復習

プログラミング1

(第5回) ループ処理(for文)、range()関数とリストによるシーケンス集合表現

1. レポートの書き方
2. Chapter 3.2 For Loops
    1. もう一つのループ処理
    2. シーケンス集合とコード例
3. Chapter 3.4 A Few Words About Using Floats
    1. 浮動小数点数の取り扱い
4. If文、ループ文の補足
    1. elif, continue, break
5. 演習
    1. 演習1～4: 初めてのペア・プログラミング
    2. 演習5: 数当てゲーム1 (大小ヒント付き) を実装してみよう
6. 宿題

シーケンス集合(リスト、文字列)に対する反復処理を書けるようになろう。

小数を使う際には丸め誤差と桁あふれに注意。

細かな制御方法を理解しよう

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

プログラミング1

(第6回) デバッグ実行、関数とスコープ、仕様、ユニットテスト、モジュール

1. Chapter 1 Turing Complete
2. Chapter 4.1.2 Keyword Arguments and Default Values
3. Chapter 4.1.3 Scoping
4. Chapter 4.2 Specifications
    1. docstring + doctest (教科書にありません)
5. Chapter 4.5 Modules
6. 演習（恐らく時間なし）
7. 宿題

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

Turing complete (チューリング完全)

＊教科書1章と4章冒頭

コンピュータの原型

- Universal Turing Machine (万能チューリングマシン)
  - 0か1を記述できる無限長のテープ（メモリ）があり、テープ上の移動と読み書きする命令を持つ。
  - Church-Turing Thesis (チューリングの提唱)
    - 「もし」ある関数が有限回の操作で計算可能なら、チューリングマシンでプログラム可能であり、それを計算できる。

参考:
チューリング・マシンとコンピュータ工学:
http://www.slideshare.net/junpeitsuji/ss-57954980

プログラミング1

(第6回) デバッグ実行、関数とスコープ、仕様、ユニットテスト、モジュール

コンピュータの原型である万能チューリングマシンについて調べてみよう。

1. Chapter 1 Turing Complete
2. Chapter 4.1.2 Keyword Arguments and Default Values
3. Chapter 4.1.3 Scoping
4. Chapter 4.2 Specifications
    1. docstring + doctest (教科書にありません)
5. Chapter 4.5 Modules
6. 演習（恐らく時間なし）
7. 宿題

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

Chapter 4, 4.1.2, 4.1.3, 4.2, 4.5の補足

4 FUNCTIONS, SCOPING, AND ABSTRACTION
4.1.2 Keyword Arguments and Default Values
4.1.3 Scoping
4.2 Specifications
4.5 Modules

4.1.2 Keywords Arguments and Default Values

```
class range(object)
 |  range(stop) -> range object
 |  range(start, stop[, step]) -> range object
 |
 |  Return an object that produces a sequence of integers from start (inclusive)
 |  to stop (exclusive) by step.
```

```
# コード例1
def myrange(start, stop, step=1):
    result = []
    num = start
    while num<stop:
        result.append(num)
        num += step

    return result
```

引数名とデフォルト値の指定

```
# 実行例1
>>> myrange(0, 5)
[0, 1, 2, 3, 4]
>>> myrange(start=0, stop=5)
[0, 1, 2, 3, 4]
>>> myrange(0, 5, step=2)
[0, 2, 4]
```

プログラミング1

(第6回) デバッグ実行、関数とスコープ、仕様、ユニットテスト、モジュール

1. Chapter 1 Turing Complete
2. Chapter 4.1.2 Keyword Arguments and Default Values
3. Chapter 4.1.3 Scoping
4. Chapter 4.2 Specifications
    1. docstring + doctest (教科書にありません)
5. Chapter 4.5 Modules
6. 演習（恐らく時間なし）
7. 宿題

関数の引数にはデフォルト値を設定できる！

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

Stack frame と Name Space

```
def f(x):
    y = 1
    x = x + y
    print('f(x): x = {0}'.format(x))
    print('f(x): y = {0}'.format(y))
    return(x)

x = 3
y = 2
z = f(x)
print('x = {0}'.format(x))
print('y = {0}'.format(y))
print('z = {0}'.format(z))

def g():
    print('z = {0}'.format(z))g()
```

(1) 関数f()の定義

＊コード内部については実行時に作成される。定義時点では関数名のみ。

Stack frame: 1
f()

(2) コードの実行

Stack frame: 1
f(), x=3, y=2

(3) 関数実行(関数呼び出し)

