プログラミングII

黒瀬　浩

kurose@neptune.kanazawa-it.ac.jp

OH: 講義の前後，Eメール問合せ，月3限21-405

居室　67・121

第1回復習　辞書，辞書のメソッド

```
d = dict(a=1, b=2, c=3, d=4)        # 変数dとキーdは別物
```
(d = {"a":1, "b":2, "c":3, "d":4} と書いても上と同じ)

キー一覧	d.keys() ⇨*1	['a', 'b', 'c', 'd']
値一覧	d.values() ⇨*1	[1, 2, 3, 4]
ペア一覧	d.items() ⇨*1	[('a',1), ('b',2), ('c',3), ('d',4)]
要素数	len(d)	

1つづつ処理する書き方

```
for k,v in d.items():
    print("key=", k, "val=", v)
# items()メソッドで得られたタプルの1個目が変数kに，2個目が変数vに入る
```

並び替え(1番上の変数dを定義して以下を確認せよ)

キー昇順	sorted(d.items())
キー降順	sorted(d.items(), reverse=True)
値昇順	sorted(d.items(), key=lambda x:x[1])
値降順*2	sorted(d.items(), key=lambda x:x[1], reverse=True)

*1: 正確には，list(d.keys()) のようにリスト化が必要

*2: lambdaはその場のみで有効な関数を定義する(後述)

第2回復習 map, reduce, filter　　無名関数lambda

map　全ての要素に同じ演算を行う
reduce 要素を順に演算し集約する(functoolsパッケージ)
filter　条件に合うものを残す

上記は，第1引数に演算の無名関数を，第2引数に対象を指定する
いずれも，for文やlist()などを使わないと値が解らない(遅延評価)

```
map( lambda x: x*x, range(1,11) )    # 数列をそれぞれ2乗
import functools as ft # recude はimportが必要
ft.reduce( lambda x,y: x+y, range(1,11) ) # 数列の総和
filter( lambda x: x%2==0, range(1,11) ) # 条件による抽出
```

map,reduce,filterはリスト内包や関数でも実装可能

```
[x*2 for x in range(11)]                    # 2倍の数列
sum( range(11) )                            # 総和
[x for x in range(11) if x%2==0]            # 偶数のみ残す
```

math.sin()は，引数は数値だがnumpy.sin()はリストを渡せる map機能あり

第3回復習

イテレータ(iterator)

- iter()関数　で1つづつ値を返すオブジェクトを作れる
- next(イテレータオブジェクト)　で1つづつ値を取り出せる
- 終わりまでいくとStopIterationエラーが返る
- forはイテレータを取り出す動作を行う

ジェネレータ(generator)

- 関数で定義するが，returnでなくyieldで値を返す
- 関数は終了しないので関数内の変数は保持される
- 関数でやることがなくなる（またはreturn)でジェネレータは消滅

range()関数　整数列生成

- 引数1個　0から終了の1つ前まで
- 引数2個　開始から終了の1つ前まで
- 引数3個　開始から終了の1つ前まで指定間隔ごと

可変長引数

- 引数の数が異なっても受けられる
- 受け取る変数に*をつける（*の付いていないものより後に指定)
- タプルで渡される

第4回復習

処理継続できないエラーが出るとプログラム強制終了となる

```
try：
    試す処理
except:
    例外発生時の処理
```

で例外発生時に処理を継続できる

```
except 例外種類:
    発生した例外ごとの処理をかく
else:
    例外が発生しなかった場合の処理を書く
finally:
    例外が発生しても，しなくても実行する処理を書く
```

`raise` 例外種類（メッセージ） で例外を意図的に発生できる

第5回復習 パッケージ

パッケージを読み込み使えるようにする

```
import math
```

math内の関数や変数，定数が使えるようになる

例 `math.sin( math.pi )`

パッケージに別名をつける

```
import math as m
```

例 `m.sin( m.pi )`

パッケージ名を省略したい

```
from math import *
```

例 `sin( pi )`

パッケージを追加する

pip コマンド

conda コマンド(anacondaの場合)

easy_install (パッケージをダウンロードしてインストールする場合)

pythonのパッケージをまとめたサイト

pypi https://pypi.org

注意：importするパッケージ名と同じ名前のpythonファイルをつくると自作のほうがimportされ，本来のパッケージ機能が使えない

第5回　復習 ファイル

ファイルを使う前にopenする

```
ファイル変数 = open(ファイル名, モードなどのオプション)
変数 = ファイル変数.read()  # 変数に読んだデータが入る
ファイル変数.write( データ ) # print()のように↲は付かない
ファイル変数.close()        # 以後，ファイル変数は使えない
```

