Python の「きも」

前田利之（まえだ　としゆき）
maechan@hannan-u.ac.jp

阪南大学　経営情報学部

(2024.10.11)

はじめに

- この授業ではプログラミング言語として
  [　　] を全面的に使用します
- とはいえ授業時間の制約から、細かい文法の説明は厳しいので、頑張って自習して、来週 (10/06) までに少なくとも読める程度になっておいてください
- 例えばの参考 URL:
  Python 入門【初心者向けに使い方を解説】
- とはいえ、それだけでは申し訳ないので、前田的重要部分＝「きも」を少しだけ解説します

はじめに

- この授業ではプログラミング言語として python を全面的に使用します
- とはいえ授業時間の制約から、細かい文法の説明は厳しいので、頑張って自習して、来週 (10/06) までに少なくとも読める程度になっておいてください
- 例えばの参考 URL:
  Python 入門【初心者向けに使い方を解説】
- とはいえ、それだけでは申し訳ないので、前田的重要部分＝「きも」を少しだけ解説します

Python の「きも」

- [ ]なので覚えることが少ない
- [ ]
- 標準[ ]として多くの機能があらかじめ用意されている
- AI 関連を含め外部[ ]が豊富に存在している

Python の「きも」

- シンプルな構文なので覚えることが少ない
-
- 標準 として多くの機能があらかじめ用意されている
- AI 関連を含め外部 が豊富に存在している

Python の「きも」

- シンプルな構文なので覚えることが少ない
- 記述性が高い
- 標準 [　　] として多くの機能があらかじめ用意されている
- AI 関連を含め外部 [　　] が豊富に存在している

Python の「きも」

- シンプルな構文なので覚えることが少ない
- 記述性が高い
- 標準 ライブラリ として多くの機能があらかじめ用意されている
- AI 関連を含め外部 [　　] が豊富に存在している

Python の「きも」

- シンプルな構文なので覚えることが少ない
- 記述性が高い
- 標準ライブラリとして多くの機能があらかじめ用意されている
- AI 関連を含め外部ライブラリが豊富に存在している

シンプルな構文

- Python の哲学
  - 複雑であるよりも [　　] な方がよい
  - 何かをする方法は、出来るだけ [　　] （誰もが理解できる）
- ⇒ プログラマーが覚えなければならない構文規則が他の言語と比べて少ない
- ⇒ [　　] に優しい

シンプルな構文

- Python の哲学
  - 複雑であるよりも シンプル な方がよい
  - 何かをする方法は、出来るだけ [　　] （誰もが理解できる）
- ⇒ プログラマーが覚えなければならない構文規則が他の言語と比べて少ない
- ⇒ [　　] に優しい

シンプルな構文

- Python の哲学
  - 複雑であるよりも シンプル な方がよい
  - 何かをする方法は、出来るだけ １つだけ （誰もが理解できる）
- ⇒ プログラマーが覚えなければならない構文規則が他の言語と比べて少ない
- ⇒ [　　] に優しい

シンプルな構文

- Python の哲学
  - 複雑であるよりも **シンプル** な方がよい
  - 何かをする方法は、出来るだけ **１つだけ** （誰もが理解できる）
- ⇒ プログラマーが覚えなければならない構文規則が他の言語と比べて少ない
- ⇒ **初心者** に優しい

記述性の高さ

- 1 行で [　　] を記述できる
  - 前頁の [　　] と背反するようだがそうではなく、単純なパーツを組みあわせることが出来る、ということ
- ⇒ 何かの処理を行うために書かなければならないコードが少なくて済む（＝ [　　] だということ）

記述性の高さ

- 1 行で **多くの処理** を記述できる
  - 前頁の [　　] と背反するようだがそうではなく、単純なパーツを組みあわせることが出来る、ということ
- ⇒ 何かの処理を行うために書かなければならないコードが少なくて済む（＝ [　　] だということ）

記述性の高さ

- 1 行で **多くの処理** を記述できる
  - 前頁の **シンプルさ** と背反するようだがそうではなく、単純なパーツを組みあわせることが出来る、ということ
- ⇒ 何かの処理を行うために書かなければならないコードが少なくて済む（＝ [　　] だということ）

記述性の高さ

- 1 行で **多くの処理** を記述できる
  - 前頁の **シンプルさ** と背反するようだがそうではなく、単純なパーツを組みあわせることが出来る、ということ
- ⇒ 何かの処理を行うために書かなければならないコードが少なくて済む（＝ **シンプル** だということ）

