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せん。「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
    2. [教科書] 2.1.2 変数と代入
    3. Reserved words, 予約語
3. 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
4. 変数名・ファイル名の命名規則
5. マニュアルの参照
6. 演習
7. 宿題

教科書に書かれていないか、後回しになってる内容

講義ページ: http://ie.u-ryukyu.ac.jp/~tnal/2016/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."

Imperative knowledge (命令的知識)

- "how to" knowledge, or recipes for deducing information. (情報を導くためのハウツーやレシピ)
- 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 untile g*g is close enough to 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 though a set of well-defined states and eventually produce an output.

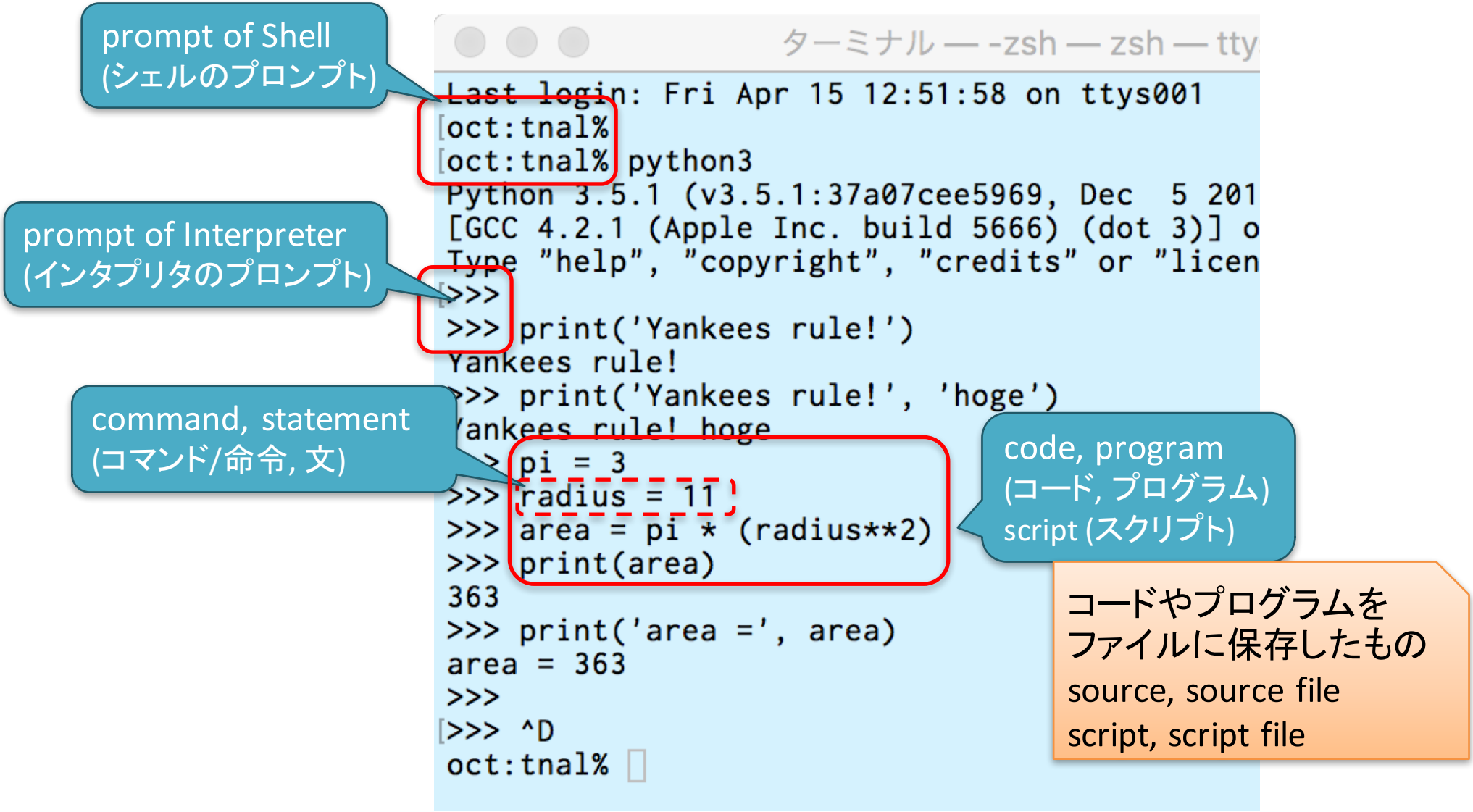
Chapter 2 -- 2.1.2までの補足

Glossaries, 用語集1, 2

[教科書] 2.1.2 変数と代入

Reserved words, 予約語

Glossaries, 用語集 1



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)

