実践統計解析【第2回】

Pythonの文法_分岐と繰り返し

松田　史生[1]*・川瀬　雅也[2]

前回，Pythonの基本的な使い方を説明した[1]が，これだけではプログラムを書くことはできない．今回は，プログラムを書くうえで必要不可欠な分岐（正確には，条件分岐）と繰り返しの命令を説明する．プログラミングの上達のカギは，沢山のプログラムを書いて，失敗をすることだと筆者は思っている．また，プログラムの解説に，すべてが書かれているわけではなく，解説にないところも，プログラム例から学ぶことが重要である．

条件分岐

Aさん：プログラミング言語の基本といえば，条件分岐と繰り返しですよね．Pythonの文法になにか特徴があるんですか？

H研究員：Pythonのすごさはなんといってもインデント（字下げ）だよね．B君ってさ，数か月まえに自分が書いたRのスクリプトがぐちゃぐちゃであとから読んでもわかんなかったこととかない？

B君：しょっちゅうです．とくに条件分岐や繰り返しが入れ子になると，こんがらがってますね．

H研究員：条件分岐とは「もし，〜なら，○となり，そうでなければ，△となる」という形の処理だよね．

【例1-1】xに代入した数値が負の数なら0，正の数ならそのまま出力するRのスクリプト（よくない見本）

```
 x < -1
if (x < 0) {x =0 }
          print(x)
```

このプログラムをRで実行すると0と出力されるのだけど，各行の前にスペースがあってもなくても，{}で囲んだif文のブロックが1行でも複数行でもいい．自由にかけるのは楽でいいけど，あとで読み返すときとか，バグがあるときに混乱のもとになる．Pythonはこういう自由を認めてくれない．

【例1-2】例1をPython用に書き直したもの．

```
x = -1
if x < 0:
      x = 0
print(x)
```

#はコメントであることを示し，(:)はifでの処理の際，必須の書式である．

まず，同じ階層の行の左端は揃っていないといけない．勝手にスペースを入れてはだめ．次に，if文のブロックはタブでのインテンド（字下げ）で表現する．

〔エラーが出る例A〕

```
x = -1
if x < 0:
      x = 0
 print(x) #printの前に空白がある．
```

〔エラーが出る例B〕

```
x = -1
if x < 0:
x = 0 #if 文のブロックのインテンドがない
print(x)
```

Aさん：これは，いいかも．読みにくいスクリプトがそもそもかけないんですね．B先輩向けかも．

H研究員：ちなみに例1-2を実行したら表示されるのは？

B君：0ですよね．

【例2】合計が60以上なら"OK"，60未満なら"False"を出力するPythonスクリプト

```
S=10+20+15
      if S>=60:
          R="OK"
      else:
          R="False"
      print(S, R)
```

Aさん：これを実行した時の出力はelseが「そうでなければ」の意味なので，

```
>45, False
```

ですよね．

【例3】1〜100の乱数を用いて，90以上をS，80〜89をA，70〜79をB，60〜69をC，60未満をDとするプログラムを書くスクリプト

著者紹介　[1]大阪大学大学院情報科学研究科（教授）　E-mail: fmatsuda@ist.osaka-u.ac.jp
[2]長浜バイオ大学（教授）

```
import random　# randomモジュールの読み込み
  p=random.randint(0,100)　# 0–100の整数乱数の発生
  この書き方は0を含む101個の整数を意味している.

  if p>=90:
      R="S"
  elif p>=80:
      R="A"
  elif p>=70:
      R="B"
  elif p>=60:
      R="C"
  else:
      R="D"
  print(p, R)
```

H研究員：これを実行すると，乱数の値と判断の組合せが，47 D　や　81 A　というように示される.

条件を満たす間の繰り返し

H研究員：たとえば，ある方程式の解を見つけるような場合を考える．解が見つかるまで計算を繰り返す必要がある．このようなプログラムでは"while~　"を用いる.

【例4】1～11の間の3つの数の合計が17となる組合せを求めるスクリプト

```
import random　# randomモジュールの読み込み
    a = 0
    b = 0
    c = 0
    while (a+b+c) !=17: # a+b+cが17以外は繰り返し
      a= random.randint(1,11) # 1-11の整数乱数の発生
      b= random.randint(1,11)
      c= random.randint(1,11)
    print(a,b,c)
```

B君：実行すると（6 4 7）や（1 10 6）などの組合せが出力されますね．なんでだろ？

Aさん：while (a+b+c) !=17:というのは，a+b+cの値が17以外の時にループを回すという意味なので，a+b+c = 17の組合せが見つかったときにループが終了して，print(a,b,c)で結果が表示されるわけですね.

指定した回数の処理を繰り返す

H研究員：指定した回数の繰り返しの場合には“ for in ”を用いる.

【例5】0～9までの数字を表示させるスクリプト

```
for i in range(10):
    print(i)
```

Aさん：0～9の数字が縦に表示されました．range(10)なのに9で終わってしまうんですね.