１行で多くの処理を記述できる例

☆例：２重ループによる九九の表…

- 次頁のコードをGoogle Colab で新しいノートブックとして動かして result の値を確認してみてください
- Google アカウントでログインしていない場合は、ログインしてください
- copy&paste しても良いですが、インデントが無くなったり、’ (quote) が変になったりするので、変なところは手で打ち直してください

1行で多くの処理を記述できる例

```
result = []
for num1 in range(1, 10):
  tmp = []
  for num2 in range(1, 10):
    tmp.append(f'{num1 * num2:2}')
    # f'...' は書式付きの文字列
  result.append(tmp)
for row in result:
  print(', '.join(row))
```

１行で多くの処理を記述できる例

☆前頁のループは、以下のようにも書ける（本当は１行）

```
result = [[f'{num1 * num2:2}'
   for num2 in range(1, 10)]
   for num1 in range(1, 10)]
```

1行で多くの処理を記述できる例

☆ python 独特の [ ]

[式 for 変数 in イテラブルオブジェクト]

- range オブジェクト等のイテラブル（＝繰り返しの）オブジェクトから順に要素を取り出し [ ]
- そしてその変数を使った式の値をリストの [ ]
- 式には変数の値に対する [ ] も記述できる

1行で多くの処理を記述できる例

☆ python 独特の **「リスト内包表記」**
[式 for 変数 in イテラブルオブジェクト]

- range オブジェクト等のイテラブル（＝繰り返しの）オブジェクトから順に要素を取り出し
- そしてその変数を使った式の値をリストの
- 式には変数の値に対する も記述できる

1行で多くの処理を記述できる例

☆ python 独特の **「リスト内包表記」**
[式 for 変数 in イテラブルオブジェクト]

- range オブジェクト等のイテラブル（＝繰り返しの）オブジェクトから順に要素を取り出し **変数に代入**
- そしてその変数を使った式の値をリストの [　　]
- 式には変数の値に対する [　　] も記述できる

1行で多くの処理を記述できる例

☆ python 独特の **「リスト内包表記」**
[式 for 変数 in イテラブルオブジェクト]

- range オブジェクト等のイテラブル（＝繰り返しの）オブジェクトから順に要素を取り出し **変数に代入**
- そしてその変数を使った式の値をリストの **要素として追加**
- 式には変数の値に対する [　　] も記述できる

1行で多くの処理を記述できる例

☆ python 独特の **「リスト内包表記」**
[式 for 変数 in イテラブルオブジェクト]

- range オブジェクト等のイテラブル（＝繰り返しの）オブジェクトから順に要素を取り出し **変数に代入**
- そしてその変数を使った式の値をリストの **要素として追加**
- 式には変数の値に対する **演算** も記述できる

リスト内包表記の例

```
res = [x**2 for x in range(10)]
print(res) # 0-9 の二乗を出力
```

標準ライブラリ

- 数学関連
- ファイル操作およびさまざまなフォーマットのファイルの操作
- GUI（グラフィカルユーザーインタフェース）
- ネットワーク処理
- 並列処理
- …など、など

外部ライブラリ

☆ AI 技術（機械学習）関連だと…

- [ ] (AI というより数値計算用)
- [ ]
- [ ]
- [ ], [ ] (Google 謹製で、最近セットで使うことが多い)
- …など、など（どんどん増えている）

外部ライブラリ

☆ AI 技術（機械学習）関連だと…

- NumPy (AI というより数値計算用)
-
-
- , (Google 謹製で、最近セットで使うことが多い)
- …など、など（どんどん増えている）

外部ライブラリ

☆ AI 技術（機械学習）関連だと…

- NumPy (AI というより数値計算用)
- scikit-learn
-
- , (Google 謹製で、最近セットで使うことが多い)
- …など、など（どんどん増えている）

外部ライブラリ

☆ AI 技術（機械学習）関連だと…

- NumPy (AI というより数値計算用)
- scikit-learn
- Chainer
- , (Google 謹製で、最近セットで使うことが多い)
- …など、など（どんどん増えている）

外部ライブラリ

☆ AI 技術（機械学習）関連だと…

- NumPy (AI というより数値計算用)
- scikit-learn
- Chainer
- TensorFlow, [ ] (Google 謹製で、最近セットで使うことが多い)
- …など、など（どんどん増えている）

外部ライブラリ

☆ AI 技術（機械学習）関連だと…

- NumPy (AI というより数値計算用)
- scikit-learn
- Chainer
- TensorFlow, keras (Google 謹製で、最近セットで使うことが多い)
- …など、など（どんどん増えている）

