

# 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)

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。ペアプロで互いに教え合おう。

### 2. 演習1: 教科書のコードを動かしてみる

ターミナル+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。

### 3. 演習2: オブジェクトと式と型

代表的な型(int, float, str)、型の確認方法と演算子を覚えよう。

### 4. 演習3: 変数と等号の利用、実行の様子

### 5. シラバス

達成目標、評価方法等について必要な時に参照。

### 6. 授業方針

疑問に感じた点はどんどん質問しよう。

### 7. 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。

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2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

1. Chapter 1 の補足1
  1. Calculations and Remembers
  2. Computational thinking
2. Chapter 2 -- 2.1.2までの補足
  1. Glossaries, 用語集1, 2, 3
  2. Reserved words, 予約語
3. 文字列結合の例
4. スクリプトの利用
  1. スクリプトとは？
  2. スクリプトを書いて動かしてみよう
  3. スクリプト vs. インタプリタ
5. 変数名・ファイル名の命名規則
6. マニュアルの参照
7. 演習
8. 宿題

教科書に書かれていないか、後回しになっている内容

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

# Chapter 1 の補足1

複数回に分けて補足します

# Chapter 1,

## A Computer does two things

- **Calculations** (計算)
- **Remembers** the results of those calculations (計算結果を覚える)

# Chapter 1, Computational thinking (計算的思考)

## Declarative knowledge (宣言的知識)

- composed of statements of fact. (明確な事実の宣言)
- “the square root of  $x$  is a number  $y$  such that  $y*y = x$ .”

## Algorithm (アルゴリズム)

An algorithm is **a finite list of instructions** that describe a computation that when executed on a provided **set of inputs** will proceed through a set of well-defined states and eventually produce **an output**.

## Imperative knowledge (命令的知識)

- “how to” knowledge, or recipes for deducing information. (情報を導くためのレシピ)
- start with a guess,  $g$ .
- if  $g*g$  is close enough to  $x$ , stop and say that  $g$  is the answer.
- otherwise create a new guess by averaging and  $x/g$ , i.e.,  $(g+x/g)/2$ .
- Using this new guess, which we again call  $g$ , repeat the process until  $g*g$  is close enough to  $x$ .

# プログラミング1

## (第2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

### 1. Chapter 1 の補足1

1. Calculations and Remembers
2. Computational thinking

欲しい出力を得るためのレシピを考える必要がある。**レシピ≡アルゴリズム**。

### 2. Chapter 2 -- 2.1.2までの補足

1. Glossaries, 用語集1, 2, 3
2. Reserved words, 予約語

### 3. 文字列結合の例

### 4. スクリプトの利用

1. スクリプトとは？
2. スクリプトを書いて動かしてみよう
3. スクリプト vs. インタプリタ

### 5. 変数名・ファイル名の命名規則

### 6. マニュアルの参照

### 7. 演習

### 8. 宿題

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

# Chapter 2 -- 2.1.2までの補足

Glossaries, 用語集  
Reserved words, 予約語

# Glossaries, 用語集 1

prompt of Shell  
(シェルのプロンプト)

prompt of Interpreter  
(インタプリタのプロンプト)

command, statement  
(コマンド/命令, 文)

code, program  
(コード, プログラム)  
script (スクリプト)

コードやプログラムをファイルに保存したもの  
code, program, program file  
script, script file  
source, source file

```
ターミナル — -zsh — z
Last login: Mon Apr 16 12:32:06 on tt
[oct:tnal%
[oct:tnal% python
Python 3.6.4 |Anaconda, Inc.| (default
[GCC 4.2.1 Compatible Clang 4.0.1 (ta
Type "help", "copyright", "credits" c
[>>>
[>>> print('Yankees rule!')
Yankees rule!
>>>
>>> pi = 3
>>> radius = 11
>>> area = pi * (radius**2)
>>> print(area)
363
>>>
oct:tnal%
```

# Glossaries, 用語集 2

operator  
(オペレータ, 演算子)

type of object  
(オブジェクトの型)

assignment  
(代入文)

variable  
(変数)

comparison (relational) operator  
(比較演算子)

int (整数), float (浮動小数点数), str(文字列)  
**bool**: boolean values, True/False (ブール型)  
See also,

<https://docs.python.org/3.6/library/stdtypes.html#built-in-types>

$a == b$ : aとbが等しい場合にTrue、それ以外の場合にはFalseを返す。  
 $a != b$ : aとbが等しくない場合にTrue、それ以外の場合にはFalseを返す。