アクセスモード

```
open()の第2引数または mode= で文字列で指定
最初の文字    r(read), w(write), a(append)のいずれか
次の文字      t(text), b(binary)
例      'r', 'rb', 'w'
```

withブロック

```
with open(ファイル,オプション) as ファイル変数
        ファイル変数を使った操作(read,write等)
ブロックを抜けるとファイル変数は使えない(closeされる)
```

正規表現(Regular Expression)

汎用的な検索，置換を行いたい

数字（列）で検索したい

0か1か2か3...9 を表す記法

(0|1|2|3|4|5|6|7|8|9)　　|　はいずれか

[0-9]　　0から9の文字集合

¥d　　数字1字（¥Dは数字以外）

繰り返し記号

? オプション　あってもなくても良い

+ 1回以上の連続

* 0回以上の連続

変数名は英字で始まり，英数字が何個連続してもよい（0回でも良い）

[A-Za-z][A-Za-z0-9]*

正数は符号がついても無くても良い，1から9の後数字が0回以上繰り返す

(¥+|-)?[1-9][0-9]*

符号+の前の¥は繰り返し記号+と区別するため

正規表現をpythonで使う

```
# stは検査対象文字列
# リストregは検査条件を正規表現で指定，数字列，大文字で始まる英単語
import re
def main():
st='''¥
924-8383 076-274-7173 3-1 Yatsukaho, Hakusan-City
'''
reg=[ r'¥d+', r'[A-Z][a-z]+']
for i in reg:
    print('¥n'+i)
    print('start end matched')
    print('----- ----- ------------------')
    for j in re.finditer( i, st ):
        print( '%5d %5d %-s'%(j.start(), j.end(), j.group()) )

main()
```

正規表現の検索パターン文字列を r'･･･'とするのは
¥nなどを↲と認識させないため

実行結果

数字列

```
¥d+
start   end   matched
-----  -----  ------------------
    0      3  924
    4      8  8383
    9     12  076
   13     16  274
   17     21  7173
   22     23  3
   24     25  1
```

startは文字列がマッチした開始場所
endはマッチ終了+1

正規表現で　+(1回以上),*(0回以上）は
マッチし続ける限り最長のものを探す

最長一致検索(longest match)

最初が大文字の単語（固有名詞)

```
[A-Z][a-z]+
start   end   matched
-----  -----  ------------------
   26     35  Yatsukaho
   37     44  Hakusan
   45     49  City
```

次回　試験（配点20点）

試験範囲：

辞書	教科書 3-2
無名関数, map, filter	教科書 3-5
イテレータとジェネレータ	教科書 3-6
例外処理	教科書 3-7
リスト内包表記	教科書 4-6
モジュール	教科書 4-1
ファイル処理とwith構文	教科書 4-3

教科書持ち込み可，PC持ち込み不可

遅刻，欠席に注意

第8回からJavaの学習になります

クイズ

- 時間内に提出
- クラス名列番号，氏名を明記すること

演習 プログラムリスト出力

自身のソースファイルを読み込み　各行に行番号を　整数4桁頭0詰め　で
別のファイル（拡張子.txt)に出力するプログラムをつくれ

ヒント：　一括して読んだデータは改行コードで分割する　か
　　行毎に読む（何行あるか不明なのでwhileで脱出する方法を確認）

出力例（端末アプリケーションから　type ファイル名↲ で確認できる)
文字化けする場合は,　コマンドchcp 65001(utf-8の場合)などで表示可

```
0001 def aaa(n):
0002     ''' これは出力例であり，2つのファイルopenが必要 '''
0003       for i in range(n):
0004           print(i)
0005
0006 aaa( int( input("n? ") ) )
0007                         (↲のみの行も行番号を表示すること)
```

レポート3

タイトル　　プログラミングII　レポート3 プログラムリスト
タイトルを上部中央
クラス名列，氏名を上部右
前ページの実行結果(ハードコピーでない場合は，等幅フォントを使用すること)

A4 1枚に収める(両面可，横長配置可)

画像のフォントサイズの注意
レポート本文のフォント10〜12ポイントと同じ程度の字の大きさ
(大きすぎないこと)
画像の縦横比を変えない
見難い配色は避ける

期限：　第8回開始時
注意：　不明点は提出前に確認すること