NumPy の例：行列の積

```
import numpy as np # ...後述
arr1 = np.arange(4).reshape((2, 2))
print(arr1)
# [[0 1]  [2 3]] ..2x2 の行列
arr2 = np.arange(6).reshape((2, 3))
print(arr2)
# [[0 1 2]  [3 4 5]] ..2x3 の行列
print(arr1.dot(arr2))
# [[ 3  4  5]  [ 9 14 19]]
# ..dot メソッドでかけ算が出来る
```

NumPy の例：行列の積

☆前頁の計算は以下の行列積を求めていた

$$\begin{pmatrix} 0 & 1 \\ 2 & 3 \end{pmatrix}\begin{pmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{pmatrix}$$

$$= \begin{pmatrix} 0\cdot0 + 1\cdot3 & 0\cdot1 + 1\cdot4 & 0\cdot2 + 1\cdot5 \\ 2\cdot0 + 3\cdot3 & 2\cdot1 + 3\cdot4 & 2\cdot2 + 3\cdot5 \end{pmatrix} = \begin{pmatrix} 3 & 4 & 5 \\ 9 & 14 & 19 \end{pmatrix}$$

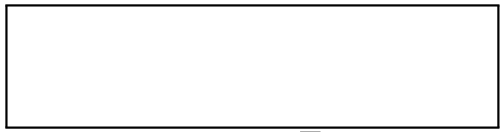
ほかの外部ライブラリの例

☆QR コード作成 (user: semi, pass: hannannipamo)

以下は抜粋 (qrcode を import している)

```
import cgi, os, sys, io, qrcode
sys.stderr = io.TextIOWrapper(
 sys.stderr.buffer, encoding='utf-8')
sys.stdout = io.TextIOWrapper(
 sys.stdout.buffer, encoding='utf-8')
def qrmk(s,fn):
  img = qrcode.make(s)
  img.save(fn)
```

前田（個人）的お気に入りな点

- ブロックは {}（中括弧）ではなく
  
  で示す
  - ちゃんとインデントしないと動かない＝
    
    ことを強制される
  - 初心者は「とりあえず動けばよい」としがちだが、それを許さず最初から
    
    書かないといけない

前田（個人）的お気に入りな点

- ブロックは {}（中括弧）ではなく
  インデント で示す
  - ちゃんとインデントしないと動かない＝
    [ ] ことを強制される
  - 初心者は「とりあえず動けばよい」としがちだが、それを許さず最初から [ ] 書かないといけない

前田（個人）的お気に入りな点

- ブロックは {}（中括弧）ではなく
  インデントで示す
  - ちゃんとインデントしないと動かない＝
    綺麗に書くことを強制される
  - 初心者は「とりあえず動けばよい」としがちだが、それを許さず最初から　　書かないといけない

前田（個人）的お気に入りな点

- ブロックは {}（中括弧）ではなく
  インデントで示す
  - ちゃんとインデントしないと動かない＝
    綺麗に書くことを強制される
  - 初心者は「とりあえず動けばよい」としがちだが、それを許さず最初から **綺麗に**書かないといけない

前田（個人）的お気に入りな点（続き）

- 前に関連して、括弧が [　　] なので読みやすい
  - C/C++/Java/... だと {} の対応が必須なのでブロックが深くなると非常に読み辛いし、抜けがち→バグにつながる
- import の as で名前を [　　] できる（これは些細、でも Java のクラスライブラリの長い綴りは閉口）

前田（個人）的お気に入りな点（続き）

- 前に関連して、括弧が 必要最小限 なので読みやすい
  - C/C++/Java/... だと {} の対応が必須なのでブロックが深くなると非常に読み辛いし、抜けがち→バグにつながる
- import の as で名前を [　　] できる（これは些細、でも Java のクラスライブラリの長い綴りは閉口）

前田（個人）的お気に入りな点（続き）

- 前に関連して、括弧が **必要最小限** なので読みやすい
  - C/C++/Java/... だと {} の対応が必須なのでブロックが深くなると非常に読み辛いし、抜けがち→バグにつながる
- import の as で名前を **短縮** できる（これは些細、でも Java のクラスライブラリの長い綴りは閉口）

C と python

☆３重ループだと…C/C++ なら

```
for (i=0;i<10;i++) {
for (j=0;j<10;j++) {
for (k=0;k<10;k++) {
sum = i*100+j*10+k; }}}
```

が、python だと

```
for i in range(10):
  for j in range(10):
    for k in range(10):
      sum = i*100+j*10+k
```

となる

前田的お気に入ってない点

- def, if, while, などの行の最後に  が必要 ⇐ 無くても構文解析的には問題なさげ
-  ⇐ elseif でいいんじゃない？
- ... （あとは思いつかないくらい、現状に満足）

