プログラミング（Python）

第２回　数式とリスト / モジュールの使い方

早稲田大学本庄高等学院　2020年度版

飯島涼

復習

前回のキーワード

- プログラミング
- Python
- ソフトウェア・ハードウェア
- 変数
- 代入

代入

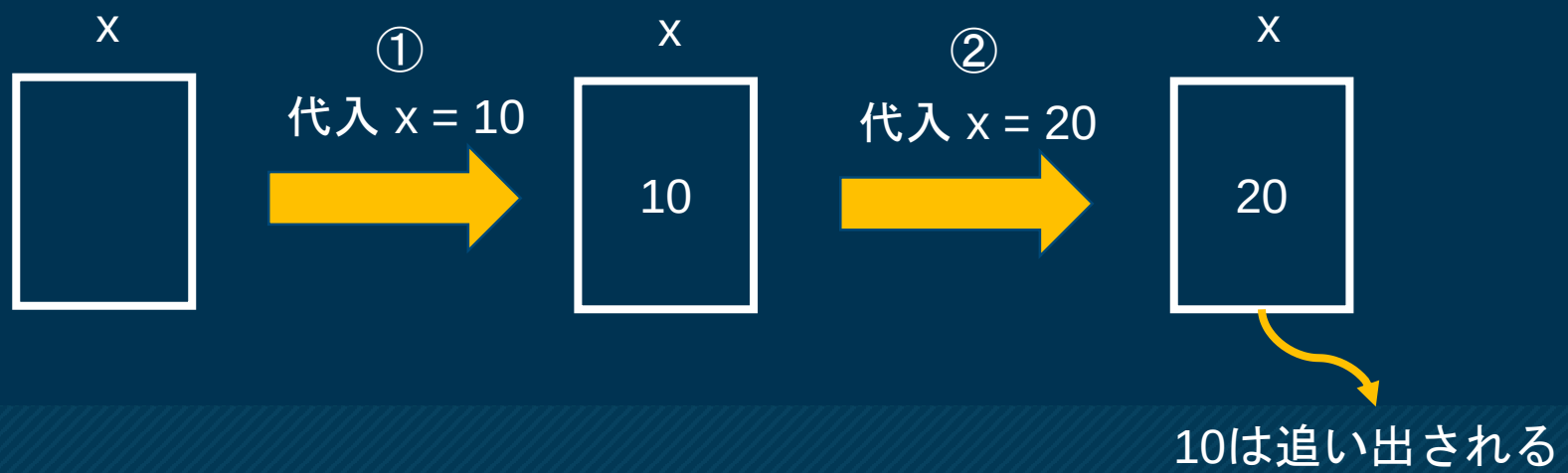
- 箱の中に具体的な数字や文字を入れること

代入しなおしたらどうなるか

- 変数の中身が入れ替わる

① x = 10
② x = 20

上から順に実行



エラーが出たら

- エラー文を読む
  - 英語を読んでください、どうしてもわからなければgoogle翻訳など使ってください。

- ググる
  - デバッグ
  - エラー
  - Qiita

  などの単語と合わせてエラー文を検索してください。

今日の内容

- 式と変数
  - 式の計算
  - 変数と式
  - 式と代入
- データ構造
  - リスト

式と代入

- 式の計算
- 変数を使った式
- 数学とプログラミングの違い

(グラフの表示は後日)

式の計算

- 計算自体は数学と同じようにできる
  - いくつか数学での使われ方と異なるポイントがあるので整理する
  - 数値計算、シミュレーションに使う
  - もちろん数学でこれまで習ってきたものを再現することもできる

演算子の種類

演算子	意味	例
+	足す	a + b
-	引く	a - b
*(アスタリスク)	かける	a * b
/	割る	a / b
%	あまり	a % b
**	べき乗	a ** b (aのb乗)

変数の計算と代入

- 変数を使った計算結果を代入することができる
  - 4行目:
    xを5に置き換え ＝>
    5 ＊ 3 を計算 ＝>
    y に計算結果(15)を代入
  - 8行目:
    計算結果を代入 (前の値は追い出される)

```
1 # 変数の計算と代入
2
3 x = 5
4 y = x * 3
5 print("1度目の代入")
6 print(y)
7
8 y = x / 3
9 print("2度目の代入")
10 print(y)
```

```
1度目の代入
15
2度目の代入
1.6666666666666667
```

数学の代入とプログラミングの代入

- 「変数への代入」のプログラムを書かないまま、数学と同じような式を書いたらどうなるでしょうか？ その結果からわかることは何でしょうか？
- 以下のプログラムはなぜエラーが出てしまったのでしょうか？

```
1 x
2 y = 2x
```

```
  File "<ipython-input-2-603de413fd57>", line 2
    y = 2x
         ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
1 x
2 y = 2*x
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-d3dd8f1e3df1> in <module>()
----> 1 x
      2 y = 2*x

NameError: name 'x' is not defined
```

