復習

プログラミング1

(第4回) 関数の利用2、ループ処理(while文)

1. Chapter 4.1.1の補足2
    1. 関数とローカル変数
2. Chapter 3.1の補足
    1. Iteration, looping (反復処理)
    2. ループ処理の例、実行例
    3. 3種類の処理流れ制御
3. 演習
4. 宿題

関数内の変数と、関数外の変数はスコープが異なることに注意。

繰り返させたい処理をブロックで指定し、繰り返し条件を設定。

処理の流れは逐次・条件分岐・反復処理の3タイプのみ。

基本道具
- 型(int,float,str,bool)
- 演算(数値・文字列・比較・論理)
- フロー制御(if, while, for)
- 関数定義(def)

講義ページ: http://ie.

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

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

Chapter 3.2, 3.4の補足

3.2 For Loops

3.4 A Few Words About Using Floats

3.2 For Loops (反復)
＊2つ目の反復制御

「range型オブジェクト」の中身を確認したいなら、list型にキャストしよう。

```
# 確認例
data = range(1, 4)
print(list(data))
```

range ()関数
range(stop): 0～stop-1までの全int型オブジェクトを生成
range(start, stop): start～stop-1

```
x = 4
for i in range(0, x):
    print(i)
```

for文
「in ～」で指定されたシーケンス集合（連続したデータ）に対して、
(1)1つずつ要素を取り出し、
(2)その要素を対象としてブロックを実行（反復処理）。
(3)全要素に対して(2)を実行し終えたらfor文を終了。

シーケンス集合の例: str, range, list

シーケンス集合の例1（str型オブジェクト）

```
# コード1
# 文字列を1文字ずつ処理。
for c in 'abc':
    print(c)

# 結果1
a
b
c
```

```
# srt型オブジェクト

>>> len('abc')
3
>>> 'abc'[0]
'a'
>>> 'abc'[1]
'b'
>>> 'abc'[2]
'c'
>>> enemy = 'naltoma'
>>> enemy[0]
'n'
```

シーケンスなら、[index]で順番を指定して参照できる。

2.3 Strings（文字列の補足）

'文字列'[0] : 文字列の0番目
＊指定する順番を Index (インデックス) と呼ぶ。
＊Indexは0番目から数える。
＊範囲外にはアクセス出来ない（index out of range）。
＊Indexがマイナス指定されると、後ろから数える。

Tips: コマンド実行時の出力を読もう
「IndexError: string index out of range」

'文字列'[x:y] : slicing (スライス処理)
文字列のx番目からy番目までを切り出す

変数に代入されてる時も同じ操作が可能

len(): 文字列等、シーケンスの長さ（要素数）

```
>>> len('abc')
3
>>> 'abc'[0]
'a'
>>> 'abc'[2]
'c'
>>> 'abc'[3]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> 'abc'[-1]
'c'
>>> 'abc'[-2]
'b'
>>> 'abc'[1:3]
'bc'
>>> 'abc' == 'abc'[:]
True
>>> 'abc' == 'abc'[0:len('abc')]
True
>>>
>>> string = 'abc'
>>> string[0]
'a'
>>>
```

シーケンス集合の例2（list型オブジェクト）

list (リスト) 型オブジェクト
- 順序付けられたオブジェクト集合。
- 「リスト名」＝集合名
- 「リスト名[インデックス]」＝指定した順番のオブジェクト。
  - e.g., scores[0] = 80

```
# コード2
scores = [80, 100, 0]
sum = Index = 0
for score in scores:
    sum += score
    print('sum = {0}'.format(sum))
    index += 1

average = sum / len(scores)
print('average = ' + str(average))
```

```
# 結果2
sum = 80
sum = 180
sum = 180
average = 60.0
```

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。

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

3.4 A Few Words About Using Floats
(浮動小数点数を使う際の補足)

```
# コード例3 (教科書版を少し編集)
x = 0.0
for i in range(10):
    x = x + 0.1

if x == 1.0:
    print('{0} = 1.0'.format(x))
else:
    print('{0} != 1.0'.format(x))

# 実行結果3
0.9999999999999999 != 1.0
```

コンピュータにおける数：2進数

丸め誤差(rounded error):
整数は適切に表現できるが、小数点のある数は必ずしも表現できない。

参考:
【5分で覚えるIT基礎の基礎】ゼロから学ぶ2進数　第4回: http://goo.gl/xvrN6n
倍精度浮動小数点数: https://ja.wikipedia.org/wiki/倍精度浮動小数点数

大きすぎる数字と小さすぎる（0より小さい）数を同時に扱うのは難しい

```
# 1000桁の整数に0.1を足してみる
>>> bignumber = 10**1000
>>> bignumber += 0.1
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
OverflowError: int too large to convert to float

# 100桁の整数に0.1を足してみる
>>> bignumber = 10**100
>>> bignumber += 0.1
>>>
>>> print(bignumber)
1e+100
>>> print('{0:f}'.format(bignumber))
100000000000000015902891109759918046836080856394528138978132755774783877217038106081346998568151 04.000000
```

