ソフトウェア工学（2018 年度）　第 02 回 例題
(後日配布用)

飯島 正

2018 年 10 月 03 日 (第 02 回の実施日)
2018 年 10 月 26 日 (スタイルファイルのテスト中)

1　例題 02_001x: triangle_area

Listing 1　例題 001x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=============================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=============================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------------
# 例題[02]-(001x)
#               Ex_02_001x_triangle_area.py
#-----------------------------------------------------------------------------
# Ex(Example) 02-001x: 【関数定義】
#               三角形の面積を返す関数を定義する.
#               [底辺]と[高さ]から[三角形の面積]を返す関数.
#-----------------------------------------------------------------------------
#       ※ エラー処理は省略（例外処理は後日取り扱う）.
#       ※ (読み込んだデータ文字列がfloat型へ変換できるかどうかは，判定していない.)
#-----------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に，main関数等の調整，整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=============================================================================
#======【関数定義】　三角形の面積を返す関数
#=============================================================================
#   ※ 引数:   bottom (float) 底辺
#   ※ 引数:   height (float) 高さ
#   ※ 返戻値:  (float) 三角形の面積
#-----------------------------------------------------------------------------
def area_of_triangle( bottom, height ):
```

```
32      # ----- 計算と結果の返戻 -----
33      return( ( bottom * height ) / 2 )
34  # ================================================================================
35  # ======【関数定義】 オープニングメッセージの表示 ==========
36  # --------------------------------------------------------------------------------
37  def opening_message():
38      # ----- オープニングメッセージの表示 ----
39      print( "三角形の面積を求めます: " )
40      print()
41  # ================================================================================
42  # ======【関数定義】 パラメータの入力 ==========
43  #   ※ 返戻値: (tuple(float,float)) (底辺, 高さ)
44  # --------------------------------------------------------------------------------
45  def get_parameters():
46      # ----- パラメータの入力 -----
47      teihen = float( input( "  底辺の長さ[cm]を入力してください>>> " ) )
48      takasa = float( input( "  高さの長さ[cm]を入力してください>>> " ) )
49      print()
50      # ----- 入力したデータの返戻 -----
51      return( ( teihen, takasa ) )
52  # ================================================================================
53  # ======【関数定義】 実行と結果の表示 ==========
54  #   ※ 引数:  teihen (float) 底辺
55  #   ※ 引数:  takasa (float) 高さ
56  # --------------------------------------------------------------------------------
57  def execution( teihen, takasa ):
58      # ----- 計算 -----
59      result = area_of_triangle( teihen, takasa )
60      # ----- 結果の表示 -----
61      print( "(1) 三角形の面積は ", result, " [cm^2]", sep="" )
62      print( "(2) 三角形の面積は",  result, "[cm^2]" )
63      print( "(3) 三角形の面積",    result, sep="は ", end=" [cm^2]\n" )
64      print( "(4) 三角形の面積は " + str( result ) + " [cm^2]" )
65      print()
66  # ================================================================================
67  # ======【関数定義】 メイン関数 ==========================================================
68  # --------------------------------------------------------------------------------
69  def main():
70      # ----- オープニングメッセージ ----------
71      opening_message()
72      # ----- パラメータの入力 -----
73      (teihen, takasa) = get_parameters()
74      # ----- 実行と結果の表示 ----
75      execution( teihen, takasa )
76  # ================================================================================
77  # ======【メイン・プログラム】 ==========
78  if __name__ == "__main__":
```