代入の原則

- プログラミング
  - 変数を計算、表示に使う場合は、事前に代入をしている必要がある。

- 数学
  - まだわからない数字を変数とおいて、式変形したりできる。

数字、文字の呼び名

- 数字
  - 整数: 5, 8, -12, -256
  - 浮動小数点数: 1.666, 2.3. -0.901
  - その他: 2進数、8進数、16進数 (1年の復習)
- 文字
  - 文字列: “Hello, world”, “Ryo Iijima”, “a”

リスト

- 複数の値をまとめて扱えるしくみ

リストの例

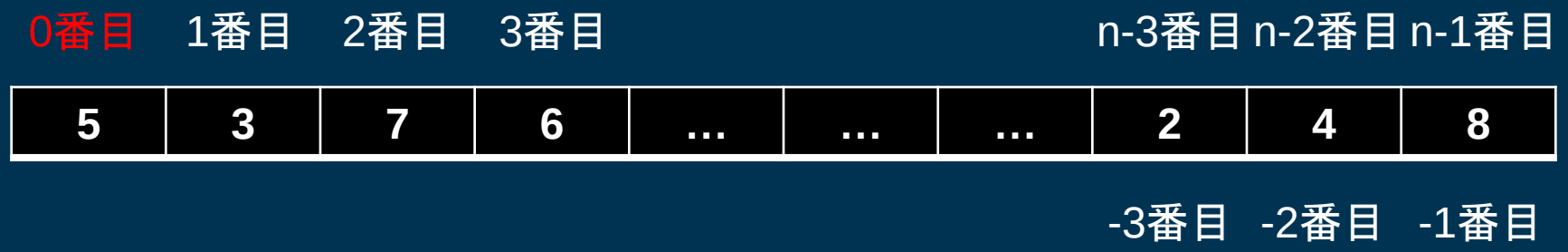
5	3	7	6	9	9	1	2	4	8

aoki	iijima	ueno	kibe	sasaki	takano	nitta	pizza	manabe

リスト

- リストの要素番号

要素番号の例 (nこの要素があるものとします)



リスト

- 代入

```
1 # リストの代入
2 L = [5, 2, 1, 4, 6, 1, 9, 2]
3 name_list = ["aoki", "iijima", "kato", "tanaka", "watanabe"]
```

- 表示

```
5 # リストの表示
6 print(L)
7 print(L[2]) #2番目のものを表示（1番前は0番目として数える）
8 print(L[2 : 5]) #2番目から4番目までを表示
9 print(L[-1]) # -1は一番うしろ  -2は後ろから2番目
```

```
[5, 2, 1, 4, 6, 1, 9, 2]
1
[1, 4, 6]
2
```

range(開始数、終了数、ステップ)

- 等差数列を作るための機能

```
1 L = list(range(5))
2 print(L)
3 L = list(range(2, 6))
4 print(L)
5 L = list(range(2, 20, 2)) #2から20の手前まで、2を足していくlist
6 print(L)
```

```
[0, 1, 2, 3, 4]
[2, 3, 4, 5]
[2, 4, 6, 8, 10, 12, 14, 16, 18]
```

モジュール

- プログラムを便利に書くための機能が詰まっているフォルダ

import [モジュール名] でモジュールを呼び出す
import [モジュール名] as [モジュール名を省略した書き方(自分で好きに決めていい)]

例)

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

モジュールの使い方

[モジュール名].機能名()

- mathモジュール
  math.sqrt(4)　　$= \sqrt{4}$

  => mathというモジュールの.sqrt(平方根)という機能を使います　ということ

```
1 # math モジュール
2 import math
3
4 #平方根
5 x = math.sqrt(4) #mathというモジュールに入ったsqrtという機能を使いますということ
6 print(x)
```

```
2.0
```

numpy

- ベクトルや行列の計算をするためのモジュール
  - arange(1, 2, 0.1)
    - rangeのベクトル版
  - array([1, 3, 5, 4])
    - リストをベクトルの計算に使えるようにする機能

```
1 # numpy モジュール
2 import numpy as np
3
4 x = np.arange(1, 2, 0.1)
5 print(x)
6 y = np.array([2, 3, 5, 1])
7 print(y)
```

```
[1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9]
[2 3 5 1]
```

