制御情報システム工学 4 年実験指導書

ソフトウェア・リテラシ 3: Script 言語 python 入門

熊本高等専門学校 制御情報システム工学科 藤本信一郎
fuji@kumamoto-nct.ac.jp

概要

本実験では, 制御情報システム工学科パソコンにインストールされた Linux 上で, Script 言語 python の基礎を学習する. 併せて,

目次

1 はじめに

1.1 本実験の目的

まず C 言語と比べて効率的なプログラム作成が可能な Script 言語 python の基礎を身に付ける. 今後様々な授業・実験では python を用いるので, 本実験では, その準備として python を学習する.

1.2 python とは

Script 言語 python は, C 言語などの compile 言語と比べて, 少ない命令で同等の処理を行なうことが可能である.[*1] python は, Windows [*2], Linux, Mac OS など複数の OS 上で実行可能である.

1.3 課題 0

python に関する下調べを行なうこと. 参考になりそうなページを検索し, その URL をファイルにまとめ, USB メモリや Dropbox などに保存すること.

2 python の基礎

制御情報システム工学科パソコン室では Linux 上で python の version 2, 3 が共に利用可能である. python の最新版は version3 であり, version2 とは**上位互換ではない**. 本実験では, Linux 上での python3 (python version3) の利用方法を解説する. 実用上, 最低限必要な python の以下の内容を学ぶ.

1. 変数, リスト (配列)
2. 制御文 (条件, 繰り返し)
3. ファイル入出力
4. 実行時の引数
5. 正規表現

2.1 python プログラムの記述と実行

python のプログラムは, C 言語プログラムと同様に kwrite などの Editor で記述する. python は Script 言語なので, python のプログラムの compile は不要である. また python のプログラムは python script と呼ばれることが多い.

まず簡単な python script の例とそれを実行する方法を説明する. script 例リスト 1 を **hello.py** というファイルに保存する.

リスト 1: 例題 1

```
#!/usr/bin/python3
```

[*1] ただし処理速度は遅い.

[*2] Windows 版は cygwin に含まれている. cygwin のことやインストール方法に関しては Web を検索してください.

```
# -*- coding: shift_jis -*- # Windows の場合

print('Hello world!') # 日本語
```

ここで先頭の"#!/usr/bin/python3"はこの script が python3 で処理 (インタープリット) されることを表しており. **それ以外の"#"から始まる行はコメント**である.

また 2,3 行目は Script の文字コードを指定しており, **Linux や Mac 上で Script を実行する場合には, 2 行目は不要です.**

この python3 script を Linux で実行するには, ターミナル上で図 1 のようにする.

```
$ chmod u+x hello.py
$ ./hello.py
```

図 1　python scirpt の実行

ここで, "$"はプロンプトと呼ばれ, 入力する必要はない. また**以下で "$" から始まる部分は, 端末 (ターミナル) で入力, 実行されるコマンド**を表す.

1 行目のコマンドで**ファイルに実行権を付加**している. 実行権の付加は一回行なえば良いので, 2 回目以降の実行の際に単に hello.py を実行すれば良い.

```
$ ./hello.py
```

図 2　python scirpt の実行 (2 回目以降)

2.1.1　練習課題

上記に従って, hello.py を作成し, 実行し, python script の作成と実行方法を身に付けなさい.

2.2　変数とリスト (配列) の取り扱い

本節では変数とリスト (配列) の取り扱いの取り扱い方法を説明する. script 例リスト 2 は変数とリストの値を初期化し, それらを表示する script である.

リスト 2: 例題 2

```
#!/usr/bin/python3
# -*- coding: shift_jis -*- # Windows の場合

a = 1            ## 整数型変数
```

```
## リスト (配列): 整数型, 文字列型
b = [1, 'test'] ## 値の初期化, b の要素数=2
b[0] = 2        ## リストの各要素の値の変更

## 標準出力 (端末) への出力
print( 'a = ', a)                    ## 変数
print( 'b          = ', b)          ## リスト全体
print( 'b[0], b[1] = ', b[0], b[1]) ## リストの各要素

## リストへの要素の追加
#b[2] = 6      ## エラー: 要素数を超えたリストへのアクセス
b.append(5.80) ## 浮動小数型の要素の追加, b の要素数=2+1=3

## Format 付き出力: 整数型, 文字列型, 浮動小数型
print('b[0]=%5d, b[1]=%5s, b[2]=%5.2f' % ( b[0], b[1], b[2] ))

b[0] = 'a' ## 型の変更: 整数型から文字列型
print('b[0]=%5s' % ( b[0] ))
#print('b[0]= %5d' % ( b[0] )) ## エラー: 型の異なる Format での出力
```

これを **var.py** というファイルに保存し, 実行すると図 3 のように表示される.

```
$ ./var.py
a =  1
b          =  [2, 'test']
b[0], b[1] =  2 test
b[0]=    2, b[1]= test, b[2]= 5.80
b[0]=    a
```

図 3　var.py の実行

この例のように python では, 変数とリストは以下のような特徴をもつ.