H研究員：ここもpythonでよくつまずくポイントだね．range(10)は0から10個，つまり9までを表示してという意味だね．もし，range(4,8)とすると4から8-4=4個，4,5,6,7が縦に表示される（自分でやってみるといいよ）.

H研究員：じゃ，ここまで習ったif文，for in文を使って，モンテカルロ法による円周率を求めるプログラムを作ってみようか．こういう感じ.

乱数により0～1.0の間のxとyの数値を多数発生させる（0≦x≦1，0≦y≦1）.
半径1($x^2+y^2=1$)の円内の点の数を数える.
全点の数をN+1，円内の点の数をnとするとn/(N+1)が円の面積の1/4となるので，円周率(pi)を求めることができる.

B君：これをアルゴリズムにするんですね．えーと…

Aさん：こんな感じでしょうか？

```
nにゼロをセット
繰り返し回数Nをセット
N回繰り返す:
        x, yの値を乱数で生成
        もしx^2+y^2＜1だったら:
            nに1を追加
4*n/(N +1)を出力
```

H研究員：いい感じだね．これを命令に置き換えていけばいいよ．0-1の間の一様乱数を生成する方法を知りたかったら“一様乱数 Python”でググればすぐ見つかるから.

B君：numpy.random.rand()を使えとありますから，numpyモジュールをインポートすればいいんですね.

【例6】モンテカルロ法で円周率を求める

```
import numpy　# numpyモジュールの読み込み
n=0 # nにゼロをセット
N=input("number: ") #繰り返し回数Nを入力
N=int(N) #繰り返し回数Nを数値に変換
for i in range(N):
```

```
        x= numpy.random.rand () # x, yの値を乱数で生成
        y= numpy.random.rand ()
        if(x**2+y**2<=1): # もしx²+y²＜1だったら:
            n=n+1 # n に1を追加
pi=4*n/(N +1) #4*n/(N +1) を出力
print(N +1, pi)
```

N=1000とした時，pi=3.156
N=10000とした時，pi=3.1208
N=100000とした時，pi=3.13804

H研究員：となった．点を乱数で打つので，この数値は計算をするごとに異なってくる．一度試してみてね．
Aさん：今回は，いろいろ教えてくださりありがとうございました．
Bくん；研究データをPythonで頑張って解析します．
H研究員:今後，データ解析で困ったら「バイオインフォマティクス相談部会」に相談してね．待ってるよ．

実験のシミュレーション

C君：Aさん，あの，ちょっと今いいですか？こないだのセミナーで，先輩，メタボロームデータ解析の発表をされてましたよね．
Aさん:ばらつくデータの取り扱い方を考察したやつね．
C君：メタボローム解析では，【サンプルBのピーク面積値】/【サンプルAのピーク面積値】を算出する相対定量を行うのが一般的だ，と．
Aさん：そうそうそれで，サンプル間のピーク面積値のばらつきが相対標準偏差で10％くらいだから，そこから必要な実験の反復数について説明したやつね．
C君：すごくおもしろかったです．B先輩からは「とりあえず実験は3反復だ」と教わったんですけど，Aさんの説明でその理由がわかりました．
Aさん：わ！そう言ってもらえると嬉しいな．
C君：でも，ひとつ引っかかることがあって……．2つのピーク面積値を割り算して，相対定量値を計算するんですよね？それがばらつきに影響しないんですか？
Aさん：ほんとだ．じゃ，調べてみよっか．Pythonで．今ちょうど復習をしてたのよね．まずは反復数は3にして作ってみましょ．

【例7】相対定量のシミュレーション

```
import numpy # nunpyのインポート
n=3  # 反復数
sample_A = numpy.random.normal(1.0, 0.1, n) #平均値1, 標準偏差0.1の正規分布からn回ランダムサンプリングしたリストを生成する
sample_B= numpy.random.normal(2.0, 0.2, n) #平均値2, 標準偏差0.2の正規分布からサンプリング
rel = numpy.average(sample_B)/ numpy.average(sample_A) #【サンプルBのピーク面積値】/【サンプルAのピーク面積値】を算出
print(rel)
```

実行結果（数値が毎回少し変わる）

```
>>>
1.9582580023570477
```

C君：numpy.average()は平均値を計算する関数ですね．2.0に近い数字になってますが，1回だけしか計算してくれないし．
Aさん：まかせなさい．10000回繰り返して，その結果をリストに保存しましょ…….
C君：（しばらくして）先輩，あれ，どうしました？
Aさん：リストにデータを順番に保管していきたいんだけど，どうするんだったけ？
C君：そういう場合は「Python リスト　追加」で検索してみるといいそうですよ．
Aさん：append()を使えってでてきた．あと，標準偏差を計算するのはnumpy.std()みたいね．

【例8】相対定量を1万回繰り返す

```
import numpy # nunpyのインポート
n=3  # 反復数
list_A = [ ] #空のリストを作るおまじない．Aの平均値を保存する
list_B = [ ] #Bの平均値を保存する
list_rel = [ ] #B/Aを保存する
```