大文字小文字で意味が異なる。  
シングルクォート(')の有無でも変わる。  
Trueは予約語のboolean value。  
trueはここでは未定義の変数。  
'true'はstr型のオブジェクト。

```
>>> 3 + 2
5
>>> type(5)
<class 'int'>
>>> a = 3 + 2
>>> type(a)
<class 'int'>
>>> type(5.0)
<class 'float'>
>>>
>>> 3 + 2 == 5
True
>>> 3 + 2 == 4
False
>>> 3 + 2 != 4
True
>>> type(True)
<class 'bool'>
>>> type(true)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'true' is not defined
>>> type('true')
<class 'str'>
>>>
```

# Glossaries, 用語集 3

comment  
(コメント)

「**#**から行末まで」は、プログラムを読みやすくするための**コメント**(=プログラマへの補足文)。命令文の途中からでもコメント可能。「...」と表示されたら、単にリターンキーを押そう。

```
[>>> # これはコメント行
```

```
[...]
```

```
[>>> score = 80 # プログラミング1の成績
```

```
[>>> score >= 60
```

```
True
```

```
[>>> number_of_absence = 5
```

```
[>>> number_of_absence < 5
```

```
False
```

```
[>>> (score >= 60) and (number_of_absence < 5)
```

```
False
```

```
>>> █
```

comparison (relational)  
operator (比較演算子)

boolean  
operator (論理演算子)

「**>=** (greater than or equal)」は、左辺が右辺以上のときにTrue。  
「**<=** (less than or equal)」は、左辺が右辺以下のときにTrue。  
「**>** (greater than)」は、左辺が右辺より大きいときにTrue。  
「**<** (less than)」は、左辺が右辺より小さいときにTrue。

「**and**」は、左辺と右辺が共にTrueの時にTrue。それ以外ならばFalseとなる。

「**or**」は、左辺と右辺いずれか一つでもTrueの時にTrue。それ以外の時にはFalseとなる。

「**not**」は、右辺がTrueの時にFalse。そうでなければTrueとなる。

# Reserved words, 予約語

<https://goo.gl/4TclUz>

- 一覧(赤丸は先週～今回出てきた予約語)

|        |          |         |          |        |
|--------|----------|---------|----------|--------|
| False  | class    | finally | is       | return |
| None   | continue | for     | lambda   | try    |
| True   | def      | from    | nonlocal | while  |
| and    | del      | global  | not      | with   |
| as     | elif     | if      | or       | yield  |
| assert | else     | import  | pass     |        |
| break  | except   | in      | raise    |        |

# プログラミング1

## (第2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

1. Chapter 1 の補足1
  1. Calculations and Remembers
  2. Computational thinking
2. Chapter 2 -- 2.1.2までの補足
  1. Glossaries, 用語集1, 2, 3
  2. Reserved words, 予約語
3. 文字列結合の例
4. スクリプトの利用
  1. スクリプトとは？
  2. スクリプトを書いて動かしてみよう
  3. スクリプト vs. インタプリタ
5. 変数名・ファイル名の命名規則
6. マニュアルの参照
7. 演習
8. 宿題

レシピを記述するための道具(基本的な型・算術演算子・比較演算子・論理演算子)を使えるようになる。

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

# 文字列結合の例

(教科書に書かれてません)

# 文字列結合の例1(先週の内容)

数値と文字列を直接足すことはできない

**TypeError: must be str, not int**  
**型エラー: str型であるべき**

```
>>> # 文字列の演算
... 'スライム' + 1
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
TypeError: must be str, not int
```

文字列同士の足し算は、結合になる。

```
>>> 'スライム' + 'が2体現れた'
'スライムが2体現れた'
>>> enemy = 'スライム'
>>> enemy + 'が2体現れた'
'スライムが2体現れた'
>>> output = enemy + 'が2体現れた'
>>> print(output)
スライムが2体現れた
>>> □
```

# 文字列結合の例2

```
>>> enemy = 'スライム'
>>>
>>> # case 1: 出力したい文字列を1変数に用意してからprint()する。
... output = enemy + 'が2体現れた'
>>> print(output)
スライムが2体現れた
>>>
>>> # case 2: print()内で演算処理する。
... print(enemy + 'が2体現れた')
スライムが2体現れた
>>>
>>> # case 3: str.format()形式（Python公式）を利用する。
... print('{0}が2体現れた'.format(enemy))
スライムが2体現れた
>>> □
```