(3-1) 関数呼び出し直前

Stack frame: 1
f(), x=3, y=2, z

(3-2) 関数実行,return直前

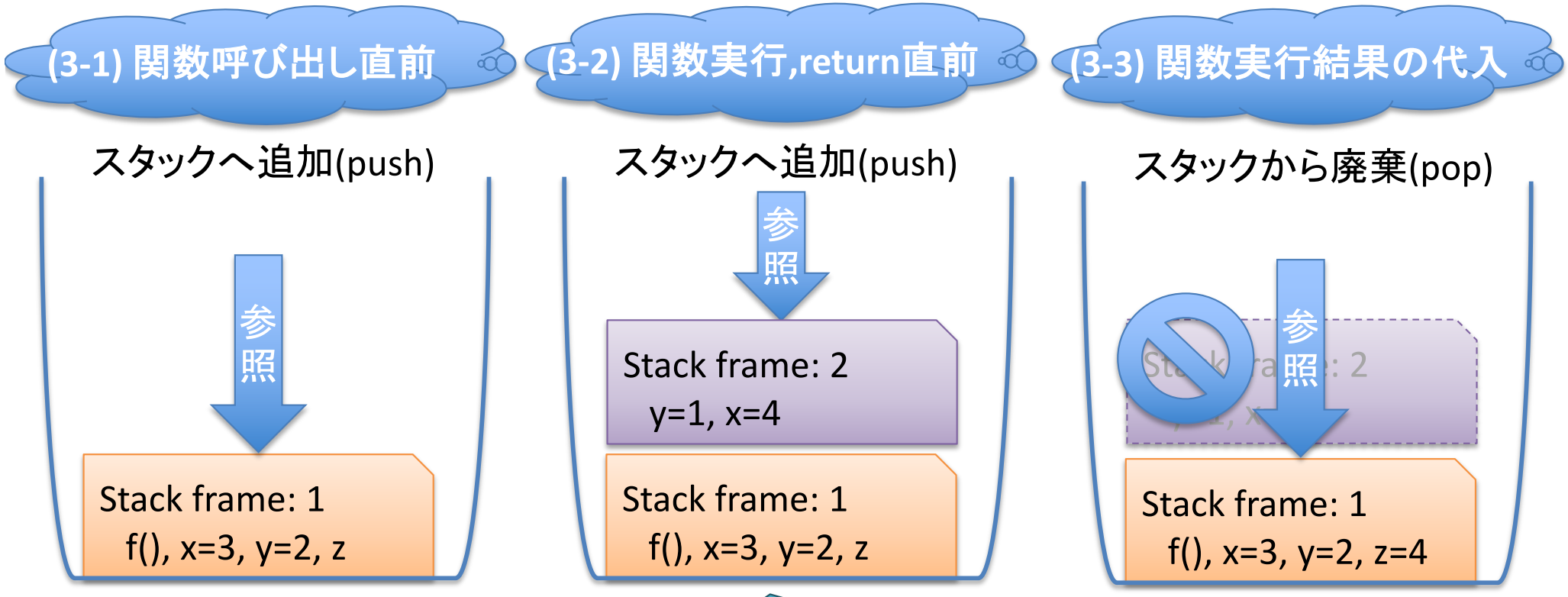
Stack frame: 2
y=1, x=4

(3-3) 関数実行結果の代入

Stack frame: 1
f(), x=3, y=2, z=4

参考: 基本的なデータ構造,
https://ja.wikipedia.org/wiki/データ構造

「スタック」構造=Last-in, First-out (LIFO)



関数を呼び出すたびに新しいStack frameが作られ、そこに局所変数が保存される。

Stack Frame のポイント

- トップレベル（インタプリタ起動時、もしくはスクリプトファイル実行時の最初のブロック）では、全ての関数名・変数名をトレースし、最初のスタックフレームで紐付けされる。
- 関数が呼び出されると、新しいスタックフレームが作られる。
  - ここでの処理は、スコープ外にあるスタックフレームには影響を及ぼさない。（例外あり）
  - 関数が終了すると、スタックフレームは廃棄(pop)される。
    - 廃棄しないとメモリに残り続るため、無駄。
  - 現スタックフレームに記録されていない名前が参照されると、「呼び出し元のスタックフレーム」を検索する。この遡り検索をトップレベルまで繰り返しても見つからない場合、NameErrorとなる。

ループ処理の例 (2.4節の改良版)

http://ie.u-ryukyu.ac.jp/~tnal/2017/prog1/loop.py

```
# スライムのHPが0より大きい間タコ殴りにするゲーム

import random

def encount_enemy():
    hitpoint = random.randint(3, 7)
    return hitpoint

enemy_hitpoint = encou                my()
print('スライムに遭遇し                enemy_hitpoint, ')

while (enemy_hitp
    attack = random
    enemy_hitpoint
    print('あなたの攻
HP=', enemy_hitpo

print('スライムを倒
```

4週目のwhile文コード例

(1) 関数encount_enemy()で参照してる「random」は、この関数内では定義されていない。

(2) 関数実行中のスタックフレームに存在しないため、トップレベルのスタックフレームを参照する。

(3) トップレベルのスタックフレームには、「import random」で読み込んだrandomモジュールが登録されており、これを関数encount_enemy()でも利用する。