- C 言語とはことなり, 変数宣言する必要はない
- C 言語とはことなり, 変数を必ず初期化する必要がある
- 出力には print を用いて, C 言語の printf と使い方は似ている

2.2.1　練習課題

1. var.py を作成し, 変数とリストの取り扱い方法を確認せよ.
2. また#b[2] = 6 のコメント外して, Script を実行し, エラーメッセージを確認せよ.
3. 同様に最終行だけのコメント外して, Script を実行し, エラーメッセージを確認せよ.

2.3 制御文 (条件, 繰り返し)

2.3.1 if, for 文

本節では制御文 (条件 if 文, 繰り返し for 文) の取り扱い方法を説明する. script 例リスト 3 にあるように, python での if 文は C 言語と同様であるが, for 文は文法がやや異なる.

リスト 3: 例題 3-1

```
#!/usr/bin/python3
# -*- coding: shift_jis -*- # Windows の場合

b = [1, 'test', 5.80, 4] ## リスト (配列): 値の初期化, b の要素数=4

## i = 0,1,2,3 に対して繰り返し (0 から 4 ではない!)
for i in range(0,4):
    if ( i == 3 ):       ## i=3 なら
        continue         ## 次の i に対する処理へ
    elif ( i % 2 == 0 ): ## i が偶数なら
        ## 浮動小数として出力
        print('b[%d]=%5.2f' %(i,b[i]))
    else:
        ## 文字列として出力
        print('b[%d]=%5s' %(i,b[i]))
```

python では, 字下げ (インデント) により, ブロックを指定する. **インデント**はタブもしくはスペース (空白) で行う.

ただし**タブとスペースを併用しないこと.** 併用すると環境によっては想定外のブロック構造と python に解釈され, 実行時に意味不明のエラーが出力されることがある. (**特に kwrite を使っている人は注意すること**)

この script を **if-for.py** というファイルに保存し, 実行すると図 4 のように表示される.

```
$ ./if-for.py
b[0]= 1.00
b[1]= test
b[2]= 5.80
```

図 4 if-for.py の実行

2.3.2 練習課題

if-for.py を作成し, 制御文 (条件, 繰り返し) の動作を確認せよ.

2.3.3 2 重 For 文

script 例リスト 4 は 2 重 For 文の例である.

リスト 4: 例題 3-2

```
#!/usr/bin/python3
# -*- coding: shift_jis -*- # Windows の場合

b = [1, 'test', 5.80] ## リスト (配列): 値の初期化, b の要素数=3

## 二重 for 文
## i = 0,1,2 に対して繰り返し
for i in range(3):
    ## k = 0 から i に対して繰り返し
    for k in range(0, i+1): ## range(i+1) と同じ
        ## end='' --> 改行なし
        print ('b[%d]=' %(k), end='')
        ## str で文字列に変換
        print ('%5s, ' %(str(b[k])), end='')
    print('') ## 改行, i の繰り返しのみ
```

この script を **for2.py** というファイルに保存し, 実行すると図 5 のように表示される.

```
b[0]=     1,
b[0]=     1, b[1]=  test,
b[0]=     1, b[1]=  test, b[2]=   5.8,
```

図 5 for2.py の実行

2.3.4 Script の補足説明: ブロック

python では, 字下げ (インデント) により, ブロックを指定する. 図 6 の最初の for 文では, 続く 1 行だけが繰り返されるが, 二つ目の for 文では, 続く 2 行が繰り返される.

2.3.5 練習課題

for2.py を作成し, 制御文 (繰り返し) の動作を確認せよ.

