Accelerated Programming 2

Python Tutorial Exercises – Solutions

Here are sample solutions to the tutorial exercises. For some of the exercises (especially coding exercises), alternative correct solutions are possible. If in doubt, consult the lecturer.

Attempt each exercise before consulting the sample solution.

Exercises for Week 1

1A. *(Names)*

The following are valid names in Python:

```
a    a1    A123    and    second
day_time    over_the_moon
```

The following are invalid names:

```
1a    2nd                (start with digits)
income_£    income_$      (contain invalid characters)
```

1B. *(Arithmetic expressions)*

(a) `m-n` yields –12
(b) `m**3` yields 15625
(c) `-m` yields –25
(d) `n//10` yields 3
(e) `n%10` yields 7
(f) `(x+y+z)/2` yields 0,6

1C. *(Assignment-statements)*

	p	q
`p = 7`	7	
`q = p+1`	7	8
`p = p+1`	8	8
`q = 2*q`	8	16

1D. *(Built-in functions)*

(a) `abs(x-y)` yields 0.1
(b) `min(x,y)` yields 0.3
(c) `round(6*y)` yields 2
(d) `round(7*y)` yields 3

1E. *(Defining functions)*

(a)
```
def double (n):
    return 2*n
```

(b)
```
def close (x, y):
    return (abs(x-y) < 0.5)
```

(c)
```
def leap (y):
    return (y%4 == 0)
```

(d)
```
def leap (y):
    return (y%4 == 0) and (y%100 != 0 or y%400 == 0)
```

Exercises for Week 2

2A. *(If-statements)*

(a)
```
if gender:
    print 'male'
else:
    print 'female'
```

(b)
```
if x < y:
    x = y
```

(c)
```
if season == 0:
    print 'Spring'
elif season == 1:
    print 'Summer'
elif season == 2:
    print 'Autumn'
elif season == 3:
    print 'Winter'
else:
    print 'invalid'
```

2B. *(Evaluating conditional and short-circuit expressions)*

(a) `+1 if n > m else -1` yields –1 or –1, respectively

(b) `+1 if n > m else (-1 if n < m else 0)` yields –1 or 0, respectively

(c) `n == 0 or m/n < 3` yields False or True, respectively

2C. *(Conditional and short-circuit expressions)*

(a) `x if x < y else y`

(b) `x <= 0 or log(x, 10) < y`

2D. *(Executing while-statements)*

(a)

		m	n
`m = 2`		2	
`n = 0`		2	0
`n <= 10`	True	2	0
`print n`	prints 0	2	0
`n = n+m`		2	2
`n <= 10`	True	2	2
`print n`	prints 2	2	2
`n = n+m`		2	4
`n <= 10`	True	2	4
`print n`	prints 4	2	4
`n = n+m`		2	6
`n <= 10`	True	2	6
`print n`	prints 6	2	6
`n = n+m`		2	8
`n <= 10`	True	2	8
`print n`	prints 8	2	8
`n = n+m`		2	10
`n <= 10`	True	2	10
`print n`	prints 10	2	10

`n = n+m`		2	12
`n <= 10`	False	2	12

This code prints even integers from 0 up to 10.

(b)

		m	n
`m = 37`		37	
`n = 11`		37	11
`m > n`	True	37	11
`m = m - n`		26	11
`m > n`	True	26	11
`m = m - n`		15	11
`m > n`	True	15	11
`m = m - n`		4	11
`m > n`	False	4	11

This code computes the remainder on dividing `m` by `n`.

2E. *(While-statements)*

(a)
```
den = 1
term = 4/den
pi = term
while term >= 0.0001:
    pi = pi + term if (den-1)%4 == 0 else pi - term
    den += 2
    term = 4/den
```

(b)
```
def power (x, n):
    p = 1
    m = 0
    while m < n:
        p *= x
        m += 1
    return p
```

(c)
```
def replicate (s, m):
    rep = ''
    limit = m - len(s)
    while len(rep) < limit:
        rep += s
    return rep
```

Exercises for Week 3

3A. *(String expressions)*

(a) `s+t` yields 'bana'

(b) `t+s` yields 'naba'

(c) `2*s` yields 'baba'

(d) `s*2` yields 'baba'

(e) `s+(2*t) == u` yields True

(f) `s in t` yields False

(g) `t in u` yields True

(h)
```
'Student %s has %d grade points and a GPA of %f.' \
% (id, c, g/c)
```
yields 'Student 01234567 has 120 grade points and a GPA of 10.5.'