プログラミング1

(第6回) デバッグ実行、関数とスコープ、仕様、ユニットテスト、モジュール

1. Chapter 1 Turing Complete
2. Chapter 4.1.2 Keyword Arguments and Default Values
3. Chapter 4.1.3 Scoping
4. Chapter 4.2 Specifications
    1. docstring + doctest (教科書にありません)
5. Chapter 4.5 Modules
6. 演習（恐らく時間なし）
7. 宿題

関数を呼び出しとスタックフレームの追加、名前空間の遡り参照を理解しよう。

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

4.2 Specifications (仕様, ドキュメント)

- コメント文
  - 「#」から後の文
- docstring形式によるコメント
  - 「"""～"""」で囲った、複数行に渡るコメント。
  - インデントでブロックを揃えること。
  - 簡易ドキュメントとしての側面

```
% curl https://ie.u-ryukyu.ac.jp/~tnal/2018/prog1/week6_doctest.py -O
% pydoc -w week6_doctest
% open week6_doctest.html
```

pydoc:
ドキュメンテーション・ツール
＊引数はモジュール名
（ファイル名ではない）

参考: Example Google Style Python Docstrings
http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html

doctestによるユニットテスト (教科書にありません)

- ユニットテスト (Unit Test, 単体テスト)
  - 関数が想定通りに機能するかを検証。
  - print出力して目視確認ではなく、「想定した入力を与えたら、想定した結果が得られること（成功or失敗）」を確認。
- doctestによるユニットテスト
  - 実行できるドキュメント（docstring）
  - 例
    - 通常のスクリプト実行

      % python week6_doctest.py
    - ユニットテストの実行

      % python -m doctest week6_doctest.py -v

通常実行ではユニットテストは実行されない。（開発中に確認するもの）

doctest実行例

```
oct:tnal% python -m doctest week6_doctest.py -v
Trying:
    add(1, 2)
Expecting:
    3
ok
Trying:
    add(-1, 3)
Expecting:
    2
ok
Trying:
    add(0, 0.5)
Expecting:
    0.5
ok
```

テスト1

テスト2

テスト3

(右に続く)

2つのアイテムにある全てのテストをパスした（想定通りだった）
・1つのテストが__main__にあった。
・2つのテストが__main__.addにあった。

(左の続き)

```
2 items passed all tests:
   1 tests in __main__
   2 tests in __main__.add
3 tests in 2 items.
3 passed and 0 failed.
Test passed.
```

doctestの注意点

- 「インタプリタでの出力と完全一致」じゃないと「想定通り」と見做してくれない。
  - 「1」と「1　」は違う
  - [1, 2]と[1,2]は違う
    - doctestはスペースの有無も厳密にチェック。
    - str型オブジェクトとしての判定。

プログラミング1

(第6回) デバッグ実行、関数とスコープ、仕様、ユニットテスト、モジュール

1. Chapter 1 Turing Complete
2. Chapter 4.1.2 Keyword Arguments and Default Values
3. Chapter 4.1.3 Scoping
4. Chapter 4.2 Specifications
    1. docstring + doctest (教科書にありません)
5. Chapter 4.5 Modules
6. 演習（恐らく時間なし）
7. 宿題

ドキュメンテーションとユニットテストを用いた開発に慣れよう。

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

4.5 Modules (モジュール, 部品)

- コードを保存したファイル = モジュール
- 使用例1

```
% python
>>> import week6_doctest
>>> week6_doctest.add(1, 2)
3
```

自作モジュールをimportで読み込む

「モジュール名.関数名」でモジュール内の部品を利用。

- 使用例2

```
% python
>>> import week6_doctest as wd
>>> wd.add(1, 2)
3
```

モジュール名に別称を付ける

- 使用例3

```
% python
>>> from week6_doctest import *
>>> add(1, 2)
3
```

モジュール内の関数を指定して読み込む。「*」は全ての部品。

- 読み込んだモジュールはhelp()でドキュメントを参照できる。

```
>>> import week6_doctest as wd
>>> help(wd)
```

pydocで読めるドキュメントと、内容は同じ。

プログラミング1

(第6回) デバッグ実行、関数とスコープ、仕様、ユニットテスト、モジュール

1. Chapter 1 Turing Complete
2. Chapter 4.1.2 Keyword Arguments and Default Values
3. Chapter 4.1.3 Scoping
4. Chapter 4.2 Specifications
    1. docstring + doctest (教科書にありません)
5. Chapter 4.5 Modules
6. 演習（恐らく時間なし）
7. 宿題