2.4 演習課題 1

30 以下の偶数の 2 乗の和を計算・表示する script を作成せよ.

```
for k in range(0, i+1):
    print 'b[%d]=' %(k),   ## 繰り返される
print '%5s,' %(str(b[k])), ## 繰り返されない

for k in range(0, i+1):
    print 'b[%d]=' %(k),       ## 繰り返される
    print '%5s,' %(str(b[k])), ## 繰り返される
```

図 6　ブロック

2.5　ファイル入力

本節ではファイルからの入力の取り扱い方法を説明する. リスト 5 を **io1.py** というファイルに保存する.

リスト 5: 例題 4-1

```
#!/usr/bin/python3
# -*- coding: shift_jis -*- # Windows の場合

input = "input.csv"  ## 入力ファイル名
FD = open(input,'r') ## 読み込みモードでファイル open
for line in FD:
    ## 各行の改行を除去
    line = line.rstrip('\n')
    line = line.rstrip('\r')
    ## 各行をコンマで分割し, リスト tmp へ代入
    tmp = line.split(",") ## 文字列のリスト
    i = int(tmp[0]) ## 文字列を整数へ変換
    for j in range(4): ## j=0 から 3
        print("%d 行目: tmp[%d] = %6s" % (i, j, tmp[j]))
FD.close()  ## ファイル FD を閉じる
```

```
1, 0.1, 5, 1st
2, 0.8, 8, 2nd
3, 0.8, 7, 3rd
4, 0.2, 2, 4th
```

図 7　io1.py の入力データ input.csv

この script を **io1.py** というファイルに保存し, 入力ファイル input.csv 図 7 に対して, io1.py を実行すると図 8 のように表示される.

```
$ ./io1.py
1 行: tmp[0] =      1
1 行: tmp[1] =    0.1
1 行: tmp[2] =      5
1 行: tmp[3] =    1st
2 行: tmp[0] =      2
2 行: tmp[1] =    0.8
2 行: tmp[2] =      8
2 行: tmp[3] =    2nd
3 行: tmp[0] =      3
3 行: tmp[1] =    0.8
3 行: tmp[2] =      7
3 行: tmp[3] =    3rd
4 行: tmp[0] =      4
4 行: tmp[1] =    0.2
4 行: tmp[2] =      2
4 行: tmp[3] =    4th
```

図 8　io1.py の実行結果

2.5.1　script の補足説明: continue, break

リスト 5 で用いられている for 文を使えば, open で開いた入力ファイルの各行に対して, 一連の処理を行なうことができる. 次の入力行へ処理を飛ばしたい場合は”continue”文を用い, 以降の入力行に対して処理を行わない場合は”break”文を用いる.

2.5.2　script の補足説明: split 関数

split は非常に便利な関数である. 図 9 では, ”,”で区切られた CSV データを”,”を目印にして分割し, 分割されたデータがリスト tmp に代入される. 従って”tmp[0]”に各行の 1 番目の要素, ”tmp[1]”に各行の 2 番目の要素が代入される.

```
tmp = line.split(",")
```

図 9　split 関数の利用例

同様の処理を C 言語で記述するとそれなりの行数の code を書く必要があるが, **python では数行の文で簡潔に記述**できる. **様々な便利な関数, 命令が標準で利用可能な点は python の大きなメリット**である.

2.5.3 練習課題

io1.py(および入力ファイル input.csv) を作成し, ファイルの入力方法を確認せよ.

2.6 ファイル出力

本節ではファイルへ出力する方法を説明する. リスト 6 を **io2.py** というファイルに保存する. io2.py を実行すると出力ファイル output.txt が新らたに作成される. これを cat コマンドで表示すると図 10 のように表示される.

```
$ ./io2.py
$ cat output.txt
1 行: tmp[0] =      1
1 行: tmp[1] =    0.1
1 行: tmp[2] =      5
1 行: tmp[3] =    1st
2 行: tmp[0] =      2
2 行: tmp[1] =    0.8
2 行: tmp[2] =      8
2 行: tmp[3] =    2nd
3 行: tmp[0] =      3
3 行: tmp[1] =    0.8
3 行: tmp[2] =      7
3 行: tmp[3] =    3rd
4 行: tmp[0] =      4
4 行: tmp[1] =    0.2
4 行: tmp[2] =      2
4 行: tmp[3] =    4th
```