```
79          main()
80  # ============================================================================
```

2　例題 02_002x: mersenne_number

Listing 2　例題 002x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=============================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=============================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------------
#   例題          [02]-(002x) Ex_02_002x_mersenne_number.py
#-----------------------------------------------------------------------------
#   Ex(Example) [02]-(002x) 【関数定義】
#                 メルセンヌ数を求める関数を定義する.
#-----------------------------------------------------------------------------
#       ※ n番目のメルセンヌ数は,2の冪よりも 1 小さい自然数,
#       ※ すなわち 2^n - 1.
#-----------------------------------------------------------------------------
#       ※ エラー処理は省略 (例外処理は後日取り扱う).
#       ※ (読み込んだデータ文字列がint型へ変換できるかどうかは,判定していない.)
#       ※ (int型へ変換できるかどうかは,ある程度はstr.isdigit()でも判定できる.)
#       ※ (str.isdigit()では,負の数や,前後に空白が入っているときはFalseとなる.)
#-----------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に,main関数等の調整,整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=============================================================================
#=====【関数定義】 n番目のメルセンヌ数
  ==========================================================
#                ※n番目のメルセンヌ数は,2のn乗-1
#   ※ 引数:     n (int) 何番目か
#   ※ 返戻値: (int) メルセンヌ数
#-----------------------------------------------------------------------------
def mersenne_number( n ):
    # ----- 計算と結果の返戻 -----
    return( 2 ** n - 1 )
#=============================================================================
#=====【関数定義】 オープニングメッセージの表示
  ==========================================
#-----------------------------------------------------------------------------
def opening_message():
    # ----- オープニングメッセージの表示 ----
    print( "メルセンヌ数を求めます: " )
    print()
#=============================================================================
```

```
45 # ======【関数定義】　パラメータの入力
     ==========================================
46 #    ※ 返戻値:  (int) 何番目か
47 # ------------------------------------------------------------------------------
48 def get_parameters():
49     # ----- パラメータの入力 -----
50     n = int( input( "正の整数を入力してください>>> " ) )
51     print()
52     # ----- 入力したデータの返戻 -----
53     return( n )
54 # ==============================================================================
55 # ======【関数定義】　実行と結果の表示
     ==========================================
56 #    ※ 引数:   n (int) 何番目か
57 # ------------------------------------------------------------------------------
58 def execution( n ):
59     # ----- 計算と結果の表示 -----
60     result = mersenne_number( n )
61     print( "    ", n, "番目の，メルセンヌ数は ", result )
62     print()
63     # ----- 計算と結果の表示 -----
64     print( "1から", n, "までの，メルセンヌ数列: ", sep="" )
65     for i in range( 1, n+1 ):
66         print( "    ", n, "番目のメルセンヌ数は ", mersenne_number( i ) )
67     print()
68 # ==============================================================================
69 # ======【関数定義】　メイン関数 ==============================================
70 # ------------------------------------------------------------------------------
71 def main():
72     # ----- オープニングメッセージ --------------
73     opening_message()
74     # ----- パラメータの入力 -----
75     n = get_parameters()
76     # ----- 実行と結果の表示 ----
77     execution( n )
78 # ==============================================================================
79 # ======【メイン・プログラム】
     ==========================================
80 if __name__ == "__main__":
81     main()
82 # ==============================================================================
```

3　例題 02_003x: triangle_number

Listing 3　例題 003x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=============================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=============================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------------
#   例題          [02]-(003x) Ex_02_003x_triangle_number.py
#-----------------------------------------------------------------------------
#   Ex(Example) [02]-(003x) 【関数定義】
#                   n番目の三角数を求める関数を定義する.
#-----------------------------------------------------------------------------
#       ※ n番目の三角数は，1からnまでの総和と等しい.
#-----------------------------------------------------------------------------
#       ※ エラー処理は省略（例外処理は後日取り扱う）.
#       ※ (読み込んだデータ文字列がint型へ変換できるかどうかは，判定していない.)
#       ※ (int型へ変換できるかどうかは，ある程度はstr.isdigit()でも判定できる.)
#       ※ (str.isdigit()では，負の数や，前後に空白が入っているときはFalseとなる.)
#-----------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に，main関数等の調整，整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=============================================================================
#======== 【関数定義】 n番目の三角数 ========================================
#   ※ 引数:   n (int) 何番目か
#   ※ 返戻値: (int) n番目の三角数
#-----------------------------------------------------------------------------
def triangle_number( n ):
    return( int( n * (n + 1) / 2 ) )
#=============================================================================
#======== 【関数定義】 1からnまでの総和 ======================================
#   ※ 引数:   n (int) 何番目までか
#   ※ 返戻値: (int) 1からnまでの総和
#-----------------------------------------------------------------------------
def sum_of_one_to_n( n ):
    # ----- 変数の準備 -----
    sum = 0
    # ----- 計算のループ -----
    for i in range( 1, n+1 ):
        sum += i
    # ----- 結果の返戻 -----
```

