今日の流れ

- 第4回
- これまでの復習
- スコープ
- Webスクレイピングとは

2

これまでの復習

3

これまでの復習

- 大前提
  - Pythonは1行ずつ実行する
    インタープリタ(スクリプト)型言語
  - 実行コマンドは
    `$ python (パス/ファイル名).py`
  - 意味のまとまりを**インデント**で
    示す（C言語の { } と同じ）

4

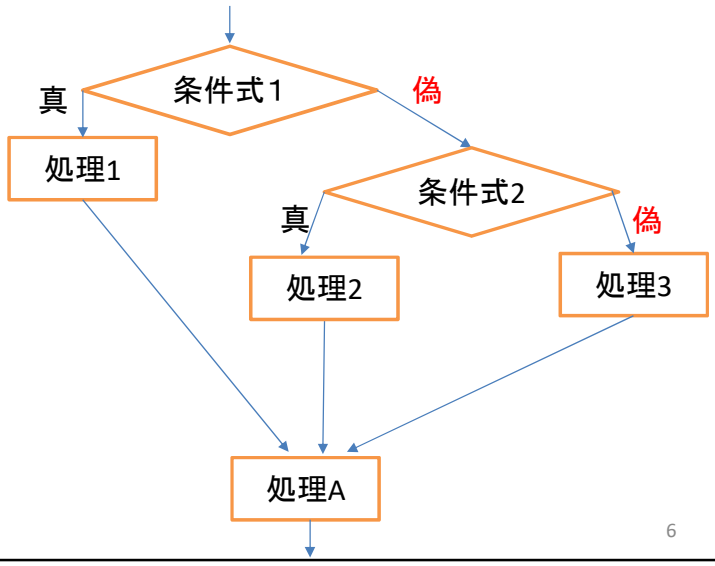
復習 – if文

条件分岐 **if文**

```
if(条件式1):
    処理1
elif(条件式2):
    処理2
else:
    処理3
処理A
```

5

復習 – if文



6

復習 – 条件式

- A == B … AとBの**値**が等しい
- A >= B … A ≧ B
- A <= B … A ≦ B
- A < B
- A > B

  同じ意味
- A != B … AとBの値が等しくない
  (A ≠ B)

- (A is B … AとBが同じ**オブジェクト**)
  - 例えば type(“hello”) is string

7

復習 – 条件式

- かつ

```
(条件式1) and (条件式2)
```

- または

```
(条件式1) or (条件式2)
```

8

復習 – データ構造

- list … 順序付けられた要素の集まり
  [要素1,要素2, …]
- tuple … 変更できないlist
  (要素1,要素2, …)
- dict … キーを指定して値を得る
  順序付けられていない
  {キー1：値1,キー2：値2, …}

9

復習 – データ構造

- 順序付けられた要素の呼び出し
  print(a[0]) … 1 が出力
  a[0] = 1 … 1 を代入
- index(添え字) は0から始まります

$$a_0 = 1$$

- dictの要素の呼び出し
  b[キー] = 値 … 1 を代入

10

復習 – スライス

- listなどの順序付けられたデータ構造の一部を得る
- a[始点: 終点: スライスの増分]
- 例えばa[s:t:k]というスライスの場合

$$a_s, a_{s+k}, a_{s+2k}, \ldots, a_{s+nk} (s + nk < t)$$

が順番に得られます

11

復習 – for文

- ループを書く文 **for文**

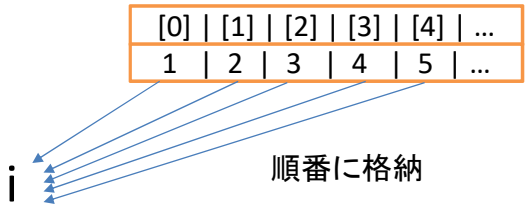
```
for 変数 in iterableなデータ構造:
    処理
```

- iterableなデータ構造… 値を順番に取り出せるデータ構造のこと
(list, tuple, dict, set, range, 文字列)

12

復習 – for文

- 変数(ここではiとする) の中にlistの中身が順番に入れられていきます
- 1周目では i = a[0],
  2周目では i = a[1], となります



13

復習 – range

- listの中身として順番に並んだ数列が欲しい場合(ただn回ループを回したい)

```
a = [0, 1, 2, 3, 4]
for i in a:
```

の代わりに

```
for i in range(5):
```

と書けます

14

復習 – while文

- ループを書く文 **while文**

```
while(条件式):
    処理
```

- 条件式が満たされている限りずっと
  ループし続ける
- 無限ループしないように注意！

15

復習 – 関数の宣言

- 処理や計算をひとまとまりにしたもの
  関数

```
def 関数名(引数1, …):
    処理
    return 返り値
```

- 引数: 関数内で利用するために、
  外部から受け取る値 (省略可)
- 返り値: 関数の呼び出し元で利用して
  もらうために返す値 (省略可)