前田的お気に入ってない点

- def, if, while, などの行の最後に : (colon) が必要 ⇐ 無くても構文解析的には問題なさげ
- [ ] ⇐ elseif でいいんじゃない？
- ... （あとは思いつかないくらい、現状に満足）

前田的お気に入ってない点

- def, if, while, などの行の最後に : (colon) が必要 ⇐ 無くても構文解析的には問題なさげ
- elif ⇐ elseif でいいんじゃない？
- ... （あとは思いつかないくらい、現状に満足）

前田（個人）的言語勉強法

1. まずは入門書（Webも可）をざっと読む
2. サンプルプログラム（コード）をとにかく動かす
3. サンプルプログラムをひたすら（？ある程度の量を）

   - 読むことで、目が馴染む
   - 理解できないところは調べる。最近は大抵のことは検索すると出てくるので便利
   - →  を高める

前田（個人）的言語勉強法

1. まずは入門書（Webも可）をざっと読む
2. サンプルプログラム（コード）をとにかく動かす
3. サンプルプログラムをひたすら（？ある程度の量を） **読む**
   - 読むことで、目が馴染む
   - 理解できないところは調べる。最近は大抵のことは検索すると出てくるので便利
   - → [　　] を高める

前田（個人）的言語勉強法

1. まずは入門書（Webも可）をざっと読む
2. サンプルプログラム（コード）をとにかく動かす
3. サンプルプログラムをひたすら（？ある程度の量を）**読む**
   - 読むことで、目が馴染む
   - 理解できないところは調べる。最近は大抵のことは検索すると出てくるので便利
   - → **経験値**を高める

前田（個人）的言語勉強法（続き）

4. サンプルプログラムを少しだけ［　　］て挙動の違いを見る
   ［　　］
   - 感じることで、体に［　　］
5. …２以下を、繰り返す

前田（個人）的言語勉強法（続き）

4. サンプルプログラムを少しだけ書きかえて挙動の違いを見る
   - 感じることで、体に
5. …２以下を、繰り返す

前田（個人）的言語勉強法（続き）

4. サンプルプログラムを少しだけ**書きかえ**て挙動の違いを見る**（感じる）**
   - 感じることで、体に［　　］
5. …２以下を、繰り返す

前田（個人）的言語勉強法（続き）

4. サンプルプログラムを少しだけ**書きかえ**て挙動の違いを見る**（感じる）**
   - 感じることで、体に**馴染む**
5. …２以下を、繰り返す

前田（個人）的コードの読みかた

1. できるだけプログラマの [　　] を読みとく
2. １行ごとに、[　　]
3. その上で、コードの [　] のパターンを読みとく

前田（個人）的コードの読みかた

1. できるだけプログラマの **意図** を読みとく
2. １行ごとに、 [　　]
3. その上で、コードの [　] のパターンを読みとく

前田（個人）的コードの読みかた

1. できるだけプログラマの **意図** を読みとく
2. １行ごとに、 **意味がある（はず）**
3. その上で、コードの □ のパターンを読みとく

前田（個人）的コードの読みかた

1. できるだけプログラマの **意図** を読みとく
2. １行ごとに、 **意味がある（はず）**
3. その上で、コードの **塊** のパターンを読みとく

前田（個人）的コードの読みかた（続き）

4. ある目的にはこのパターン、のような［　　］を見付ける（アルゴリズムの教科書も適宜参照すべき）
5. 先人の知恵に学ぶ［　　］気持ちが必要（誇張あり）

☆これらが体感でき［　　］を積んでいけば、読むのも書くのもワンランク上のレベルになる

前田（個人）的コードの読みかた（続き）

4. ある目的にはこのパターン、のようなイディオム（慣用句）を見付ける（アルゴリズムの教科書も適宜参照すべき）
5. 先人の知恵に学ぶ［　　］気持ちが必要（誇張あり）

☆これらが体感でき［　　］を積んでいけば、読むのも書くのもワンランク上のレベルになる

前田（個人）的コードの読みかた（続き）

4. ある目的にはこのパターン、のような**イディオム（慣用句）**を見付ける（アルゴリズムの教科書も適宜参照すべき）
5. 先人の知恵に学ぶ**謙虚な**気持ちが必要（誇張あり）

☆これらが体感できを積んでいけば、読むのも書くのもワンランク上のレベルになる

前田（個人）的コードの読みかた（続き）

4. ある目的にはこのパターン、のような**イディオム（慣用句）**を見付ける（アルゴリズムの教科書も適宜参照すべき）
5. 先人の知恵に学ぶ**謙虚な**気持ちが必要（誇張あり）