**演算子が含まれる場合**

先に演算し、その結果を出力。

**str.format()形式**

「'文字列'.format(変数)」  
文字列中の{0}等を変数に  
置き換えて文字列を作成。

```
>>> enemy = 'スライム'
>>> num = 3
>>> '{0}が{1}体現れた'.format(enemy, num)
'スライムが3体現れた'
>>> output = '{0}が{1}体現れた'.format(enemy, num)
>>> print(output)
スライムが3体現れた
>>> □
```

# プログラミング1

## (第2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

1. Chapter 1 の補足1
  1. Calculations and Remembers
  2. Computational thinking
2. Chapter 2 -- 2.1.2までの補足
  1. Glossaries, 用語集1, 2, 3
  2. Reserved words, 予約語
3. 文字列結合の例
4. スクリプトの利用
  1. スクリプトとは？
  2. スクリプトを書いて動かしてみよう
  3. スクリプト vs. インタプリタ
5. 変数名・ファイル名の命名規則
6. マニュアルの参照
7. 演習
8. 宿題

基本演算と`str.format`書式を読めるようになるろう。

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

# スクリプトの利用

## (教科書に書かれてません)

スクリプトとは？

スクリプトを書いて動かしてみよう

スクリプト vs. インタプリタ

# スクリプトとは？

- これまで
  - インタプリタ上で直接コードを入力して実行する。
  - 利点: 入力したコードの結果をその場で見れる。
  - 欠点: 同じコードを複数回実行したい場合には、その都度タイプする必要がある。
- 別の方法
  - コードをファイルに保存(拡張子は「.py」)。
  - コードが保存されたファイルを「source file, source code, script file, script code」等と呼ぶ。
  - 保存したファイル名が「test.py」ならば、ターミナル上で次のようにコマンドを実行すると、スクリプトを実行できる。

```
% python test.py
```

冒頭の「%」は、シェルのプロンプトを表しています。実際にタイプするコマンドは「python test.py」

# スクリプトを書いて動かしてみよう

- やりたいこと
  - インタプリタ上で「`print('hello!')`」と入力してエンターキーを押すと、「hello!」と返してくる。
  - 同じコードをスクリプトファイルとして保存して、実行したい。
- エディタでファイルを編集。
  - 手順1: テキストエディタ(エディタ)を起動。
    - 今回はCotEditorを起動。
  - 手順2: シンタックスから「Python」を選択。
    - エディタ毎に設定方法は異なる。一般的には、下記の手順3をやると自動的にPythonモードにしてくれることが多い。
  - 手順3: コードを書いて、「.py」という拡張子でファイル保存。
    - `~/prog1/` を作成して、そこに保存しよう。(プログラミング演習の復習)
- ターミナル上でスクリプトを実行。
  - 手順1: ターミナルから、ファイルを保存したディレクトリに移動。
    - `~/prog1/` に移動しよう。(プログラミング演習の復習)
  - 手順2: 「`python ファイル名`」として、ファイル名を指定して実行。
    - ファイル名が`test.py`なら「`python test.py`」と実行。
    - `python`とファイル名の間にはスペースを挟もう。

# スクリプトファイル vs. インタプリタ

## スクリプトファイルの特徴

- 利点
  - 一度動くように仕上げたら、後は中身を見なくても同じ動作をまとめて再現することができる。
- 欠点
  - 途中結果を確認しづらい。

## インタプリタの特徴

- 利点
  - 途中結果を確認しながら検討できる。
- 欠点
  - 一度書いたコードを再現するのが面倒(その都度書く必要がある)。

## オススメ

インタプリタで細かい動作や手順を確認し、「こう書けば実現できそうだ」という感触を得る。その後、スクリプトファイルとして書き直す。

# プログラミング1

## (第2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

1. Chapter 1 の補足1
  1. Calculations and Remembers
  2. Computational thinking
2. Chapter 2 -- 2.1.2までの補足
  1. Glossaries, 用語集1, 2, 3
  2. Reserved words, 予約語
3. 文字列結合の例
4. スクリプトの利用
  1. スクリプトとは？
  2. スクリプトを書いて動かしてみよう
  3. スクリプト vs. インタプリタ
5. 変数名・ファイル名の命名規則
6. マニュアルの参照
7. 演習
8. 宿題

インタプリタ実行と**ファイル実行**を使い分けよう。

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

# 変数名・ファイル名の命名規則

(教科書に書かれてません)