図 10　io2.py の実行と出力ファイル output.txt の表示

リスト 6: 例題 4-2

```
#!/usr/bin/python3
# -*- coding: shift_jis -*- # Windows の場合

input  = "input.csv"  ## 入力ファイル名
output = "output.txt" ## 出力ファイル名
FDI = open( input,'r') ## 読み込みモードでファイル open
FDO = open(output,'w') ## 書き込みモードでファイル open
```

```
for line in FDI:
    ## 各行の改行を除去
    line = line.rstrip('\n')
    line = line.rstrip('\r')
    ## 各行をコンマで分割し, リスト tmp へ代入
    tmp = line.split(",") ## 文字列のリスト
    i = int(tmp[0]) ## 文字列を整数へ変換
    for j in range(4): ## j=0 から 3
        FDO.write("%d 行目: tmp[%d] = %6s\n" % (i, j, tmp[j]))
FDI.close() ## ファイル FDI を閉じる
FDO.close() ## ファイル FDO を閉じる
```

2.6.1 script の補足説明: ファイルへの出力

Script リスト 6 において, **入力と出力のファイル open 方法の違い**に注意すること.

また既存のファイルに script の出力結果を追加したい場合には, 図 11 のように, 追加書き込みモードでファイルを open する.

```
FDA = open(output,'a') ## 追加書き込みモードでファイル open
```

図 11　出力ファイルの open: 既存ファイルへの追記

2.6.2 練習課題

io2.py を作成し, ファイルの入出力方法を確認せよ.

2.7 演習課題 2

input.csv を読み込んで, 図 12 のように, 奇数行の場合は行数および 2 列目 +3 列目を浮動小数型で, 偶数行の場合は行数および 4 列目を文字型で, 画面に表示する script を作成せよ.

```
1 行目: 2 列目 +3 列目:    5.10000
2 行目:        4 列目:        2nd
3 行目: 2 列目 +3 列目:    7.80000
4 行目:        4 列目:        4th
```

図 12　演習課題 2 の出力

2.8 演習課題 3

まず data1.csv を保存し, data1.csv の 1,2 行目のコメント行を Editor などで削除しなさい.

このコメント行を削除したデータを読み込み, その 3,8,10 行目のみ, 各行の 1,2 列目を表示する script を作成せよ. (Script 例リスト 7 を参考にして, コメント行を削除せずに, その行を読み飛ばす処理を行っても良い.)

さらに出力行を図 13 のようなリストで指定し, 指定された行の 1,2 列目を表示する script を作成せよ. ただしリスト loutput の要素数は 3 とは限らない. また len を用いるとリストの要素数を取得することができる.

```
loutput = [3, 8, 10] ## 出力行の指定
```

図 13 演習課題 3 のリストの例

2.9 正規表現

本節では正規表現の基礎的な使い方を説明する.

入力ファイル input2.csv 図 14 のようにデータの先頭にデータに関するコメントがある場合など, 特定の行に対して, 処理を飛ばすようなことが要求されることは良くある.

```
## Comment for data
1, 0.1, 5, 1st
2, 0.8, 8, 2nd
3, 0.8, 7, 3rd
4, 0.2, 2, 4th
```

図 14 io1.py の入力データ input2.csv

この例の場合, 行の先頭の"#"でコメント行を判断し, その行は通常行に対して行なう処理を前述の"continue"でスキップする. 正規表現を用いれば, 行頭に"#"がある行を簡単に検出することができる.

script 例リスト 7 を **regexp.py** というファイルに保存し, 実行すると 1 行目に関しては処理が行なわれていないことがわかる.

リスト 7: 例題 6: 正規表現

```
#!/usr/bin/python3
# -*- coding: shift_jis -*- # Windows の場合
import re ## 正規表現用モジュール re の import
```

```
input = "input2.csv"  ## 入力ファイル名
FD = open(input,'r') ## 読み込みモードでファイル open
for line in FD:
    if ( re.search(r'^#', line ) ): ## 各行の行頭が # の場合
        continue  ## 次の行の処理へ
    ## 各行の改行を除去
    line = line.rstrip('\n')
    line = line.rstrip('\r')
    ## 各行をコンマで分割し, リスト tmp へ代入
    tmp = line.split(",") ## 文字列のリスト
    i = int(tmp[0]) ## 文字列を整数へ変換
    for j in range(4): ## j=0 から 3
        print("%d 行目: tmp[%d] = %6s" % (i, j, tmp[j]))
FD.close()  ## ファイル FD を閉じる
```

この regexp.py では図 15 の部分で正規表現が用いられている．re.search() は re.search(パターン, 検索対象文字列) のように用いられ, r'^#' が"正規表現のパターン"を表す．"^"が"先頭"を指すので, r'^#' は"先頭が"#"であること"を表す.