（注：上記[]の間に半角スペースを入れているが，見やすさのために入れており，本来ならスペースは入りません）

```
for i in range(10000):
        sA = numpy.random.normal(1.0, 0.1, n)
        sB= numpy.random.normal(2.0, 0.2, n)
        rel = numpy.average (sB)/numpy.average (sA)
        list_A.append(numpy.average(sA))#list_Aの最後にsAを追加
        list_B.append(numpy.average(sB))
        list_rel.append(rel)
#10000回の試行結果の平均値と標準偏差を表示
print("A", numpy.average(list_A), numpy.std(list_A))
print("B", numpy.average(list_B), numpy.std(list_B))
print("rel", numpy.average(list_rel), numpy.std(list_rel))
```

n=3のときの実行結果（数値が毎回少し変わる）
>>>
A 1.0003365002658988　0.05814165591750018
B 2.0007606003285012　0.11512307080399363
rel 2.006791446650835　0.16370240105126477

C君：Aの行には，Aの平均値（n=3）を1万回計算した時の平均値と，標準偏差が表示されているんですよね．平均値がおおよそ1なのでうまくいっているっぽいんですが，標準偏差はこれで正しいんでしょうか？確かめないといけないですね．
Aさん：確かに平均値の標準偏差は，もとの標本集団の標準偏差より小さくなるもんね．でもどうしたらいいんだろ．うーん．こういうのはB先輩に聞いても，頼りにならないし．
C君：n=3の平均を取るからややこしくなるので，n=1にしてみたらどうでしょうか？
Aさん：2行目のn=3をn=1にすればいいのね．

n=1のときの実行結果（数値が毎回少し変わる）
>>>
A 1.0009504244086636　0.09966095679847731
B 1.999366148874661　0.1988293879813715
rel 2.0179806991821505　0.2903994501621393

C君:AとBの標準偏差はほぼ0.1と0.2になってるから，プログラムはきちんとこちらの意図通りに書けたみたいですね．
Aさん：問題はrelよね．B/Aを計算すると，標準偏差が大きくなってない？
C君：確かに，相対標準偏差に直すと0.145だから，元のデータの0.1に比べて悪化してますね．
Aさん：昔，X教授から，誤差の伝播っていうのを教えてもらったんだけど……
C君：それで，先輩がプログラムを書いている間に，スマホで検索して調べてみたんですけど，すぐページがいくつも見つかりました．除算の誤差の伝播の式は

$$(M_B + e_B)/(M_A + e_A)$$
$$= M_B / M_A \pm \sqrt{(e_B / M_A)^2 + (e_A M_B / M_A^2)^2}$$

（ここで，Mは測定値，eは誤差，A，Bは，どちらのサンプルかを示している）
Aさん：なるほど！じゃ，$M_A = 1.0$, $e_A = 0.1$, $M_B = 2.0$, $e_B = 0.2$を代入したらいいのね…
C君：それで，スマホの電卓でさっき計算したんですけど，2.0 ± 0.283となりました．シミュレーション結果とおおよそ合いますね．おもしろい！
Aさん：そっか，2つのデータの相対値を計算するために除算すると，ばらつき（標準偏差）が大きくなっちゃうのね．
C君：それであの，セミナーの話に戻るんですけど……
Aさん：なるほど！私はあの時，相対値のばらつきも除算する前と同じとして，議論しちゃってたわ！間違いだったのね．ありがとう．教えてくれて嬉しいな．
C君：いや，データを解析するためにきちんと理詰めて考察してるA先輩がすごいです．でも，どうすればいいんでしょうか？
Aさん：相対値を計算すると，ばらつきがルート2倍になるんだったら，割り算をする前のAとBの平均値のばらつきを小さくすればいいのかな．
C君：じゃ，反復数をn=2にして試してみましょうよ．

n=2のときの実行結果（数値が毎回少し変わる）
>>>
A 1.0003905177358074　0.07043537538376017
B 1.9969789441527441 0.14296065661561533
rel 2.0062417162260653　0.2030769138796842

C君：あ，relの標準偏差が0.2になりましたね．
Aさん：平均値の標準偏差（標準誤差）は標本集団の標準偏差の$\sqrt{n}$分の1になるから．そうか，前回の報告では，n=3の反復数が必要ってことになったけど，本当はその2倍のn=6はいるってことね．そか，メタボローム解析用の実験を考え直さなくっちゃ．
C君：Pythonでシミュレーションをするといろいろな条件を簡単に試せるから楽ですね．でも，結果を見ているとrelの平均値や標準偏差の値がちょっと理論値とずれるのが気になるんですが…….
Aさん：ほんとだ，なんでだろ．ちょっと調べてみよっか？付き合ってくれたら晩御飯ごちそうするよ．
B君：お，Aさん，C君元気？さっきX教授からメールが届いたよ．来月号から登場するってさ．

文　献

1) 松田史生，川瀬雅也：生物工学，**97**, 226 (2019).

参考文献

大重美幸著「Python3　入門ノート」（ソーテック社）を参考にし，プログラムコードを一部改変して紹介した．

協　力

日本生物工学会バイオインフォマティクス相談部会