

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

クイズ1:「プログラム」という言葉から 連想するものは？

- 授業メモ

- 命令がずっと並んでる感じ
- アルファベットとか記号が画面いっぱいにならんでる感じ

「プログラム」という言葉から連想するものは？

- スーパー大辞林

- (1) 物事の予定。行事の進行についての計画。

- (2) 映画演劇コンサートなどの演目や曲目，あらすじや解説などを書いた表や小冊子。

- (3) コンピューターに，情報処理を行うための動作手順を指定するもの。また，それを作成すること。

- 授業「プログラミング」でやるのは(3)。でも、(1)～(3)は何かしら共通してるところがあるから「プログラム」と呼ばれているのでは？

行事式次第を例に考えてみよう

1. 2017/4/5

1. コース別オリエンテーション

1. コース長挨拶およびコース紹介
(コース長)
2. コースの教員紹介(全教員)
3. 新入生の自己紹介
4. 懇談・質疑応答

上司からの指示

「このプログラムに沿ってオリエンテーションを開催してね。期待してるよ😊」

2. 教職説明

3. 数学プレースメントテスト

4. インストール大会

クイズ2: あなたならどう
開催する？

皆さんの考える開催方法

- 授業メモ

- 正確に全部、淡々とこの通りに実行する。
- 時間配分考える。

「行事の開催」と「プログラムの実行」における 共通部分と差異

行事式次第の開催(実行)

- 書かれている順番通りに司会者が実行する。
- 司会者は人間であり、式次第は自然言語で記述されている。
- 式次第をどう実行するかは、司会者の裁量に委ねられている。
- 実行する度に何かしら違いがある。

プログラムの実行

- 書かれている順番通りにコンピュータが実行する。
- コンピュータは計算機であり、プログラムはプログラミング言語で記述されている。
- プログラムをどう実行するかは、言語仕様で厳密に規程されている。
- 何度実行しても必ず同じように実行する。

「プログラム」の特徴

- 実行するのはコンピュータ

- コンピュータ言語(プログラミング言語)
- Machine code (機械語)
- Assembly language (アセンブラ)
- Basic, C言語,,,,
- Java (後期授業「プログラミング2」)
- 軽量プログラミング言語
 - 何らかの実際の機能によるカテゴライズではなく、習得・学習・使用が容易な言語
 - Perl, PHP, Ruby, Python,,,,

- 再現が容易

- 書いた通りに動く

- 複製が容易

- 複製コストはほぼ無視できる

- (条件付きで)編集や再利用が容易

- テスト(Testing)
- バージョン管理

プログラミングとは？

- 広義（広辞苑）

- コンピューターのプログラムを作成すること。プログラムの仕様の決定、誤りの修正などの作業などを含めていうこともある。

- 狭義

- (仕様通りに)プログラムを書くこと。
- 「コーディング」

- 仕様

- (1) やりかた。方法手段。「返事のーが気に入らない」
- (2) →仕様書に同じ。
- 当面は仕様決定済みの課題に取り組む。

プログラミングに含まれる3ステップ

- (コンピュータはプログラミング言語しか知らないので、)プログラマは、実現したいことをプログラミング言語に翻訳する必要がある。
 1. 「実現したいこと」の理解が大前提。この理解が不十分の場合、何を翻訳したら良いかが分からない。
 2. 「理解したこと」を手順として説明できるレベルまで整理する。
 3. 最後に、プログラムへ「翻訳(記述)」する。
 - 初学者は、プログラミング言語の構文・仕様を学ぶ必要がある。

一般的に、「プログラム」には3番目の翻訳結果しか残らない。1の理解や、2の整理が欠落してしまう傾向にある。レポートを書く時、また友人らと一緒に課題に取り組む際には、1,2の説明をするように心がけよう。

翻訳技術を学ぶ

授業計画:
第1回～第8回

- 代表的な原則
- **KISS原則**
 - Keep it simple, stupid!
 - 小さく作り、組み合わせる。
 - 各部品(関数)をテストする。
- **DRY原則**
 - Don't repeat yourself.
 - 繰り返しを避ける。

授業計画:
第2回～

- 経験を積む
 - 当面は決められた仕様を「どう実装するか」。
 - 少しずつ検討範囲を広げていく。
 - **様々な問題にトライする。**
 - **ペア・プログラミング。**
 - 互いに教え合う
 - 知識の共有化

