システムアーキテクト演習
6週目

吉田 則裕

Pythonプログラミング

Pythonの概要

複数言語使える人は
優秀な人であることが多い

- 特長
  - 読みやすく，覚えやすい文法
  - 高い生産性
  - 高い移植性
  - 豊富なライブラリ（機械学習，自然言語処理，統計処理）
  - 他のプログラミング言語によるプログラムとの連携

- 制限
  - バイトコードを逐次解釈するインタプリタ型言語であるため，C言語に比べると少し遅い

基本データ型

- 数値を表すデータ型
  - 整数型
    - 0b ではじまる数字は2 進数
    - 0o ではじまる数字は8 進数
    - 0x ではじまる数字は16 進数
  - 浮動小数点型

C言語と表記がちょっと違う

- 論理型データ
  - True
  - False

C言語と異なり予め用意されている

- 文字列型
  - " "で囲う

算術演算子

演算子	説明
x + y	xとyの加算
x – y	xからyの減算
x * y	xとyの乗算
x / y	xのyによる除算
x // y	xのyによる整数除算（小数点以下切り捨て）
x % y	xのyによる除算の剰余
-x	xの正負の反転
x ** y	xのyによる，べき乗計算

算術演算子の中で加算や乗算は文字列型データにも適用できる.

- ’Hello ’+’World’ の結果は’Hello World’
- ’Hello ’*3 の結果は’Hello Hello Hello ’ となる.

演習1

1. 以下のプログラムを記入

```
print (2+5)
print (2-5)
print (2*3)
print (2/4)
print (2**3)
print (12%5)
```

2. 右上にある▶️ボタンを押し実行
3. 下側に実行結果が表示される

演習2

- 2 進数0110と8進数34 と16進数1Aを加算してみよう

演習2の解答例

```
print (0b110+0o34+0x1A)
```

60が出力される

代入文

```
a = 10
print (a)

a = 'Hello'
print (a)

a, b, c, d = 10, 3.0, 'Hello ', 'World'
print (a + b)
print (c + d)

a, b, c, d = 551, 6735, '551', '6735'
print (a + b)
print (c + d)
print(len(c))
```

文字列の代入が代入文でできる

複数変数への代入を1度にできる

len関数で文字数をカウント

```
10
Hello
13.0
Hello World
7286
5516735
3
```

入出力文

- キーボードからの入力はinput() 関数
- ディスプレイへの出力はprint() 関数

```
text = input()
print(text)

data = input()
print(data)

print(data*2)

print(int(data))
print(int(data)*2)

name = input('Enter your name: ')

print(name)
```

```
abc
abc
120
120
120120
120
240
Enter your name: Ritsumei
Ritsumei
```

input関数の出力は文字列なので，
文字列が2回繰り返される

int関数は文字列を数値に変換

input関数に文字列を付加することができる.

演習3

- a=input(’Enter your name: ’) を実行し，自分の名前をローマ字で入力しよう.
- さらにprint(’My name is: ’, a) を実行してみよう.

```
a = input('Enter your name:')
print('My name is:', a)
```

論理演算子

- 論理和を表すor
- 論理積を表すand
- 論理否定を表すnot

```
print (False or 0)                0
print (0 or 0.0)                  0.0
print (3 or 0)                    3
print (0 or 4)                    4
print (0 or False or 0.0)         0.0
print (0 or 1 or 0.0)             1
print (0 and 0.0)                 0
print (2 and 0)                   0
print (True and 3)                3
print (not 0)                     True
print (not 3)                     False
print (not '')                    True
print (not 'Hello')               False
```

