Exercise 1. Consider the following functions:

```
def f(x):
    return x + 1

def g(x):
    return x ** 2

def h(x):
    return x % 5
```

a. What does `f(5)` return?

b. What does `g(f(5))` return?

c. What does `h(g(f(5)))` return?

d. What does `f(g(h(17)))` return?

Exercise 2. Consider the following function:

```
def f(x, y, n = 1):
    return x ** n + y ** n
```

a. What does `f(2, 3)` return?

b. What does `f(2, 3, 2)` return?

c. What does `f(n = 3, y = 2, x = 3)` return?

Exercise 3. What does the following code fragment write?

```
def duplicate(s):
    return s + s

s = 'Hello'
s = duplicate(s)
t = 'Bye'
t = duplicate(duplicate(duplicate(t)))
stdio.writeln(s + t)
```

Exercise 4. Consider the following code fragment:

```
a = list(filter(lambda x: x % 7 == 0, range(1, 28)))
```

a. What is the value of `a`?

b. What does `sum(a)` return?

Exercise 5. Consider the following code fragment:

```
import functools

a = list(map(lambda x: x + 2, range(1, 6)))
b = functools.reduce(lambda x, y: x + y, a)
```

a. What is the value of `a`?

b. What is the value of `b`?

Exercise 6. Consider the following program:

× mystery.py

```
import stdio
import sys

def f(a = 1.0, b = 1.0, c = 1.0):
    return lambda x: a * x ** 2 + b * x + c

def main():
    a = float(sys.argv[1])
    b = float(sys.argv[2])
    c = float(sys.argv[3])
    x = float(sys.argv[4])
    stdio.writeln(f(a, b, c)(x))

if __name__ == '__main__':
    main()
```

a. What does the program write in general?

b. What does the program write when run with the command-line arguments $a = 0$, $b = 2$, $c = 5$, and $x = 2$?

c. What is the value of y in the following interactive Python session?

×

```
>>> import mystery
>>> y = mystery.f()(3)
```

Exercise 7. Implement a library called `logic.py` that supports the following API, along with a suitable `_main()` function that tests all the functions in the API:

`any(a)`	returns `True` if *any* of the entries in the boolean list a is `True`, and `False` otherwise
`all(a)`	returns `True` if *all* of the entries in the boolean list a are `True`, and `False` otherwise

Exercise 8. Consider the following recursive function:

```
def mystery(a, b):
    if b == 0:
        return 0
    if a == 0:
        return mystery(b - 1, a)
    return b + mystery(b, a - 1)
```

a. What is the value returned by the call `mystery(10, 0)`?

b. What is the value returned by the call `mystery(0, 10)`?

c. What is the value returned by the call `mystery(3, 7)`?

d. What is the value returned by the call `mystery(10, 3)`?

e. What is the value returned by the call `mystery(200, 300)`?

f. What does the function `mystery()` compute in general about a and b?

Exercise 9. Consider the function $S(n) = 1^2 + 2^2 + 3^2 + \cdots + n^2$, where n is a positive integer.

a. What is the value of $S(5)$?

b. Provide a recursive definition for $S(n)$.

c. Implement a function `S(n)` using recursion, such that it computes and returns $S(n)$

d. Trace the function call `S(5)`.

Solutions

Solution 1.

a. `6`

b. `36`

c. `1`

d. `5`

Solution 2.

a. `5`

b. `13`

c. `35`

Solution 3. `HelloHelloByeByeByeByeByeByeBye`

Solution 4.

a. `[7, 14, 21]`

b. `42`

Solution 5.

a. `[3, 4, 5, 6, 7]`

b. `25`

Solution 6.

a. The program takes four floats a, b, c, and x as command-line arguments and writes the value of the quadratic equation $ax^2 + bx + c$.

b. `9.0`

c. `13.0`

Solution 7.

× logic.py

```
def any(a):
    for v in a:
        if v:
            return True
    return False

def all(a):
    for v in a:
        if not v:
```

```
            return False
    return True

def _main():
    import stdio

    x = [True, True, False, True]
    y = [True, True, True, True]
    stdio.writeln(any(x))
    stdio.writeln(any(y))
    stdio.writeln(all(x))
    stdio.writeln(all(y))

if __name__ == '__main__':
    _main()
```

Solution 8.

a. 0

b. 0

c. 21

d. 30

e. 60000

f. The product ab.

Solution 9.

a. 55

b. $S(n) = \begin{cases} n^2 + S(n-1) & \text{if } n > 1, \text{ and} \\ 1 & \text{if } n = 1. \end{cases}$

c.

```
def S(n):
    if n == 1:
        return 1
    return n * n + S(n - 1)
```

d.

```
S(5)
  S(4)
    S(3)
      S(2)
        S(1)
          return 1
        return 2 * 2 + 1 = 5
      return 3 * 3 + 5 = 14
    return 4 * 4 + 14 = 30
  return 5 * 5 + 30 = 55
```