プログラミングを円滑に進めるために

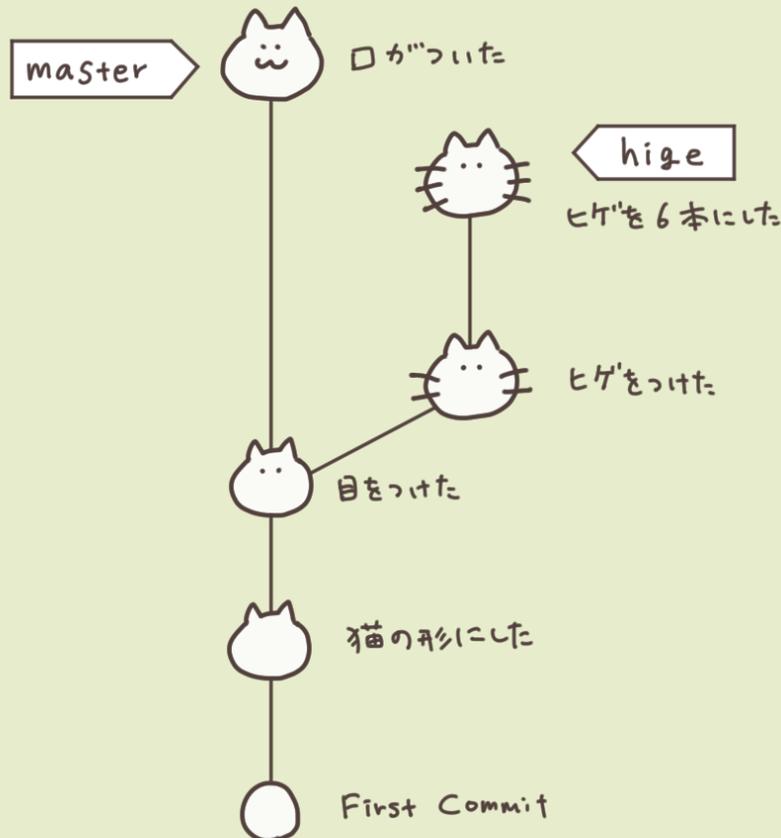
授業計画:
第15回

授業計画:
第6回～

• バージョン管理

• より高度な機能

- テスト(Testing)
- デバッグ実行
- ベクトル・行列演算
- グラフ描画
- 文字コード
- 正規表現
- (クラス)
- (オブジェクト指向)



<http://kray.jp/blog/git-pull-rebase/>

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？

1. プログラムの特徴

2. プログラミングにおける2大原則

3. プログラミングを円滑に進めるための周辺技術

2. 演習1:教科書のコードを動かしてみる

3. 演習2:変数と等号の利用、実行の様子

4. シラバス

5. 授業方針

6. 宿題・補足

実現したいことを理解し、手順として整理し、プログラミング言語で記述(翻訳)すること。KISS原則、DRY原則を意識しよう。ペアプロで互いに教え合おう。テスト・バージョン管理・デバッグ実行で円滑に開発を進めよう。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

演習1:

教科書のサンプルコードを動かしてみる

1. 「ターミナル」を起動。
 - Finder -> アプリケーション -> ユーティリティ -> ターミナル.app
 - Dockへ登録するか、ショートカット登録しておくとも便利。
2. Pythonインタプリタの起動。
 - 「python3」と入力。
3. インタプリタにコードを入力。
 - e.g., 最初のコード例, p.9, 2.1節
`print('Yankees rule!')`
 - 注意: Python 2.x と 3.x とで書式が異なることがあります。**教科書の脚注**参照。
4. インタプリタを終了
 - 「exit()」もしくは「Ctrl-D」。
 - Controlキーを押しながらDを押す

「今日の目標」
手順1~4を数回繰り返して、**教科書のコード例を試せるようになる**。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

Pythonインタプリタを使った作業工程に慣れよう。
Python2と3の違いに注意。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

演習2:

変数の利用(教科書2.1.2節, p.11)

コード(教科書の例)	コードの意味
<pre>pi = 3 radius = 11 print(pi) print(radius) area = pi * (radius**2) print(area) radius = 14 print(area)</pre>	<ol style="list-style-type: none">1. 変数piを用意し、3を割り当てろ。2. 変数radiusを用意し、11を割り当てろ。3. 変数piの中身を出力しろ。4. 変数radiusの中身を出力しろ。5. 変数areaを用意し、等号の右辺にある式を実行し、その結果を変数areaに割り当てろ。6. 変数areaの中身を出力しろ。7. 変数radiusは既に存在しているので、中身を14に置き換えろ。8. 変数areaの中身を出力しろ。

