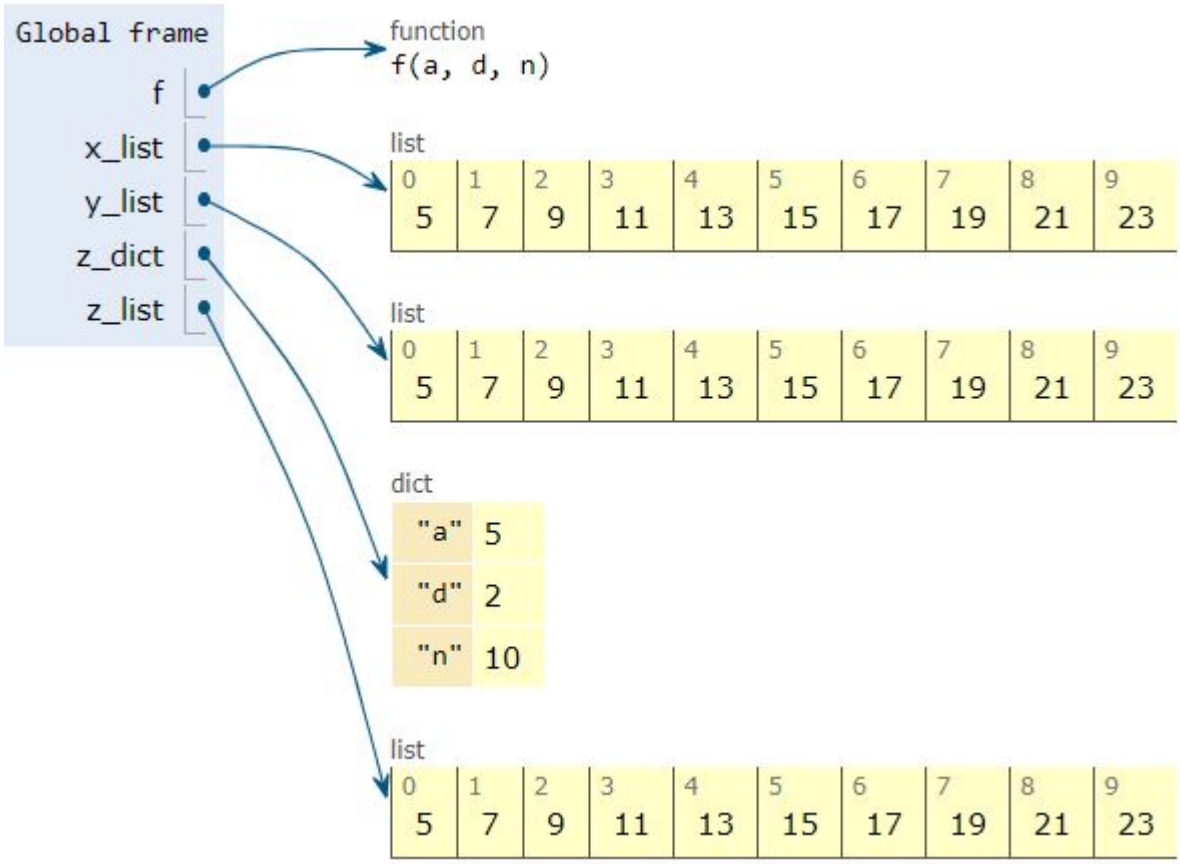
EXERCISES:

PARAMETER PASSING

- show three ways to pass parameters to function $f(a, d, n)$ that returns a list of first n values in arithmetic progression $A(a, d)$

Solution:

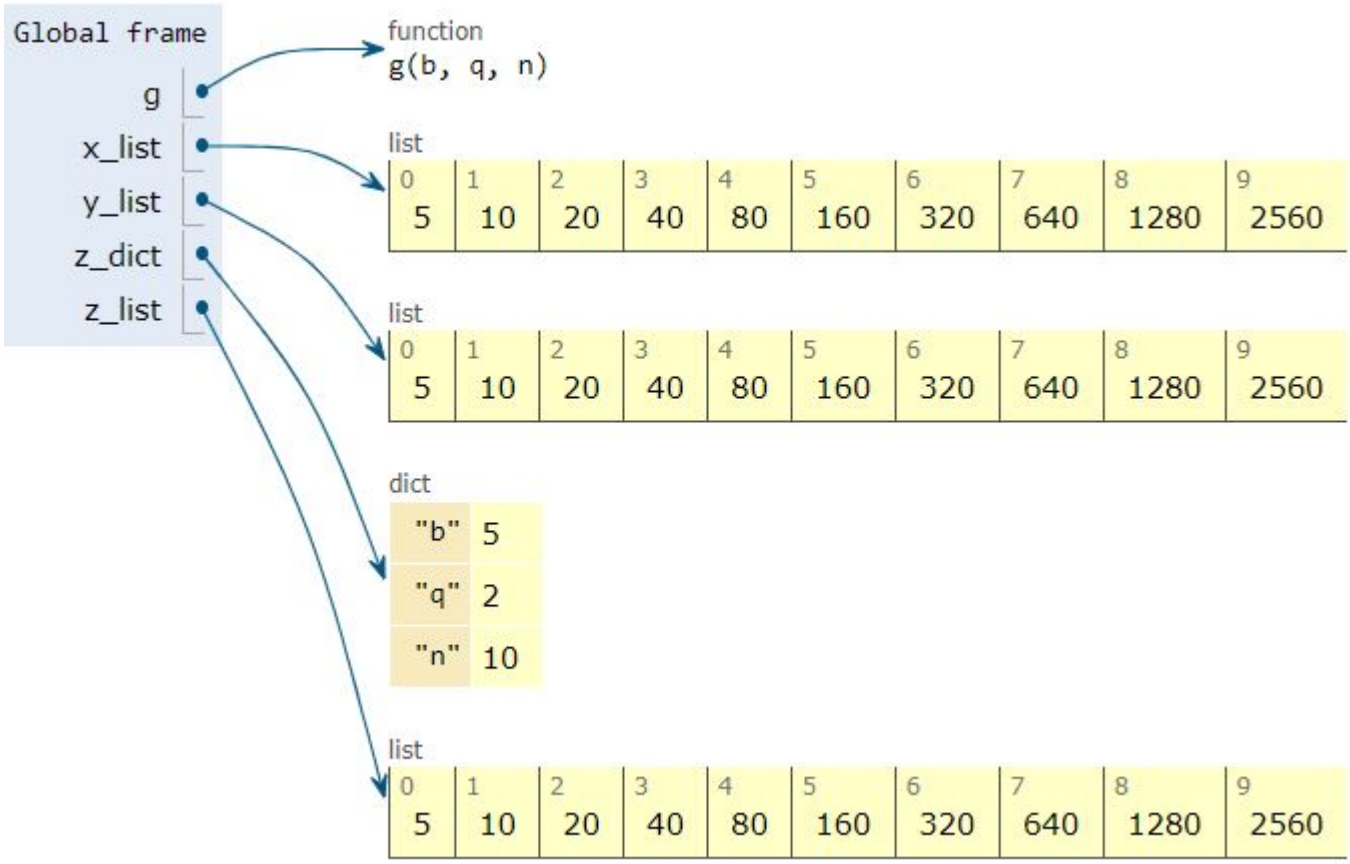
```
def f(a,d,n):
    """ list of first n elements in arith.
        progression with start a and step d """
    last = a + (n-1)*d
    result = list(range(a, last+1, d))
    return result

# by position
x_list = f(5, 2, 10)

# by keyword
y_list = f(n=10, a=5, d=2)

# by dictionary
z_dict = {"a" : 5, "d" : 2, "n" : 10}
z_list = f(**z_dict)
```