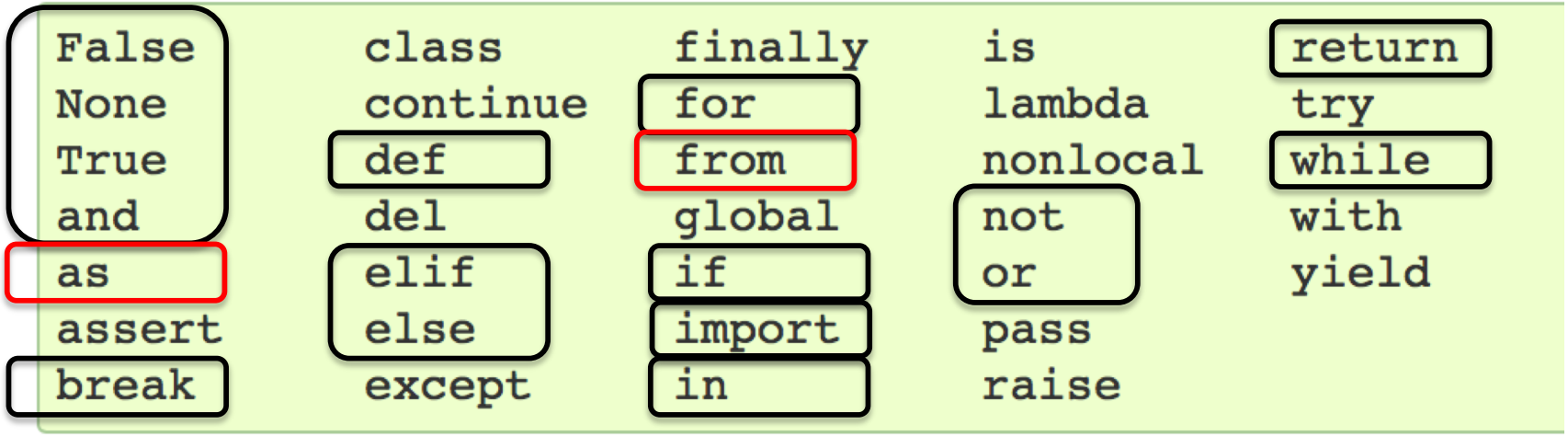
from, importによるモジュール読み込みを使えるようになろう。

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

Reserved words, 予約語

https://goo.gl/rEzdAN

- 一覧（赤丸は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	

まとめ

- スタックフレームとスコープに基づく名前空間の参照
  - 今書いてるスコープで参照されるスタックフレームはどれか？
  - 大原則
    - 関数を呼び出すと新しくスタックフレームが追加される。
    - 関数の処理を終えると、スタックフレームは破棄される。
    - 名前空間は現スタックフレームから遡って検索する。
- from, import によるモジュール読み込み
- docstringによるドキュメンテーション
  - 自由に記述できるが、できるだけガイドに従おう。
- doctestによるユニットテスト
  - 動作確認＋実行できるドキュメント。

プログラミング1

(第6回) デバッグ実行、関数とスコープ、仕様、ユニットテスト、モジュール

1. Chapter 1 Turing Complete
2. Chapter 4.1.2 Keyword Arguments and Default Values
3. Chapter 4.1.3 Scoping
4. Chapter 4.2 Specifications
    1. docstring + doctest (教科書にありません)
5. Chapter 4.5 Modules
6. 演習（恐らく時間なし）
7. 宿題

コンピュータの原型である万能チューリングマシンについて調べてみよう。

関数の引数にはデフォルト値を設定できる！

関数を呼び出しとスタックフレームの追加、名前空間の遡り参照を理解しよう。

ドキュメンテーションとユニットテストを用いた開発に慣れよう。

from, importによるモジュール読み込みを使えるようになろう。

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

演習5まで終わったペアは、進捗報告してから他ペアの様子を眺めてみよう。（第2のobserverしよう）

演習

演習1～4: 初めてのペア・プログラミング
演習5: 数当てゲーム1 (大小ヒント付き)を実装してみよう

宿題

- 復習: 適宜(これまでの内容)
  - レポート課題2の良レポートの「良さ」を真似よう。
- 予習: 教科書読み
  - 4章
    - 4.3 Recursion
    - 4.4 Global Variables
- 復習・予習(オススメ):
  - プログラミングスキルチェック　＊レベル設定のある課題集
    - https://paiza.jp/challenges/info

参考文献

- 教科書: Introduction to Computation and Programming Using Python: With Application to Understanding Data
- Python 3.5.1 documentation, https://docs.python.org/3.5/index.html
- チューリング・マシンとコンピュータ工学, http://www.slideshare.net/junpeitsuji/ss-57954980
- doctest — Test interactive Python examples, https://docs.python.org/3/library/doctest.html
- Example Google Style Python Docstrings, http://sphinxcontrib-napoleon.readthedocs.io/en/latest/example_google.html