```
46      return( sum )
47  # ================================================================================
48  # ======【関数定義】　オープニングメッセージの表示 ==========
49  # --------------------------------------------------------------------------------
50  def opening_message():
51      # ----- オープニングメッセージの表示 ----
52      print( "n番目の三角数を求めます（1～nの総和と比較します）: " )
53      print()
54  # ================================================================================
55  # ======【関数定義】　パラメータの入力 ==========
56  #    ※ 返戻値:  (int) 何番目か
57  # --------------------------------------------------------------------------------
58  def get_parameters():
59      # ----- パラメータの入力 -----
60      n = int( input( "正の整数を入力してください>>> " ) )
61      print()
62      # ----- 入力したデータの返戻 -----
63      return( n )
64  # ================================================================================
65  # ======【関数定義】　実行と結果の表示 ==========
66  #    ※ 引数:    n (int) 何番目か
67  # --------------------------------------------------------------------------------
68  def execution( n ):
69      # ----- 計算と結果の表示 -----
70      print( "    ",    n, "番目の三角数は ", triangle_number( n ) )
71      print()
72      # ----- 計算と結果の表示 -----
73      print( "    1 ～", n, "までの総和は ",    sum_of_one_to_n( n ) )
74      print()
75  # ================================================================================
76  # ======【関数定義】　メイン関数 ==========================================================
77  # --------------------------------------------------------------------------------
78  def main():
79      # ----- オープニングメッセージ ----------
80      opening_message()
81      # ----- パラメータの入力 -----
82      n = get_parameters()
83      # ----- 実行と結果の表示 ----
84      execution( n )
85  # ================================================================================
86  # ======【メイン・プログラム】 ==========
87  if __name__ == "__main__":
88      main()
89  # ================================================================================
```

4　例題 02_004x: average_of_inputs

Listing 4　例題 004x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=============================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=============================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------------
#   例題          [02]-(004x) Ex_02_004x_average_of_inputs.py
#-----------------------------------------------------------------------------
#   Ex(Example) [02]-(004x) 【関数定義】
#                   "X"が入力されるまで正の整数の平均値を返す関数.
#-----------------------------------------------------------------------------
#       ※ エラー処理は省略（例外処理は後日取り扱う）.
#       ※ (読み込んだデータ文字列がint型へ変換できるかどうかは，判定していない.)
#       ※ (int型へ変換できるかどうかは，ある程度はstr.isdigit()でも判定できる.)
#       ※ (str.isdigit()では，負の数や，前後に空白が入っているときはFalseとなる.)
#-----------------------------------------------------------------------------
#       ※ 後日，データ入力と平均値の計算を分離する(その方が再利用性が高い).
#-----------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に，main関数等の調整，整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=============================================================================
#====== 【関数定義】 "X"が入力されるまで正の整数の平均値を返す関数 ======
#   ※ 返戻値: (float) 入力されたデータの平均値
#-----------------------------------------------------------------------------
def average_of_inputs():
    # ----- 変数の準備 -----
    sum = 0.0
    num = 0
    # ----- 計算のループ -----
    while True:
        # ----- 終了条件を満たすまでデータ入力を受け付ける -----
        data = input( "正の整数を入力してください>>> " )
        if data == "X":
            break
        # ----- 入力データのカウントと，そこまでの総和の計算 -----
        num += 1
        sum += int( data )
    # ----- 平均の計算と結果の返戻 -----
    return( sum / num )
#=============================================================================
```

```
47 # ======【関数定義】 オープニングメッセージの表示 ======
48 # ------------------------------------------------------------------------------
49 def opening_message():
50     # ----- オープニングメッセージの表示 ----
51     print( "\"X\"が入力されるまで正の整数の平均値を返す: " )
52     print()
53 # ==============================================================================
54 # ======【関数定義】 実行と結果の表示 ======
55 # ------------------------------------------------------------------------------
56 def execution():
57     # ----- 平均値の計算 -----
58     average = average_of_inputs()
59     # ----- 結果の表示 ----
60     print( "\"X\"が入力されるまで正の整数の平均値は", average )
61     print()
62 # ==============================================================================
63 # ======【関数定義】 メイン関数 ==========================================================
64 # ------------------------------------------------------------------------------
65 def main():
66     # ----- オープニングメッセージ ----------
67     opening_message()
68     # ----- 結果の表示 ----
69     execution()
70 # ==============================================================================
71 # ======【メイン・プログラム】 ======
72 if __name__ == "__main__":
73     main()
74 # ==============================================================================
```