実行の様子1

コード(教科書をやや変更)

```
pi = 3
radius = 11
print(pi)
print(radius)
area = pi * (radius**2)
print(area)
radius = 14
print(area)
```

実行後に残るデータ

1. 変数名pi
2. 3
3. pi -> メモリ2番地 #紐付け

メモリに保存されている場所(番地)

実行の様子2

コード(教科書をやや変更)

```
pi = 3
radius = 11
print(pi)
print(radius)
area = pi * (radius**2)
print(area)
radius = 14
print(area)
```

実行後に残るデータ

1. 変数名pi
2. 3
3. pi -> メモリ2番地 #紐付け
4. 変数名radius
5. 11
6. radius -> メモリ5番地

実行の様子3

コード(教科書をやや変更)

```
pi = 3
radius = 11
print(pi)
print(radius)
area = pi * (radius**2)
print(area)
radius = 14
print(area)
```

実行後に残るデータ

1. 変数名pi
2. 3
3. pi -> メモリ2番地 #紐付け
4. 変数名radius
5. 11
6. radius -> メモリ5番地
(print時には変更なし)

実行の様子4

コード(教科書をやや変更)

```
pi = 3
radius = 11
print(pi)
print(radius)
area = pi * (radius**2)
print(area)
radius = 14
print(area)
```

実行後に残るデータ

1. 変数名pi
2. 3
3. pi -> メモリ2番地 #紐付け
4. 変数名radius
5. 11
6. radius -> メモリ5番地
7. 変数名area
8. 363
9. area -> メモリ8番地

実行の様子5

コード(教科書をやや変更)

```
pi = 3
radius = 11
print(pi)
print(radius)
area = pi * (radius**2)
print(area)
radius = 14
print(area)
```

実行後に残るデータ

1. 変数名pi
2. 3
3. pi -> メモリ2番地 #紐付け
4. 変数名radius
5. 11
6. radius -> メモリ5番地
7. 変数名area
8. 363
9. area -> メモリ8番地
(print時には変更なし)

実行の様子6

コード(教科書をやや変更)

```
pi = 3
radius = 11
print(pi)
print(radius)
area = pi * (radius**2)
print(area)
radius = 14
print(area)
```

実行後に残るデータ

1. 変数名 pi
2. 3
3. pi -> メモリ2番地 #紐付け
4. 変数名 radius #既に存在している
5. 11
6. radius -> メモリ10番地 #紐付けしなおし
7. 変数名 area
8. 363
9. area -> メモリ8番地
10. 14

実行の様子7

コード(教科書をやや変更)

```
pi = 3
radius = 11
print(pi)
print(radius)
area = pi * (radius**2)
print(area)
radius = 14
print(area)
```

実行後に残るデータ

1. 変数名pi
 2. 3
 3. pi -> メモリ10番地
 4. 変数名radius
 5. 11
 6. radius -> メモリ5番地
 7. 変数名area
 8. 363
 9. area -> メモリ8番地
 10. 14
- (print時には変更なし)

覚えてほしいこと

- 等号(=)の意味

- 数学では、等式。「左辺と右辺は等価である」という意味。
- Python(プログラム)では、**assignment(割り当て)**や**binding(紐付け)**。動作としては、
 - (1)右辺を**evaluate(評価≡実行)**し、
 - (2)その結果を左辺に紐付ける。

- 逐次処理

- 順番通りに実行する。
- 実行後に残るデータ
 - 何をどう処理したかは覚えていない。
 - コードを実行することで「得られた値」が一時的に残る。
 - **等号を使って変数に紐付けることで、「得られた値」を再利用することができる。**

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

プログラミングにおける等号は、(1)右辺を評価(実行)して、(2)結果を変数に紐付ける。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス  達成目標、評価方法等について必要な時に参照。
5. 授業方針  疑問に感じた点はどんどん質問しよう。
6. 宿題・補足