論理演算の対象はブール変数だけでなく，数値や文字列なども対象としている．
数値の0，空文字列，空リストは偽扱い

比較演算子

演算子	比較条件
x<y	xはyより小さい
x<=y	xはyより小さいか等しい
x>y	xはyより大きい
x>=y	xはyより大きいか等しい
x==y	xとyは等しい
x!=y	xとyは等しくない
x in y	xはyに含まれる
x not in y	xはyに含まれない

```
print (5 >= 3)                                  True
print (3 < 5)                                   True
print (3 == 4)                                  False
print (3 == 3)                                  True
print (3 != 4)                                  True
print ('Hello' in 'Hello World')                True
print ('hello' in 'Hello World')                False
```

if文

条件分岐は以下のように記述する.

```
if 条件式a:
    条件式a が真の場合の処理
    ...
    ...
elif 条件式b:
    条件式b が真の場合の処理
    ...
    ...
elif 条件式c:
    条件式c が真の場合の処理
    ...
    ...
else:
    上記すべての条件式が偽の場合の処理
    ...
    ...
```

- 条件式はブール型のTrue だけでなく, 0 以外の数値や, 空でない文字列なども真となる.
- if 文の先頭から条件式を評価していき, ある条件式が真になった場合, その条件式の直後にある段下げ(インデント) された部分を実行し, 終了する.
- 段下げが浅くなるとブロックが終わったことを意味する.
- 段下げはtabキーを押す.

while文

```
while 条件式:
    ...
    ...
    ...
else:
    ...
    ...
```

- while 文は指定した条件式が真の間，段下げした部分を実行する.
- else 節がある場合は，while ループが終了した後にelse 節を実行する.
- break 文があると，ループを抜け出し，else 節は実行しない.

最大公約数を求めるプログラム

プログラム中で日本語が使えるように文字コードを指定

```
# coding: SHIFT_JIS
# このプログラムは2 つの整数を入力して，その最大公約数を求める
# written by xxxx, 2022 年11 月7 日
print('2 つの整数の最大公約数を求めます')
print() # 1 行空ける
while True: # 無限ループ
  m = int(input('1 番目の数を入力してください: '))
    n = int(input('2 番目の数を入力してください: '))
    # input 文で読んだデータは文字列型なのでint 文で整数に変換
  if m > n:
        m, n = n, m # m がn より大きい場合はm とn を入れ替える
  while n != 0:
        m, n = n, m % n # ユークリッドの互除法
  print('最大公約数は', m, ' です．')
    print()
    ans = input('Do you want to exit?(y/n) ')
    if ans=='y': break # ループを抜ける
```

コメントを書いておくと，後日自分自身や他人が読んだときにわかりやすい

演習４

- 最大公約数のプログラムを参考にして，２ つの整数を入力とし，最小公倍数を求めるプログラムを作ってみよう.

演習4の解答例

```
# coding: SHIFT_JIS
# このプログラムは2 つの整数を入力して，その最小公倍数を求める
# written by Norihiro Yoshida, 2022 年11 月7 日
print('2 つの整数の最小公倍数を求めます')
print() # 1 行空ける
while True: # 無限ループ
  a = int(input('1 番目の数を入力してください: '))
    b = int(input('2 番目の数を入力してください: '))
    m, n = a, b
    # input 文で読んだデータは文字列型なのでint 文で整数に変換
    if m > n:
        m, n = n, m # m がn より大きい場合はm とn を入れ替える
    while n != 0:
        m, n = n, m % n # ユークリッドの互除法
    print('最小公倍数は', a*b//m, ' です． ')
    print()
    ans = input('Do you want to exit?(y/n) ')
    if ans=='y': break # ループを抜ける
```

あとで入力し数値を使えるように
別の変数にコピーしておく

関数

- 関数はdef 文で始まる.
- def 文には関数名とカンマで区切った引数をカッコで囲んで指定し，最後に:を記述する.
- 関数の本体は，改行し段下げした部分に記述する.

```
def bigger(m,n): # function named bigger displays larger value.
    if m > n:
        print(m)
    else:
        print(n)
```

演習5

- 2 つの整数を引数として入力し，大きい方の数から小さい方の数を引いた値を表示する関数subを作ってみよう.