5　例題 02_005x: date_and_time_builtin

Listing 5　例題 005x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=============================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=============================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------------
#   例題          [02]-(005x) Ex_02_005x_date_and_time_builtin.py
#-----------------------------------------------------------------------------
#   Ex(Example) [02]-(005x) 【組込関数】
#                     [日付]と[時刻]にかかわる組込み関数の使い方.
#-----------------------------------------------------------------------------
#       ※ 日付時刻を扱うためのdatetimeモジュールをつかう
#       ※ 今日の[日付]オブジェクトを返す関数と，現在の[時刻]オブジェクトを返す関数を定義する.
#-----------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に，main関数等の調整，整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=============================================================================
#=========【モジュールのインポート】==========================================
#-----------------------------------------------------------------------------
#----- 日付時刻を扱うためのdatetimeモジュールをインポートする
#---------------------------------
import datetime
#=============================================================================
#=========【関数定義】本日の日付オブジェクトを返す関数=========
#   ※ 返戻値: (datetime.date) 本日の日付オブジェクト
#-----------------------------------------------------------------------------
def today():
    return( datetime.date.today() )
#=============================================================================
#=========【関数定義】現在時刻の日付時刻オブジェクトを返す関数=========
#   ※ 返戻値: (datetime.datetime) 現在時刻の日付時刻オブジェクト
#-----------------------------------------------------------------------------
def now():
    return( datetime.datetime.now() )
#=============================================================================
#=========【関数定義】本日の日付文字列を返す関数=========
#   ※ 返戻値: (str) 本日の日付文字列
#-----------------------------------------------------------------------------
```