16

復習 – 関数の呼び出し

- 返り値のある関数を呼び出す

```
変数 = 関数名(引数1, …)
```

- 返り値のない関数(処理のまとまり)を呼び出す

```
関数名(引数1, …)
```

- 引数がない場合、

```
関数名()
```

17

補足 – 関数

- 関数は呼び出す前の行で宣言されている必要がある
- 関数内で、直接関数外の変数を書き換えることは**できない**
(スコープ という概念です: 後述)
- 引数、返り値を利用しましょう

18

補足 – 属性

- aというlistの末尾に値を追加する…
  - a.append(追加したい値)
- listのある位置に値を挿入する…
  - a.insert(挿入したい位置の直後のindex, 追加したい値)
- 後ろにドット(.)を付けて書く、変数やデータ構造を編集する関数などのことを**属性(アトリビュート)**という

19

復習 – モジュール

- 関数や定数がまとめられたものを**モジュール**という
- モジュールを使いたい場合は

```
import モジュール名
import モジュール名 as 別名
```

- モジュールに含まれる関数、定数を呼び出す際は

```
モジュール名.関数(引数1, 引数2, …)
モジュール名.定数
```

20

復習 – pip

- 公開されたモジュールを簡単に
  インストールするには**pip**を使う
- `$ pip install 欲しいモジュール名`

21

スコープ

22

スコープ

- 宣言した変数、関数は呼び出せる範囲が定められている
- これを**スコープ**(可視範囲)という

23

問題です

```
def modify_a():
    a = 10

a = 0
modify_a()
print(a)
```

- なんと出力されるでしょうか？

24

問題です

```
def modify_a():
    a = 10

a = 0
modify_a()
print(a)
```

output: 0

- modify_a() という関数では
  aは書き換わっていないことになります

25

問題です

```
def modify_a():
    a = 10

a = 0
modify_a()
print(a)
```

この中でのみ有効

- modify_a()の中の a(=10) と、外にある a(=0) は別物です

26

スコープ

- 宣言された変数、関数はその後ろの行で有効
  - そのため関数の定義は、呼び出される前に書く必要があります
- 関数内で定義された変数、関数は、その中でのみ有効（ローカル）

27

スコープ

- for文内で定義された変数は、for文の**外でも有効**です(C言語と異なる)

```
for i in range(2):
    j = 10
    print(i)
print(i,j)
```

output : 0
1
1 10

28

スコープ

- 仕様を正しく理解しないと、予期せぬ動作を招くことがあります
- とりあえず
  - 「関数内で、関数外の変数は直接変更できない」
  - 「関数は呼び出す前に定義する」

  ことは覚えておきましょう

29

問題です

```
def modify_a():
    return a

a = “hey”
print(modify_a())
```

- [高度]このコードはエラーを吐かずに実行できる？

30

問題です

```
def modify_a():
    return a

a = “hey”
print(modify_a())
```

output: hey

- 実行できます

31

問題です

```
def modify_a():
    return a

a = “hey”
print(modify_a())
```

aは定義されていない?

- [高度]関数定義内で呼び出している変数がない場合、呼び出し元のスコープにある同名の変数を探します

32

問題です

```
def modify_a():
    return a

a = “hey”
print(modify_a())
```

- [高度]関数定義内で呼び出している変数がない場合、呼び出し元のスコープにある同名の変数を探します

33

問題です

```
def modify_a():
    return a

a = “hey”
print(modify_a())
```

- [高度]同名の変数を参照するのは予期せぬ動作を招くこともあるので注意してください

34

スコープ

- [高度]Pythonにもグローバル変数はありますが、非推奨な上記述がかなり面倒なので引数や返り値で渡すのがおすすめです
- スコープの詳しい仕様 https://qiita.com/yoichi22/items/8ae2ca180407a5ad5a6d

35

やってみよう

- 今まで得たすべての知識を総動員しましょう
- 3点の座標を入力から受け取り、それを結んでできる三角形の面積を計算して表示する
- その後「続けますか？」と出力し、入力の1文字目が ‘y’ だったら入力を受け取り表示、それ以外なら三角形を図示して終了する
  プログラム Triangle.py を作成

36

やってみよう

- 三角形の描画には**matplotlib**という
  モジュールを使う
- 1. pipを使ってmatplotlibを
  インストール

37

やってみよう

- matplotlibの中には更にモジュールが
  含まれている(サブモジュールという)
- 使うのはmatplotlib.pyplotという
  モジュール
- 2. matplotlib.pyplot モジュールを
  plt という別名を付けてimport

38

やってみよう

- 使う変数

```
fig = plt.figure()
ax = fig.add_subplot(1, 1, 1)
```

- import文の次に書いてください

39

やってみよう

- グラフの軸などの設定は以下のコードで行える

```
ax.set_xlim(-10, 10) # -10<=x<=10 の範囲に設定
ax.set_ylim(-10, 10) # -10<=y<=10 の範囲に設定
ax.set_aspect('equal') # 目盛りを揃える
ax.grid() # グリッドを表示
```

- 3. 上の処理を引数、返り値なしの SetGraphGrid 関数として宣言

40

やってみよう

- 入力は以下の形式です

```
x1
y1
x2
y2
x3
y3
```

- 改行区切りで各座標が与えられます
- x座標、y座標それぞれlist x, yに格納
- list xに値を追加する関数

```
x.append(値)
```