3B. *(String functions)*

(a)
```
def bracket (s):
    return '{' + s + '}'
```

(b)
```
def trim_left (s):
    i = 0
    while i < len(s) and s[i] = ' ':
        i += 1
    return s[i:]
```

(c)
```
def trim_right (s):
    i = len(s)
    while i > 0 and s[i-1] = ' ':
        i -= 1
    return s[:i]
```

3C. *(Tuples)*

(a) `e[0]` yields 7404

(b) `e[2] * 40` yields 26.00

(c) `pay = 1.2 * e[2]` stores 7.80 in `pay`

(d) `e[2] = pay` fails (tuples are immutable)

3D. *(Lists)*

(a) `f[4]` yields 5

(b) `f[-1]` yields 21

(c) `f[4:7]` yields [5, 8, 13]

(d) `f[:2]` yields [1, 1]

(e) `[2,3,5] in f` yields False

(f) `f + [f[-2]+f[-1]]` yields [1, 1, 2, 3, 5, 8, 13, 21, 34]

3E. *(List functions)*

(a)
```
def sum (ns):
    tot = 0
    for n in ns:
        tot += n
    return tot
```

(b)
```
def concat (ss):
    cat = ''
    for s in ss:
        cat += s
    return cat
```

(c)
```
def zip (xs, ys):
    zs = []
    zslen = min(len(xs), len(ys))
    for i in range(0, zslen)
        zs += [(xs[i], ys[i])]
    return zs
```

(d)
```
def tokens(s):
    toks = []
    i = 0
    while i < len(s):
        if s[i] == ' ':
            i += 1
        else:
            start = i
            i += 1
            while i < len(s) and s[i] != ' ':
                i += 1
            tok = s[start:i]
            toks += [tok]
    return toks
```

3F. *(For-statements)*

(a)
```
p = 1
print p
for i in range(0, n):
    p *= 2
    print p
```

(b)
```
old = 1
new = 1
print old
print new
for i in range(2, n):
    (old, new) = (new, old + new)
    print new
```

(c)
```
for pair in list:
    print pair[0], '\t', pair[1]
```

3G. *(List comprehensions)*

(a) `[n + 1 for n in ns]`

(b) `[n < 0 for n in ns]`

(c) `[n for n in ns if n > 0]`

(d) `[(m, n) for m in ms for n in ns]`

Exercises for Week 4

4A. *(Dictionaries)*

(a) `bag['pear']` yields 3

(b) `bag['kiwi']` fails

(c) `bag['plum'] += 1` replaces 'plum':3 by 'plum':4

(d) `bag['kiwi'] = 1` adds 'kiwi':1 to the dictionary

4B. *(Recursive functions)*

(a)
```
def fib (n):
    return 1 if n <= 2 else fib(n-1) + f(n-2)
```

(b)
```
def sum (ns):
    return 0 if ns = [] else ns[0] + sum(ns[1:])
```

The same things can easily be done by non-recursive functions. (E.g., see the solution to exercise 3F(b).)

4C. *(Files)*

(a)
```
lines = 0
chars = 0
line = f1.readline()
while len(lin) > 0:
    lines += 1
    chars += len(line)
    line = f1.readline()
```

(b)
```
line1 = f1.readline()
while len(line1) > 0:
    f0.write(line1)
    line1 = f1.readline()
```

(c)
```
line1 = f1.readline()
while len(line1) > 0:
    line0 = line1[:-1:-1] + '\n'
    f0.write(line0)
    line1 = f1.readline()
```

4D. *(Modules)*

(a)
```
import trig
area = a**2 * trig.sin(theta) * trig.cos(theta)
```

(b)
```
from trig import *
area = a**2 * sin(theta) * cos(theta)
```