```
44 def today_str():
45     date_of_today = today()
46     return( str( date_of_today.year  ) + "年" +
47             str( date_of_today.month ) + "月" +
48             str( date_of_today.day   ) + "日" )
49 # ================================================================================
50 # ======【関数定義】現在時刻の日付時刻文字列を返す関数 ======
51 #   ※ 返戻値: (str) 現在時刻の日付時刻文字列
52 # --------------------------------------------------------------------------------
53 def now_str():
54     current_time = now()
55     return( str( current_time.hour   ) + "時" +
56             str( current_time.minute ) + "分" +
57             str( current_time.second ) + "秒" +
58             str( current_time.microsecond ) + "μ秒" )
59 # ================================================================================
60 # ======【関数定義】 オープニングメッセージの表示 ==========================
61 # --------------------------------------------------------------------------------
62 def opening_message():
63     # ----- オープニングメッセージの表示 ----
64     print( "標準モジュールdatetimeを使って，年月日からその日の曜日を求めます: " )
65     print()
66 # ================================================================================
67 # ======【関数定義】 実行と結果の表示 ======================================
68 # --------------------------------------------------------------------------------
69 def execution():
70     # ----- 実行結果の表示 ----
71     #  ----- 今日の日付の表示 -----
72     print( "今日の日付は", today() )
73     print( "今日の日付は", today_str() )
74     print()
75     # ----- 現在の時刻の表示 -----
76     print( "現在の時刻は", now() )
77     print( "現在の時刻は", now_str() )
78     print()
79 # ================================================================================
80 # ======【関数定義】 メイン関数 ==============================================
81 # --------------------------------------------------------------------------------
82 def main():
83     # ----- オープニングメッセージ ----------
84     opening_message()
85     # ----- 結果の表示 ----
86     execution()
87 # ================================================================================
88 # ======【メイン・プログラム】 ==============================================
89 if __name__ == "__main__":
90     main()
91 # ================================================================================
```

6　例題 02_006x: day_of_week_builtin

Listing 6　例題 006x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=============================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=============================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------------
#   例題          [02]-(006x) Ex_02_006x_day_of_week_builtin.py
#-----------------------------------------------------------------------------
#   Ex(Example) [02]-(006x) 【組込関数】 組込み関数の使い方.
#                         指定した日の[曜日]を文字列で返す関数を定義する.
#-----------------------------------------------------------------------------
#       ※ 日付時刻を扱うためのdatetimeモジュールをつかう
#-----------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に, main関数等の調整, 整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=============================================================================
#======【モジュールのインポート】
  ==========================================================
#-----------------------------------------------------------------------------
#----- 日付時刻を扱うためのdatetimeモジュールをインポートする
  ---------------------------------
import datetime
#=============================================================================
#======【関数定義】本日の日付オブジェクトを返す関数 ======
#   ※ 返戻値: (datetime.date) 本日の日付オブジェクト
#-----------------------------------------------------------------------------
def today():
    return( datetime.date.today() )
#=============================================================================
#======【関数定義】本日の曜日を文字列で返す関数 ======
#   ※ 返戻値: (str) 本日の曜日の文字列
#-----------------------------------------------------------------------------
def day_of_week_today():
    date_of_today = today()
    dow = date_of_today.weekday()
    return( day_of_week_str(dow) )
#=============================================================================
#======【関数定義】指定した日付の曜日を文字列で返す関数 ======
#   ※ 引数: year (int) 年
#   ※ 引数: month (int) 月
```

```
45 #    ※ 引数:  day    (int) 日
46 #    ※ 返戻値:  (str) 指定した日付の曜日の文字列
47 # ------------------------------------------------------------------------
48 def day_of_week( year, month, day ):
49     dow = datetime.date( year, month, day ).weekday()
50     return( day_of_week_str(dow) )
51 # ========================================================================
52 # ======【関数定義】曜日コード(0～6)から対応する文字列を返す関数 ======
53 #    ※ 引数:  曜日コード  (int) 0～6の整数で，順に月曜日から日曜日を意味する
54 #    ※ 返戻値:  (str) 曜日コードに対応する文字列
55 # ------------------------------------------------------------------------
56 def day_of_week_str( dow ):
57     # dowの値は0のとき月曜日，1のとき火曜日，...，6のとき日曜日を意味します．
58     # リスト(配列)や辞書を使えば，もっと簡潔に書けます．  → 後日やりましょう．
59     if dow == 0:
60         return( "月" )
61     elif dow == 1:
62         return( "火" )
63     elif dow == 2:
64         return( "水" )
65     elif dow == 3:
66         return( "木" )
67     elif dow == 4:
68         return( "金" )
69     elif dow == 5:
70         return( "土" )
71     else:
72         return( "日" )
73 # ========================================================================
74 # ======【関数定義】 結果の表示 ==========================================
75 #    ※ 引数:  message (str) 日付を表す文字列
76 #    ※ 引数:  dow_str (str) 曜日を表す文字列
77 # ------------------------------------------------------------------------
78 def print_day_of_week( message, dow_str ):
79     print( message, "の曜日は", dow_str, "曜日です．", sep="" )
80 # ========================================================================
81 # ======【関数定義】 オープニングメッセージの表示
     ==========================================
82 # ------------------------------------------------------------------------
83 def opening_message():
84     # ----- オープニングメッセージの表示 ----
85     print( "標準モジュールdatetimeを使って，年月日からその日の曜日を求めます: " )
86     print()
87 # ========================================================================
88 # ======【関数定義】 実行と結果の表示
     ================================================
89 # ------------------------------------------------------------------------
90 def execution():
91     # ----- 実行結果の表示 ----
92     #  ----- 今日の曜日の表示 -----
93     print( "今日の曜日は", day_of_week_today(), "曜日です．", sep="" )
```

```
94      print()
95      # ----- 指定した日付の曜日の表示 -----
96      print_day_of_week( "2018年10月 3日", day_of_week( 2018, 10,  3 ) )
97      print_day_of_week( "2018年10月10日", day_of_week( 2018, 10, 10 ) )
98      print_day_of_week( "2018年12月31日", day_of_week( 2018, 12, 31 ) )
99      print_day_of_week( "2019年 1月 1日", day_of_week( 2019,  1,  1 ) )
100     print()
101 # ================================================================================
102 # ====== 【関数定義】 メイン関数 ================================================
103 # --------------------------------------------------------------------------------
104 def main():
105     # ----- オープニングメッセージ -------------
106     opening_message()
107     # ----- 結果の表示 ----
108     execution()
109 # ================================================================================
110 # ====== 【メイン・プログラム】
      ================================================
111 if __name__ == "__main__":
112     main()
113 # ================================================================================
```