41

やってみよう

- 入力は以下の形式です

```
x1
y1
x2
y2
x3
y3
```

- for文をうまく使おう
- 入力を小数として受け取る関数

```
x = float(input())
```

42

やってみよう

- 三角形は折れ線を繋いでプロットしてください
- 折れ線をプロットする関数

```
ax.plot([x1, x2, …], [y1, y2, …])
```

- (x1, y1), (x2, y2), …と繋いでプロットします

43

やってみよう

- 三角形は折れ線を繋いでプロットしてください
- 折れ線をプロットする関数(2行)

```
ax.plot([x1, x2, …], [y1, y2, …])
plt.show()
```

- (x1, y1), (x2, y2), …と繋いでプロットします

44

やってみよう

- 三角形の面積は前回作成した関数を利用すると良いでしょう
- 4. 引数でx座標を格納したlist, y座標を格納したlist の2つを受けとり、三角形の面積を返り値として返す関数 CalcTriangleArea(x, y) を作成

45

やってみよう

- 「続けますか？」という質問に対する返答は input() で文字列として受け取ります
- indexを利用して、先頭の文字が ‘y’ か判定してください
- ‘y’ を受け取り続ければ座標の受け取り → 出力はずっと続くので、while文をうまく利用すると良さそう？

46

やってみよう

- 最後に plt.show() でそれまでに
  プロットしたものが出力されます

47

やってみよう

```
# import文

# fig, axの宣言
# SetGraphGrid関数, CalcTriangleArea関数の宣言

# while(入力の1文字目がyなら):
  # list x, yの宣言(最初は空っぽ)
  # 入力の受け取り

  # SetGraphGrid関数の呼び出し
  # 三角形のプロット
  # CalcTriangleArea関数を呼び出し、面積を出力

  # 「続けますか？」と出力
  # 入力の判定
# plt.show()
```

48

Webスクレイピングとは

49

Webスクレイピング

- Webサイトにある情報をプログラムで取得することを**Webスクレイピング**といいます
- モジュールとしてbeautifulsoup4を使います
- pipでインストール

50

Webスクレイピング

```
import bs4
import urllib.request


html =
urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")
print(bsObj)
```

- 上のコードを実行しよう

51

Webスクレイピング

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html lang="ja" xmlns:ext_fnc="FORECAST">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-
Type"/>
<meta content="気象庁 Japan Meteorological Agency" name="Author"/>
<meta content="気象庁 Japan Meteorological Agency"
name="keywords"/>
<title>気象庁 | 天気予報
： 宮城県</title>

…
```

- HTMLファイルと呼ばれるものです

52

Webスクレイピング

- HTMLファイルと呼ばれるものです
- Webサイトのレイアウトやデータを格納しているものです
- 備え付けのブラウザ等で読めます

53

Webスクレイピング

- 気象庁から宮城の天気予報のページを取得しています
- HTMLの中で、天気予報はforecast というtableに格納されているようです
- データを読み出してみましょう

54

Webスクレイピング

```
import bs4
import urllib.request

html = urllib.request.urlopen("http://www.jma.go.jp/jp/yoho/312.html")
bsObj = bs4.BeautifulSoup(html, "html.parser")

table = bsObj.findAll("table", {"class": "forecast"})[0]
rows = table.findAll("tr")

for row in rows:
    ls = []
    for cell in row.findAll(['td', 'th']):
        text = cell.get_text()
        # 改行コードやタブの削除
        text = text.replace('¥n', '')
        text = text.replace('¥u3000', '')
        text = text.replace('¥t', '')
        text = text.replace('¥xa0', '')
        if(text):
            ls.append(text)
        print(ls)
```

Webスクレイピング

```
['東部', '降水確率', '気温予報']
…
['明日8日', '北西の風後北の風やや強く海上では後北の風強くくもり昼過ぎから夕方雨波１．５メートル後２．５メートル', '00-0610%06-1220%12-1850%18-2410%', '00-06', '10%', '06-12', '20%', '12-18', '50%', '18-24', '10%', '朝の最低日中の最高仙台3度8度石巻2度8度古川0度7度', '朝の最低', '日中の最高', '仙台', '3度', '8度', '石巻', '2度', '8度', '古川', '0度', '7度']
['00-06', '10%']
['06-12', '20%']
['12-18', '50%']
['18-24', '10%']
['朝の最低', '日中の最高']
['仙台', '3度', '8度']
['石巻', '2度', '8度']
['古川', '0度', '7度']
['明後日9日', '北西の風やや強く後西の風やや強く晴れ時々くもり波２．５メートル後１．５メートル']
```

- 天気予報のデータが取れそう!

56

Webスクレイピング

- 天気予報のデータが取れそう!
- このデータをうまく整形して、
  - 天気予報の文章
  - 降水確率
  - 気温

  を Getしよう

57

Webスクレイピング

- Webスクレイピングについてさらに調べたい人はこちら

https://qiita.com/hujuu/items/b0339404b8b0460087f9

58