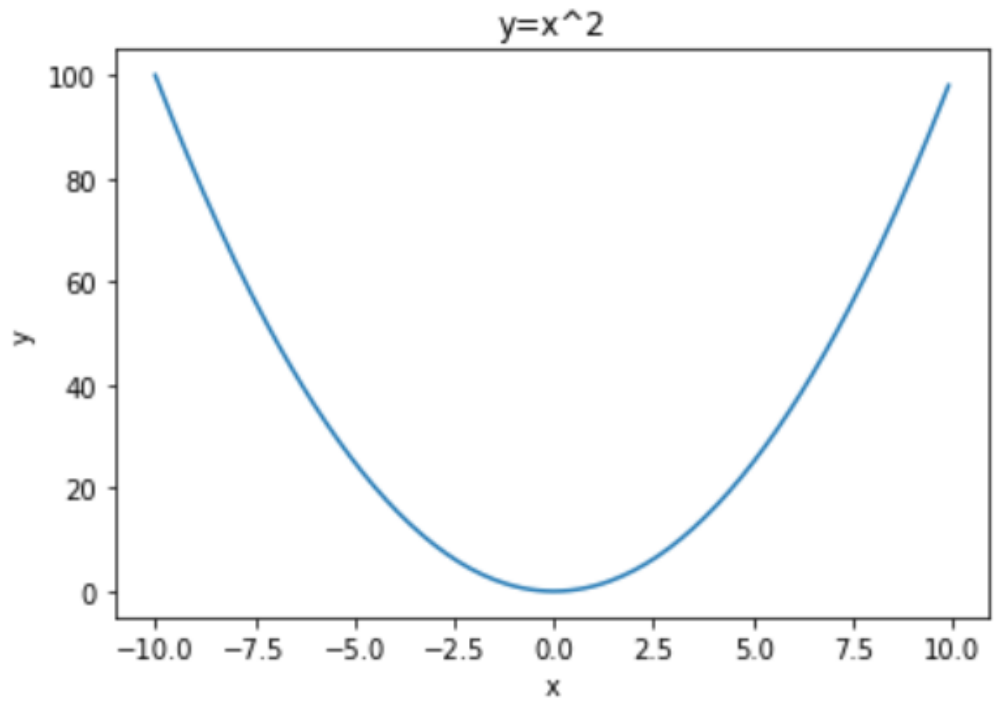
グラフの表示

- とりあえず使ってみる

```
1 # 数学のグラフ表示
2
3 import matplotlib.pyplot as plt
4 import numpy as np
5
6 x = np.arange(-10, 10, 0.1) # rangeと使い方は同じ，少数の指定ができる
7 y = x**2 #リストのブロードキャスティング
8
9 plt.plot(x, y)
10 plt.show()
```

タイトルや軸ラベルの表示

```
 9 plt.plot(x, y)
10 plt.title("y=x^2")
11 plt.xlabel("x")
12 plt.ylabel("y")
13 plt.show()
```



今日やったこと

- 数式
  - 変数と式を組み合わせて使う
  - 代入と式を組み合わせて使う
- リスト
  - データの連なりを1つの変数で使う
- モジュール
  - 便利な機能を使う方法
    - mathモジュール
    - numpyモジュール
    - matplotlibモジュール

自由課題

1. 自分の好きな食べ物のリストが代入された変数　foods を作成し，表示するプログラムを書いてください.

2. 初項 1，等差 4　となる数列のリストを作成してください.　無限に続くため，リストの要素数は10個でOKです.

3. 2. で作成したリストの平均を求めるプログラムをモジュールを用いて作成してください.

    np.mean([リスト]) と書くと，リスト内の数字の平均が求められます.

自由課題【応用】

グラフの表示方法のページ(p.22-23)にしたがって，sin，cos関数を表示させるプログラムを書いてください.

- mathモジュールの sin(), cos()という機能を使います.

興味を持った人向け（課題ではない）

- 1,2年でやったような数式を入れたらどうなるか?
  - 定義域が限定されていたらどうなる？
  - cos, sinのグラフは表示できる? それぞれを足したグラフを作るとどうなる?
- なぜベクトルを計算式の中に入れてもエラーにならなかったのか?
  - yの中身はどのようになっているか?
  - ワード: ブロードキャスト計算

⇒ 気づいたことがあれば、次の授業の最初に説明して下さい。
※ 誰にでもわかるように説明する

先にやっている人向けの情報

- まだ関数という概念をプログラミングの中ではやっていないため、[機能]という言葉にして説明しています。ユーザー定義関数をやった後にもう一度説明しなおします。

- 使い方の部分は極力少なめに説明して(すぐに調べられそうな内容)、アルゴリズムや考え方に重点を置いてやっていくつもりです。いちいち記憶する必要はありません。
  - 理解していれば大丈夫です。
    - 理解しているかの基準として、「何もみずに説明できるかどうか?」を試すことをおすすめします。
  - 何かプログラムがすぐにでも書きたいという場合はいろいろアドバイスができます。
    - ハードソフト問わずなんでも聞いてください。
    - 大会や競技、コンテストの情報は随時アナウンスします。