7　例題 02_007x: interval_between_dates

Listing 7　例題 007x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=============================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=============================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------------
#   例題          [02]-(007x) Ex_02_007x_interval_between_dates.py
#-----------------------------------------------------------------------------
#   Ex(Example) [02]-(007x) 【組込関数】　日数計算に関する組込み関数の使い方.
#                       与えられた二つの日付の間の日数を求める関数を定義する.
#-----------------------------------------------------------------------------
#       ※ 日付時刻を扱うためのdatetimeモジュールをつかう
#-----------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に，main関数等の調整，整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=============================================================================
#=====【モジュールのインポート】
  ===========================================
#-----------------------------------------------------------------------------
#----- 日付時刻を扱うためのdatetimeモジュールをインポートする -----
import datetime
#=============================================================================
#=====【関数定義】与えられた二つの日付の間の日数(経過日数)を求める関数 =====
#   ※ 引数:  date1 (datetime.date) 日付1
#   ※ 引数:  date2 (datetime.date) 日付2
#   ※ 返戻値:  (int) 経過日数
#-----------------------------------------------------------------------------
def interval_between_dates( date1, date2 ):
    #----- 経過日数を計算し返戻 -----
    return( (date2 - date1).days )
#=============================================================================
#=====【関数定義】=====
#   ※ 返戻値:  (int) 次の元日までの日数
#-----------------------------------------------------------------------------
def interval_to_next_ganjitsu():
    #----- 本日の日付オブジェクトを求める -----
    today = datetime.date.today()
    #----- 次の元日の日付オブジェクトを求める -----
    next_year = today.year + 1
    next_ganjitsu = datetime.date( next_year, 1, 1 )
```