また正規表現を用いるには, "import re"のように正規表現用モジュール re を import する必要がある.

```
    if ( re.search(r'^#', line ) ): ## 各行の行頭が # の場合
        continue  ## 次の行の処理へ
```

図 15　regexp.py の正規表現

2.10　CSV データからの特定データの抽出

本節では, CSV データから (指定する条件を満たす) 特定のデータを抽出する方法を学ぶ.

script 例リスト 8 は, 1 行に 9 つデータが存在する csv ファイルから, 温度 (1 列目), 抵抗率 (2 列目), 電気抵抗 (7 列目), 電気抵抗の測定誤差 (8 列目) を抜き出し, 表示するプログラムである. ただし, 誤差が大きいデータに関しては, 先頭に"##"を付けて表示している.

リスト 8: 例題 7: CSV データからの特定データの抽出

```
#!/usr/bin/python3
# -*- coding: shift_jis -*- # Windows の場合
import re ## 正規表現用モジュール re の import

input  = "data1.csv"  ## 入力ファイル名
err_max = 2e-6 ## 誤差の最大値

## 100 行 9 列の 2 次元配列の初期化
```

```
ni = 100 ## データ行数の上限
nj =   9 ## 各行のデータ数（列数）
## array[列][行]: 行、列の順番ではないことに注意
## array[列][行]: この方が何かと便利
array = [[0 for i in range(ni)] for j in range(nj)]

### CSV データの 2 次元配列への格納
FD  = open( input,'r') ## 読み込みモードでファイル open
nd = 0; ## データ行数のカウンタ
for line in FD:
    if ( re.search(r'^#', line ) ): ## 行頭が # の場合
        continue  ## 次の行の処理へ
    ## 各行の改行を除去
    line = line.rstrip('\n')
    line = line.rstrip('\r')
    ## 各行をコンマで分割し，リスト tmp へ代入
    tmp = line.split(",")
    for j in range(nj): ## float 型に変換し、代入
        array[j][nd] = float(tmp[j]) ## array[列][行]
    nd += 1
FD.close()  ## ファイルを閉じる

### 出力
for i in range(nd):
    T   = array[0][i] ## 絶対温度
    rho = array[1][i] ## 抵抗率
    R   = array[6][i] ## 電気抵抗
    err = array[7][i] ## 電気抵抗の測定誤差
    ## 誤差が大きいデータはコメント
    if ( err > err_max ):
        print("###", end='')
    print( "%7.2f %11.4e %11.4e %11.4e" % (T, rho, R, err) )
```

2.11 演習課題 4

前の例題で用いた csv ファイル data1.csv およびそれと同じフォーマットの csv ファイル data2.csv, csv ファイル data3.csv のそれぞれに対して, 温度 (1 列目), 抵抗率 (2 列目), 電気抵抗 (7 列目), 電気抵抗の測定誤差 (8 列目) を抜き出し, ファイルに出力する script を作成せよ. ただし, 抵抗値が負のデータに関しては, 先頭に"##"を付けて出力すること. また出力ファイル名は適切に設定すること.

3 研究課題

以下の流れで学生データ excel ファイルから調査票 PDF を生成する python コードを考える. このコードの一部 (Excel 生成, PDF 生成, 地図生成のいずれか. 当然, 全部でもよい. 複数名 (最大 3 名) での共同作成大歓迎) を実現する python コードを作成しなさい.

1. 学生データ excel ファイルを読み込む
2. 調査票テンプレート excel を読み込む
3. 以上を元にデータファイルに含まれる全学生の調査票 excel ファイルを生成
4. ただし保護者の住所を元に地図画像を生成し, 調査票 excel ファイル (またはテンプレート) の F6 に貼り付ける.
   （a）地図画像は Google map API を使って生成すると良い.
   （b）Google map API は無料で利用可能だが, API キーの取得にはクレジットカードが必要でした. (2020 年 5 月 14 日現在)
   （c）地図画像は住所を中心とした正方形画像とし, 住居の周辺部も適切に含むようにズームすること.
   （d）地図画像の画像のサイズは 300x300 画素とする
5. さらに生成した (またはテンプレート)excel ファイルから PDF を生成する.

課題を実現するための手法は Web 上の情報を検索し, それを参考にすること. また参考にした Web ページの URL は参考文献として明示すること.