演習5の解答例

```
def sub(m,n):
    if m > n:
        print(m-n)
    else:
        print(n-m)

sub(5,4)
sub(4,5)
```

モジュール

- モジュールはPython の関数をまとめたファイルで，拡張子が’py’ のテキストファイルである.
- モジュールを他のプログラムから利用する場合はimport 文でモジュールを読み込むことによって利用できる.

モジュール（続き）

- 例えば，ファイル名addition.py に以下の関数を定義したとする．

```
def add(a, b):
    return(a+b)
```

- この関数を利用するには，以下のようにすれば良い．

```
import addition
addition.add(23,45)
```

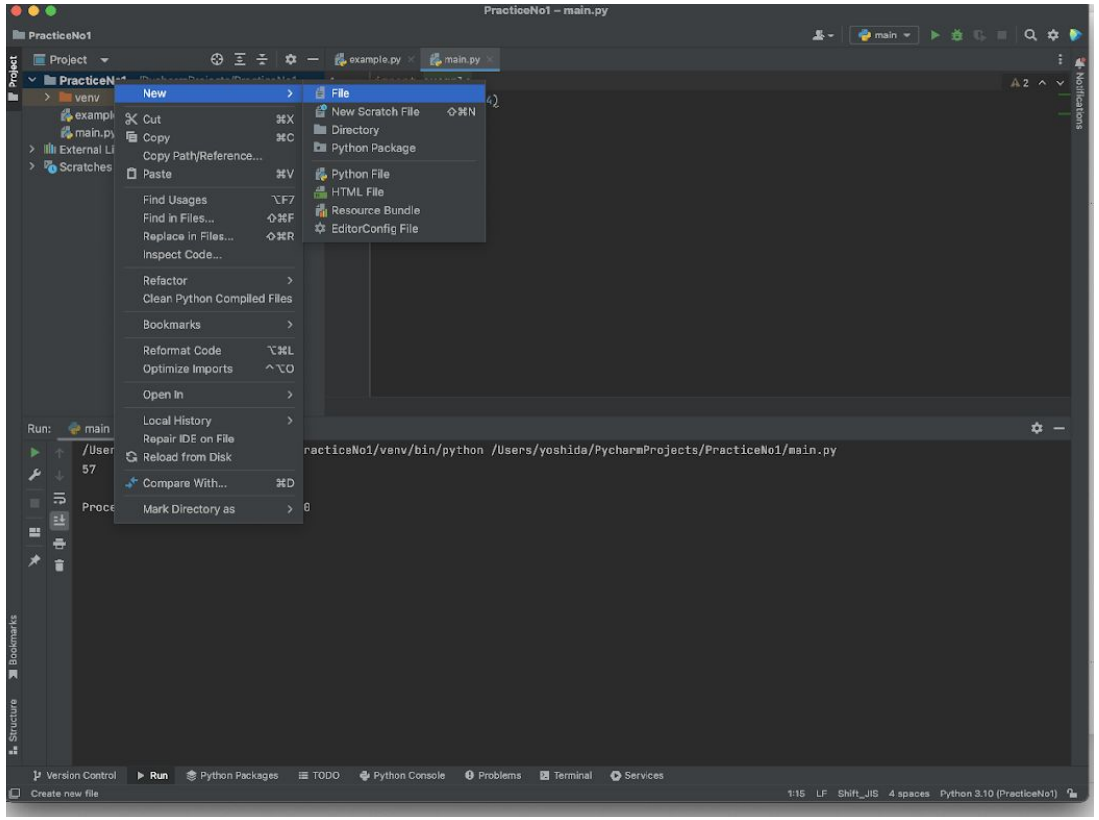
- import 文で指定するモジュール名はファイル名から拡張子 ’py’ を除いた名前とする．
- import文によって指定したファイル中に定義した関数が利用できるようになる．
- 関数を利用するには ’addition.add’のようにファイル名と関数名の間に ’.’ を指定する．