```
46      # ----- 経過日数を計算し，返戻 -----
47      return( interval_between_dates( today, next_ganjitsu ) )
48  # ================================================================================
49  # ======【関数定義】 オープニングメッセージの表示 ==========
50  # --------------------------------------------------------------------------------
51  def opening_message():
52      # ----- オープニングメッセージの表示 ----
53      print( "与えられた二つの日付の間の日数を求めます: " )
54      print()
55  # ================================================================================
56  # ======【関数定義】 実行と結果の表示 ==========
57  # --------------------------------------------------------------------------------
58  def execution():
59      # ----- 実行と結果の表示 ----
60      print( "次の元日までの日数は", interval_to_next_ganjitsu(), "日です." )
61      print()
62  # ================================================================================
63  # ======【関数定義】 メイン関数 ==========
64  # --------------------------------------------------------------------------------
65  def main():
66      # ----- オープニングメッセージ ----------
67      opening_message()
68      # ----- 結果の表示 ----
69      execution()
70  # ================================================================================
71  # ======【メイン・プログラム】 ==========
72  if __name__ == "__main__":
73      main()
74  # ================================================================================
```

8 例題 02_008x: random_builtin

Listing 8 例題 008x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#==========================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#==========================================================================
# ソフトウェア工学[02]
#--------------------------------------------------------------------------
#   例題          [02]-(008x) Ex_02_008x_random_builtin.py
#--------------------------------------------------------------------------
#   Ex(Example) [02]-(008x) 【組込関数】 [乱数]を返す組込み関数の使い方.
#                       乱数の種(シード)に関して理解するためのテスト用関数を定義する.
#--------------------------------------------------------------------------
#           ※ 乱数を取り扱うためのrandomモジュールを使う.
#--------------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#--------------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に,main関数等の調整, 整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#==========================================================================
#====== 【モジュールのインポート】
  ==========================================================
#--------------------------------------------------------------------------
#----- 乱数を取り扱うためのrandomモジュールをインポートする -----
import random
#--------------------------------------------------------------------------
#----- 時間/時刻を取り扱うためのtimeモジュールをインポートする -----
import time
#==========================================================================
#====== 【関数定義】 乱数(0～9の整数)のテスト(指定した個数生成する)) ======
#   ※乱数の種(シード)として同じ値を設定すると同じ乱数列が生成される.
#   ※乱数の種(シード)として異なる値を設定するために,時刻を与えることはよくある.
#   ※乱数の種(シード)を省略しても,現在のシステム時刻が使われる.
#   ※time.time()は(UTCで)1970年1月1日0時0分0秒(エポックという)からの秒数
#--------------------------------------------------------------------------
#   ※ 引数: seed_of_random (int/float/str/...) 乱数の種(シード)
#   ※ 引数: num            (int)      生成する乱数の個数
#--------------------------------------------------------------------------
def test_of_random( seed_of_random, num ):
    print( "    ", "----- 乱数のテスト ----" )
    print( "    ", "  ※乱数のシード:", seed_of_random )
    random.seed( seed_of_random )
    for i in range( 0, num ):
        print( "    ", random.randint( 0, 9 ) )
```

```
46      print( "    ", "-------------------" )
47  # ================================================================================
48  # ======【関数定義】 オープニングメッセージの表示 ==========
49  # --------------------------------------------------------------------------------
50  def opening_message():
51      # ----- オープニングメッセージの表示 ----
52      print( "種(シード)を指定して(疑似)乱数を生成します: " )
53      print()
54  # ================================================================================
55  # ======【関数定義】 実行と結果の表示 ==========
56  # --------------------------------------------------------------------------------
57  def execution():
58      # ----- 実行と結果の表示 ----
59      print( "※種(シード)が同じなら，同じ(疑似)乱数系列が生成されます．" )
60      print( "※5個ずつ乱数を生成します．" )
61      test_of_random( 0, 5 )
62      test_of_random( 1, 5 )
63      test_of_random( 0, 5 )
64      print()
65      # ----- 実行と結果の表示 ----
66      print( "※毎回異なる乱数列を得るために，乱数のシード(種)として，" )
67      print( "※プログラム実行時の現在時刻time.time()を使うことも多い．" )
68      test_of_random( time.time(), 5 )
69      time.sleep( 1.0 ) # 1秒経過させる
70      test_of_random( time.time(), 5 )
71      print()
72  # ================================================================================
73  # ======【関数定義】 メイン関数 ==========
74  # --------------------------------------------------------------------------------
75  def main():
76      # ----- オープニングメッセージ ----------
77      opening_message()
78      # ----- 結果の表示 ----
79      execution()
80  # ================================================================================
81  # ======【メイン・プログラム】 ==========
82  if __name__ == "__main__":
83      main()
84  # ================================================================================
```