variable
(経数)

assignment
(代入文)

comparison (relational) operator
(比較演算子)

```
>>> 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'>
>>>
```

int: integer (整数)
float: floating point number (浮動小数点数)
str: string (文字列)
bool: boolean values, True/False (ブール型)
None: lack of value, (空のデータ)
See also,
https://docs.python.org/3.5/library/stdtypes.html#built-in-types

a **and** b: a, b共にTrueの場合にTrueを返す
a **or** b: a, bどちらかがTrueの場合にTrueを返す
not a: aがFalseの場合にTrue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string。

[教科書] 2.1.2 変数と代入

- 「=」は変数とオブジェクトを紐付ける。
  - 「=」の右辺を評価(evaluate)し、その結果を左辺に代入(assignment)する。
  - コード例

```
1 pi = 3
2 radius = 11
3 area = pi * (radius **2)
4 print(area) #-> 363
5 radius = 14
6 print(area) #-> 363
7 area = pi * (radius **2)
8 print(area) #-> 588
```

変数areaは右辺の評価結果を保存する

radiusを変更しても、変数areaには影響がない

変数areaは7行目の「=」により、右辺の評価結果が保存される

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	

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（拡張子は「.py」）に保存。
  - 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。

```
% python3 test.py
```

冒頭の「%」は、シェルのプロンプトを意味するので、実際にタイプするコマンドは「python3 test.py」

スクリプトを書いて動かしてみよう

- やってみよう
  - 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。
  - これをスクリプトファイル「test.py」として保存するには、エディタ（emacs）コマンドを、ファイル名を指定して起動する。

```
% emacs test.py
```

  - emacsでファイルtest.pyに「print('hello!')」と書いて、保存(Ctrl-x, Ctrl-s)する。
  - 保存し終えたら、エディタを終了(Ctrl-x, Ctrl-c)する。
  - シェル上で次のようにスクリプトを実行する。

```
% python3 test.py
```

スクリプト vs.インタプリタ

- ポイント
  - コードをスクリプト（ファイル）に保存する。
  - ファイルに保存するにはエディタを使う。
  - スクリプトを実行するには「python3 ファイル名」として、シェル上で実行する。
  - 作業中は「ターミナル」「エディタ」「インタプリタ」を切り替えながら作業することになる。（複数ターミナル立ち上げるのも手）
- 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再現することができる。
- 欠点
  - 初学者だと、1行ずつコードの実行結果を見ながらの方が書きやすい。
- オススメ
  - インタプリタで大まかな流れを確認し、ある程度コード全体の見積もりができたらスクリプトを書く。

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」？
    - 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）: lower_with_under
      - Local Variables: lower_with_under

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/3.5/index.html
- help()コマンド
  - help(print)
- Google先生
  - e.g.,「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。

演習

初めてのレポート
ペア・プログラミング

演習

- 初めてのレポート
  - https://ie.u-ryukyu.ac.jp/~tnal/2016/prog1/week2-ex.html
  - レポートの作成手順は授業ページを参照
- ペア・プログラミング
  - https://ie.u-ryukyu.ac.jp/~tnal/2016/prog1/week2-pair-programming.html
  - driver, observer (navigator)

宿題

- 復習: 適宜
- 予習1: 教科書読み
  - 2章
    - (スキップ) 2.1.3
    - 2.2 Branching Programs
    - 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
  - Python入門編1:プログラミングを学ぶ(全9回)
  - https://paiza.jp/works/python3/primer
    - やれる範囲でok

参考文献

- 教科書: Introduction to Computation and Programming Using Python, Revised And Expanded Edition
- Python 3.5.1 documentation, https://docs.python.org/3.5/index.html
- Google Python Style Guide, https://google.github.io/styleguide/pyguide.html
- ペアプログラミングのやりかた, http://goo.gl/ZWtlW
- (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