# 変数名・ファイル名の命名規則

- 変数名やファイル名に使える文字
  - 原則として「英数字」と「\_ (underscore)」。
  - 大文字小文字は区別される。
  - 冒頭に数字は使えない。
- 変数名・ファイル名を適切に選択する
  - 「適切」とは？
    - Level 1: その変数が表す用語の英単語(小文字)を使う。
    - Level 2: 複数単語で命名したいなら「\_ (underscore)」で繋げて書く。
    - Level 3: 同じモジュール・クラス内では統一規約を採用する。
    - Level 4: 英単語の微妙なニュアンスの差に気をつける。
  - 規約の例: Google Python Style Guide
    - <https://google.github.io/styleguide/pyguide.html>
    - Naming
      - Modules(ファイル名): lower\_with\_under
      - Local Variables(変数名): lower\_with\_under

# プログラミング1

## (第2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

1. Chapter 1 の補足1
  1. Calculations and Remembers
  2. Computational thinking
2. Chapter 2 -- 2.1.2までの補足
  1. Glossaries, 用語集1, 2, 3
  2. Reserved words, 予約語
3. 文字列結合の例
4. スクリプトの利用
  1. スクリプトとは？
  2. スクリプトを書いて動かしてみよう
  3. スクリプト vs. インタプリタ
5. 変数名・ファイル名の命名規則
6. マニュアルの参照
7. 演習
8. 宿題

慣習を守ることによって「**他人が読みやすいコード (readable code)**」になる。

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

マニュアルの参照  
(教科書に書かれてない  
or  
後回しになってます)

# マニュアル

- 公式ドキュメント
  - <https://docs.python.org/3.6/index.html>
- help()コマンド
  - help(print)
- Google先生
  - e.g., 「python print」 or 「python3 print」
  - 注意
    - Python 2 <-> Python 3で異なることがある。
    - Web上の情報は間違ってることがある。

# プログラミング1

## (第2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

1. Chapter 1 の補足1
  1. Calculations and Remembers
  2. Computational thinking
2. Chapter 2 -- 2.1.2までの補足
  1. Glossaries, 用語集1, 2, 3
  2. Reserved words, 予約語
3. 文字列結合の例
4. スクリプトの利用
  1. スクリプトとは？
  2. スクリプトを書いて動かしてみよう
  3. スクリプト vs. インタプリタ
5. 変数名・ファイル名の命名規則
6. マニュアルの参照
7. 演習
8. 宿題

help()やオンラインマ  
ニュアルを活用しよう。

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

# 演習

初めてのレポート  
ペア・プログラミング

# ペアプロ演習

- 授業ページを参照

# 宿題

- 復習: 適宜
- 課題レポート1: ✕切: 第3回の講義開始前
- 予習: 教科書読み
  - 2章
    - 2.3 Strings and Input
    - 2.4 Iteration
  - (スキップ) 3章
  - 4章
    - (スキップ) 4章冒頭
    - 4.1.1 Function Definitions
- 復習・予習(オススメ): paiza, progate

# プログラミング1

## (第2回) Pythonインタプリタとスクリプトの体験1, ペア・プログラミングの導入

### 1. Chapter 1 の補足1

1. Calculations and Remembers
2. Computational thinking

欲しい出力を得るためのレシピを考える必要がある。**レシピ≡アルゴリズム**。

### 2. Chapter 2 -- 2.1.2までの補足

1. Glossaries, 用語集1, 2, 3
2. Reserved words, 予約語

レシピを記述するための道具(**基本的な型・算術演算子・比較演算子・論理演算子**)を使えるようになる。

### 3. 文字列結合の例

### 4. スクリプトの利用

基本演算と**str.format**書式を読めるようになる。

1. スクリプトとは?
2. スクリプトを書いて動かしてみよう
3. スクリプト vs. インタプリタ

インタプリタ実行と**ファイル実行**を使い分けよう。

### 5. 変数名・ファイル名の命名規則

### 6. マニュアルの参照

### 7. 演習

### 8. 宿題

慣習を守ることで「**他人が読みやすいコード (readable code)**」になる。

**help()**や**オンラインマニュアル**を活用しよう。

講義ページ: <http://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/>

# 参考文献

- 教科書: Introduction to Computation and Programming Using Python, Revised And Expanded Edition
- Python 3.6 documentation,  
<https://docs.python.org/3.6/index.html>
- Google Python Style Guide,  
<https://google.github.io/styleguide/pyguide.html>