9　例題 02_009x: greatest_common_divisor_builtin.py

Listing 9　例題 009x

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
#=======================================================================
# * Copyright (c) 2018 IIJIMA, Tadashi
# *       (IIJIMA Laboratory, Dept. of Science and Technology, Keio University).
#=======================================================================
# ソフトウェア工学[02]
#-----------------------------------------------------------------------
#   例題          [02]-(009x) Ex_02_009x_greatest_common_divisor_builtin.py
#-----------------------------------------------------------------------
#   Ex(Example) [02]-(009x) 【組込関数】 組込み関数の使い方.
#                       最大公約数 gcd(greatest common divisor)の使い方.
#-----------------------------------------------------------------------
#       ※ 数学関数を扱うためのmathモジュールを使う.
#       ※ gcd()関数は, Python 3.3から導入された.
#-----------------------------------------------------------------------
#       ※ エラー処理は省略 (例外処理は後日取り扱う).
#       ※ (読み込んだデータ文字列がint型へ変換できるかどうかは, 判定していない.)
#       ※ (int型へ変換できるかどうかは, ある程度はstr.isdigit()でも判定できる.)
#       ※ (str.isdigit()では, 負の数や, 前後に空白が入っているときはFalseとなる.)
#-----------------------------------------------------------------------
#       2018-10-03 飯島 正 (iijima@ae.keio.ac.jp)
#-----------------------------------------------------------------------
# ChangeLog:
#       2018-10-26 iijima 後日掲示用に, main関数等の調整, 整形.
#       2018-10-26 iijima LuaLaTeXでの印刷用にコメントの調整.
#       2018-10-03 iijima 最初の実装.
#=======================================================================
#=====【モジュールのインポート】
  ===================================================
#-----------------------------------------------------------------------
#----- 数学関数を扱うためのmathモジュールをインポートする -----
import math
#=======================================================================
#=====【関数定義】 オープニングメッセージの表示
  ===============================================
#-----------------------------------------------------------------------
def opening_message():
    #----- オープニングメッセージの表示 ----
    print( "2数の最大公約数gcdを求める: " )
    print()
#=======================================================================
#=====【関数定義】 パラメータの入力
  ===============================================
#   ※ 返戻値: ( tuple(int, int) ) 最大公約数を計算する二数のタプル
#-----------------------------------------------------------------------
```

```
44 def get_parameters():
45     # ----- パラメータの入力 -----
46     x = int( input( "正の整数を入力してください>>> " ) )
47     y = int( input( "正の整数を入力してください>>> " ) )
48     print()
49     # ----- 入力したデータの返戻 -----
50     return( ( x, y ) )
51 # ================================================================================
52 # ======= 【関数定義】　実行と結果の表示 ==========================================
53 #    ※ 引数:    x (int) 最大公約数を計算する二数の一つ目
54 #    ※ 引数:    y (int) 最大公約数を計算する二数の二つ目
55 # --------------------------------------------------------------------------------
56 def execution( x, y ):
57     # ----- 計算と結果の表示 -----
58     print( x, "と", y, "の最大公約数は", math.gcd( x, y ), "です", sep="" )
59     print()
60 # ================================================================================
61 # ======= 【関数定義】　メイン関数 ================================================
62 # --------------------------------------------------------------------------------
63 def main():
64     # ----- オープニングメッセージ -------------
65     opening_message()
66     # ----- パラメータの入力 -----
67     ( x, y ) = get_parameters()
68     # ----- 実行と結果の表示 ----
69     execution( x, y )
70 # ================================================================================
71 # ======= 【メイン・プログラム】 ==================================================
72 if __name__ == "__main__":
73     main()
74 # ================================================================================
```