- モジュール名を逐一記述するのが手間であれば，以下のように記述することもできる．

```
from addition import add
add(23,45)
```

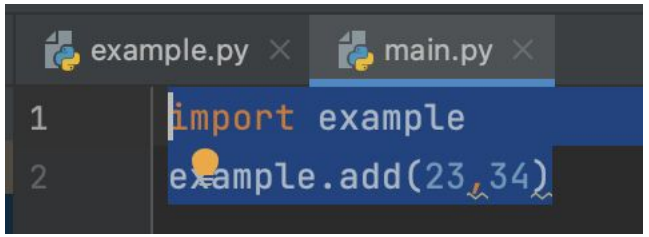
PyCharmにおけるファイルの作成

1. 左側のプロジェクトビューにおいてプロジェクト名を選択
2. 右クリックメニューからNewを選択
3. Fileを選択
4. ファイル名として，example.pyを入力



ファイルの切り替え

エディターの上のタブで，入力するファイルを切り替えることができる



example.pyに以下の内容を記述

```
def add(x,y):
    print(x+y)

def sub(m,n):
    if m > n:
        print(m-n)
    else:
        print(n-m)
```

main.pyに以下の内容を記述

```
import example
example.add(23,34)
```

実行すると57と表示される

main.pyは，以下のように記述しても良い．

```
from example import add, sub
add(23,34)
sub(23,34)
```

exampleモジュールからadd関数とsub関数を読み込む

演習6，演習7

- 2つの整数を引数とし，加算する関数add と大きい方の数から小さい方の数を減算する関数subをexample.py というファイルに作成し，importした上で，それぞれの関数を実行してみよう.
  - 前のスライドまでで説明した内容

- 2つの整数をキーボードから入力し，加算結果と大きい方の数から小さい方の数を減算した結果を表示するプログラムを演習6で作成した2つの関数を利用して作成しよう.

演習7の解答例

```
# coding: SHIFT_JIS
from example import add, sub

a = int(input('1 番目の数を入力してください: '))
b = int(input('2 番目の数を入力してください: '))
add(a,b)
sub(a,b)
```

シーケンス

- C言語の配列のようなデータの並んだ構造をシーケンスと呼ぶ.
  - シーケンスに格納されるデータ型には，リスト型，タプル型，文字列型，バイト型などがある.
- シーケンスは，要素の番号を表す添字(index) によって，その要素番号に格納されたデータ要素をアクセスできる.

- リスト型シーケンス
  - [要素, 要素, ...]
  - 変更可能
- タプル型シーケンス
  - (要素, 要素, ...)
  - 追加・変更ができない
- これまであつかった文字列もシーケンスの一種
  - 変更できない.

[] と書くと，要素の無い空リストを定義できる

()と書くと，要素の無い空タプルを定義できる

添字によるアクセス

- シーケンスに格納されたデータ要素は，seq[index] の形式によって，index で示された要素番号のデータ要素を参照できる.
- 要素番号は，先頭から0,1,2,... と割り当てられる.
- 要素番号に負の整数値を指定すると，シーケンスの末尾からの指定となる.
  - -1 が最終要素番号
  - -2 が最後から2 番目の要素番号

演習8

- 以下のプログラムを参考にし，[0,1,2,3,4,5,6,7,8,9] から構成されるリストL を定義し，先頭要素と最終要素を参照してみよう.
- 文字列’Creation Core’ から構成される文字列S を定義し，その2 番目の要素を参照してみよう.

```
L = [10,9,8,7,6]
print(L[2])
print(L[-2])

S = 'Ritsumeikan'
print(S[0])
print(S[3])
print(S[-2])
print(S)

print(L)
```

```
8
7
R
s
a
Ritsumeikan
[10, 9, 8, 7, 6]
```

演習8の解答例

- L = [0,1,2,3,4,5,6,7,8,9]
  print(L[0])　　0
  print(L[-1])　　9

  S = 'Creation Core'
  print(S[1])　　r