☆これらが体感でき**経験値**を積んでいけば、読むのも書くのもワンランク上のレベルになる

プログラミング学習の心構え

- そもそもプログラミングを学習するのは何のため？
  - プログラミング [　　] を身につけるため
  - さらに言うと、プログラミング [　　] のベースとなる [　　] を鍛えるため
- プログラムは書いたようにしか動かない（！）
  - 思うように動かすためには、[　　] ないといけない
  - そのためには、[　　] で記述する力が必要

プログラミング学習の心構え

- そもそもプログラミングを学習するのは何のため？
  - プログラミング **感覚** を身につけるため
  - さらに言うと、プログラミング [　　] のベースとなる [　　] を鍛えるため
- プログラムは書いたようにしか動かない（！）
  - 思うように動かすためには、[　　] ないといけない
  - そのためには、[　　] で記述する力が必要

プログラミング学習の心構え

- そもそもプログラミングを学習するのは何のため？
  - プログラミング **感覚** を身につけるため
  - さらに言うと、プログラミング **感覚** のベースとなる [　　] を鍛えるため
- プログラムは書いたようにしか動かない（！）
  - 思うように動かすためには、[　　] ないといけない
  - そのためには、[　　] で記述する力が必要

プログラミング学習の心構え

- そもそもプログラミングを学習するのは何のため？
  - プログラミング **感覚** を身につけるため
  - さらに言うと、プログラミング **感覚** のベースとなる **論理的思考力** を鍛えるため
- プログラムは書いたようにしか動かない（！）
  - 思うように動かすためには、 [　　] ないといけない
  - そのためには、 [　　] で記述する力が必要

プログラミング学習の心構え

- そもそもプログラミングを学習するのは何のため？
  - プログラミング **感覚** を身につけるため
  - さらに言うと、プログラミング **感覚** のベースとなる **論理的思考力** を鍛えるため
- プログラムは書いたようにしか動かない（！）
  - 思うように動かすためには、**動くように書か** ないといけない
  - そのためには、 で記述する力が必要

プログラミング学習の心構え

- そもそもプログラミングを学習するのは何のため？
  - プログラミング **感覚** を身につけるため
  - さらに言うと、プログラミング **感覚** のベースとなる **論理的思考力** を鍛えるため
- プログラムは書いたようにしか動かない（！）
  - 思うように動かすためには、**動くように書か** ないといけない
  - そのためには、 **理詰め** で記述する力が必要

プログラミング学習の心構え（躓きの要因）

- 専門用語でつまずく…概念知識の理解は必要だが、それよりまず [　　] ことが大事（！）
- エラーが出て心が折れる…
  - コードの抜け漏れや書き損じがあってエラーを出してしまう→嫌気がさしたり、向いていないのではないかと自信をなくしてしまう
  - エラーは出たら [　　] だけ。むしろエラーをたくさん出して、失敗の原因を [　　] することが成長につながる。
- 心理的なハードルを下げる工夫が必要（?!）

プログラミング学習の心構え（躓きの要因）

- 専門用語でつまずく…概念知識の理解は必要だが、それよりまず **動かす** ことが大事（！）
- エラーが出て心が折れる…
  - コードの抜け漏れや書き損じがあってエラーを出してしまう→嫌気がさしたり、向いていないのではないかと自信をなくしてしまう
  - エラーは出たら [　　] だけ。むしろエラーをたくさん出して、失敗の原因を [　　] することが成長につながる。
- 心理的なハードルを下げる工夫が必要（?!）

プログラミング学習の心構え（躓きの要因）

- 専門用語でつまずく…概念知識の理解は必要だが、それよりまず **動かす** ことが大事（！）
- エラーが出て心が折れる…
  - コードの抜け漏れや書き損じがあってエラーを出してしまう→嫌気がさしたり、向いていないのではないかと自信をなくしてしまう
  - エラーは出たら **直せばいい** だけ。むしろエラーをたくさん出して、失敗の原因を [　　] することが成長につながる。
- 心理的なハードルを下げる工夫が必要（?!）

プログラミング学習の心構え（躓きの要因）

- 専門用語でつまずく…概念知識の理解は必要だが、それよりまず **動かす** ことが大事（！）
- エラーが出て心が折れる…
  - コードの抜け漏れや書き損じがあってエラーを出してしまう→嫌気がさしたり、向いていないのではないかと自信をなくしてしまう
  - エラーは出たら **直せばいい** だけ。むしろエラーをたくさん出して、失敗の原因を **蓄積** することが成長につながる。
- 心理的なハードルを下げる工夫が必要（?!）

おしまい

質問・コメント等あれば、何でもお気軽に
aipr@e-chan.jp に
メールください！