに相談を受けたことを先生に知らせた場合は, 相談したもののみ0点となる.)
- (Webや本, 自分のノートなどはいくらでも見ていいが, 家族と相談, 友達と相談, コードの共有, 質問サイトへの投稿は禁止. この授業に限って, 見たサイトのコードをコピーすることも認められているが, 間違えていた場合は, 減点となる.)

中間テストの注意

- 当該時間に参加ができない場合、遅れての参加となりそうな場合は、参加をしないでください.
- 参加をしない場合は開始前にご連絡ください.
 - (開始後の連絡は欠席と同様の扱いになる。再受験はできるが、7-8割が成績の扱い。)
- 後々傾向の全く異なる問題を作成して、学校に登校できるようになったらの対応となります.
 - テスト期間とかぶる可能性あり
- スマホかスマートフォンどちらも所持していない場合も同様です.