- intなら1000桁でも適切に表現できる。
- float演算しようとするとOverflow(≒桁あふれ)。

指数表記 (exponential notation)
1e+100 = 1 * 10**100
0.0001 = 1 * 10**(-4) = 1e-04

これは誤差で片付けられる値？

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

小数を使う際には**丸め誤差**と**桁あふれ**に注意。

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

if文、ループ文の補足

elif, continue, break

if文補足（elif）

```
# 採点結果に応じてA～Fに振り分けるコード（途中）
score = 80
if score >= 90:
    eval = 'A'
elif score >=80:
    eval = 'B'
else:
    eval = 'others'

print(eval)    # -> B
```

Else時に、改めて条件分岐させたい場合は「else if を省略したelif」を使う。

Loop補足1（continue）

```
# レポート出した人（0点以外）の平均点を出したい。
scores = [80, 100, 0, 60]
sum = count = 0    # countはレポートを出した人数
for score in scores:
    if score == 0:
        continue
    sum += score
    count += 1

average = sum/count
print(sum, count, average)    # -> 240 3 80.0
```

デフォルトでは、スコアをsumに加算し、人数カウントを1増やす。ただし、score==0の場合には未提出扱いとして、加算処理もカウント処理もしたくない。この場合は処理せず、次のシーケンスに移り、次の要素に対して処理を継続（=continue）させたい。

Loop補足2（break）

```
# レポート出した人（0点以外）の平均点を出したい（バグ有り）
scores = [80, 100, 0, 60]
sum = count = 0    # countはレポートを出した人数
for score in scores:
    if score == 0:
        break
    sum += score
    count += 1

average = sum/count
print(sum, count, average)    # -> 180 2 90.0
```

先程と同様のコードで、continueをbreakに変更。
この場合、ループブロック自体から抜け出し（break）、その後のループ処理を終了する。

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

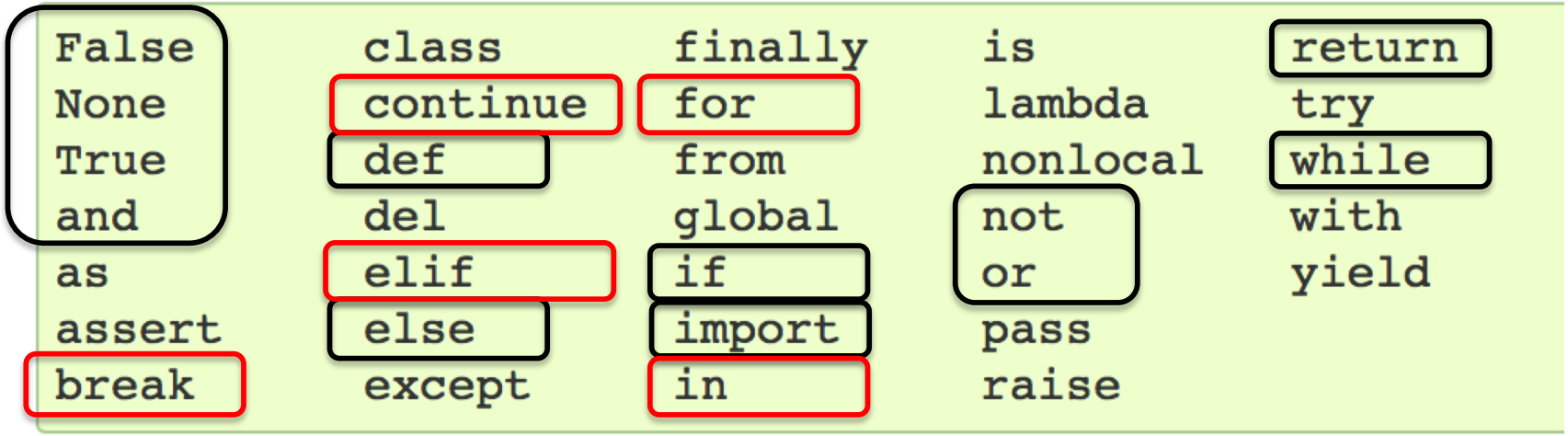
細かな制御方法を理解しよう

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

Reserved words, 予約語

https://goo.gl/rEzdAN

- 一覧（赤丸は今回出てきた予約語）



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/

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)
- 予習: 教科書読み
  - 4章
    - 4 Functions, scoping, and abstraction (冒頭1ページ)
    - 4.1 Functions and Scoping
      - (4.1.1 Function Definitions) #終了済み
      - 4.1.2 Keyword Arguments and Default Values
      - (4.1.3 Scoping) #終了済み
    - 4.2 Specifications
    - (4.3 Recursion) スキップ
    - (4.4 Global Variables) スキップ
    - 4.5 Modules
- 復習・予習(オススメ): paiza, progate

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
- str.formatメソッド, http://docs.python.jp/3/tutorial/inputoutput.html
- 【5分で覚えるIT基礎の基礎】ゼロから学ぶ2進数　第4回 2進数で小数を表す方法, http://itpro.nikkeibp.co.jp/members/ITPro/ITBASIC/20020624/1/?rt=nocnt
- 倍精度浮動小数点数, https://ja.wikipedia.org/wiki/倍精度浮動小数点数