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

予習・復習を前提に進める

- 予習・復習:教科書は各自の自習教材
 - 教科書を読んで分かることを、わざわざ授業時間に一緒にやるのは時間が勿体無い。
 - 教科書読んで分からないことは自分から相談しよう。
 - 授業では Part 1 (Chapter 8まで)をメインに扱う。約100ページ。これを4回は読もう。余裕がある人はChapter 9以降を好きな順番でトライ！
 - 同じペースで読む必要はない。分かる部分はショートカットし、分からない部分を減らしていこう。
- オリジナル課題のすゝめ
 - 予習・復習とは別に、自身で取り組みたいことをやる時間も取れると良い。例えば、「2単位授業の自習4時間」のうち、予習・復習を平均して2~3時間で終え、残り時間を自身や仲間らとのプロジェクトに割り当てる等。(課題設定時には、平均2,3時間で終わられることを想定した分量を意識するようにします)

一人ではやれないことをサポート

- 授業でやること
 - 重要な点に関する解説。デモ・演習。
 - 一人ではやれないこと(ペア・プログラミング)。
 - ペアプログラミングのやりかた: <http://goo.gl/ZWtIW>
 - 講義「プログラミング演習1」との連携。
 - 課題サポート等

教育目標

- できるとは、「次の一歩」が分かること
- 授業方針
 - 「次の一歩」が分かるまで手を貸す

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

疑問に感じた点は
どんどん質問しよう。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

宿題

- 予習1: 教科書読み
 - 1章 * 概要掴むぐらい(30分程度)でok
 - 2章～2.1.2節まで * 1～2時間程度想定
 - 2.1.3節はスキップ。
 - 余裕ある人は2.2節。
- 課題レポート: 授業ページ参照
- 自習(オススメ)
 - paiza
 - Python入門編1:プログラミングを学ぶ (全9回)
 - <https://paiza.jp/works/python3/primer>
 - progate
 - Python I, II
 - <https://prog-8.com>

補足

- 辞書アプリ

- アプリケーション->辞書.app
- (1) 起動して、左上の「辞書」メニューから「環境設定」を選び、
- (2) New Oxford American Dictionary、Oxford American Writer's Thesaurus あたりの**英英辞書**にチェック

和訳して読むというより、英語で読む練習をした方が有益

- 予習

- **内容を100%理解することは求めています。「予習＝まだならっていないところを前もって学習練習しておくこと。」**

- ミニテストの趣旨

- 復習内容についてはできるだけ正しく答えて欲しい。
- 予習内容については今は間違ってもOK。その日の授業で理解して欲しい。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？
 1. プログラムの特徴
 2. プログラミングにおける2大原則
 3. プログラミングを円滑に進めるための周辺技術
2. 演習1:教科書のコードを動かしてみる
3. 演習2:変数と等号の利用、実行の様子
4. シラバス
5. 授業方針
6. 宿題・補足

教科書・参考サイト参照し
つつ、手を動かそう。

プログラミング1

(第1回) 卓上プログラミングによる開発設計概観、Pythonインタプリタの起動と逐次処理・変数の利用

1. プログラミングとは何か？

1. プログラムの特徴

2. プログラミングにおける2大原則

3. プログラミングを円滑に進めるための周辺技術

2. 演習1:教科書のコードを動かしてみる

3. 演習2:変数と等号の利用、実行の様子

4. シラバス

5. 授業方針

6. 宿題・補足

実現したいことを**理解**し、手順として**整理**し、プログラミング言語で**記述(翻訳)**すること。**KISS原則**、**DRY原則**を意識しよう。ペアプロで互いに教え合おう。テスト・バージョン管理・デバッグ実行で円滑に開発を進めよう。

Pythonインタプリタを使った作業工程に慣れよう。**Python2と3の違い**に注意。

プログラミングにおける等号は、**(1)右辺を評価(実行)**して、**(2)結果を変数に紐付ける**。

達成目標、評価方法等について必要な時に参照。

疑問に感じた点はどんどん質問しよう。

教科書・参考サイト参照しつつ、手を動かそう。

参考文献

- 教科書: Introduction to Computation and Programming Using Python, Revised And Expanded Edition
- 来たるべき、「みんな」のコードのために: 平成4年生まれがつくるプログラマーの学校,
<http://wired.jp/2016/03/04/kusano-teacher/>
- 20歳を過ぎてからプログラミングを学ぼうと決めた人たちへ, <http://www.slideshare.net/ShuUesugi/20-9290892>
- (paiza) Python入門編1:プログラミングを学ぶ (全9回), <https://paiza.jp/works/python3/primer>
- (progate) Python I, II: <https://prog-8.com>