Global frame
f
x_list
y_list
z_dict
z_list
function
f(a, d, n)
list
0 1 2 3 4 5 6 7 8 9
5 7 9 11 13 15 17 19 21 23
list
0 1 2 3 4 5 6 7 8 9
5 7 9 11 13 15 17 19 21 23
dict
"a" 5
"d" 2
"n" 10
list
0 1 2 3 4 5 6 7 8 9
5 7 9 11 13 15 17 19 21 23

- show three ways to pass parameters to function $g(b, q, n)$ that returns a list of first n values in geometric progression $G(b, q)$

Solution:

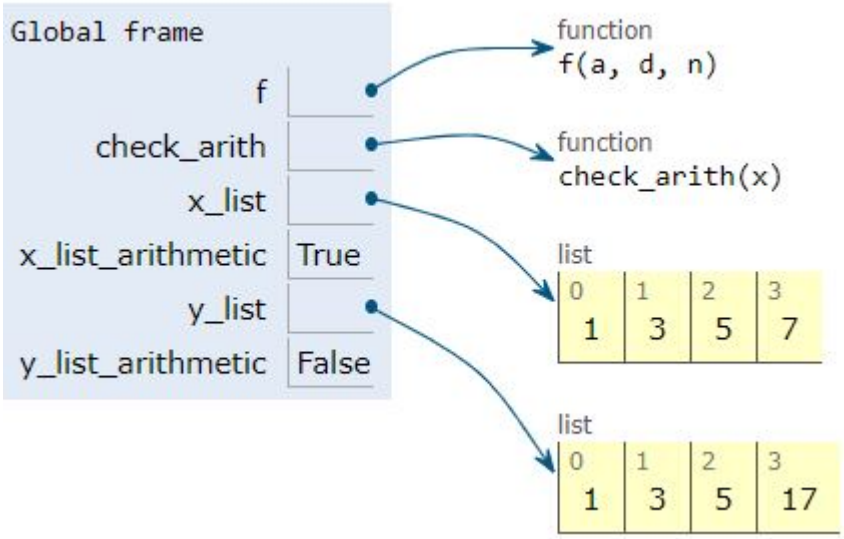
```
def g(b,q,n):
    """ list of first n elements in geom.
        progression with start b, factor d """
    result = [b*q**(i-1) for i in range(1, n+1)]
    return result

# by position
x_list = g(5, 2, 10)

# by keyword
y_list = g(n=10, b=5, q=2)

# by dictionary
z_dict = {"b": 5, "q": 2, "n": 10}
z_list = g(**z_dict)
```


Global frame
g
x_list
y_list
z_dict
z_list
function
g(b, q, n)
list
0 1 2 3 4 5 6 7 8 9
5 10 20 40 80 160 320 640 1280 2560
list
0 1 2 3 4 5 6 7 8 9
5 10 20 40 80 160 320 640 1280 2560
dict
"b" 5
"q" 2
"n" 10
list
0 1 2 3 4 5 6 7 8 9
5 10 20 40 80 160 320 640 1280 2560

- write function *check_arith*() that takes a list of values and determines if this list is an arithmetic progression.

Solution:

```
def f(a,d,n):
    """ list of first n elements in arith.
    progression with start a and setp d """
    last = a + (n-1)*d
    result = list(range(a, last+1, d))
    return result

def check_arith(x):
    """ check if x is an arith. progression """
    n = len(x)
    if n < 2:
        return True
    else:
        a, d = x[0], x[1] - x[0]
        y = f(a, d, n)
        return (x == y)

x_list = [1,3,5,7]
x_list_arithmetic = check_arith(x_list)

y_list = [1,3,5,17]
y_list_arithmetic = check_arith(y_list)
```

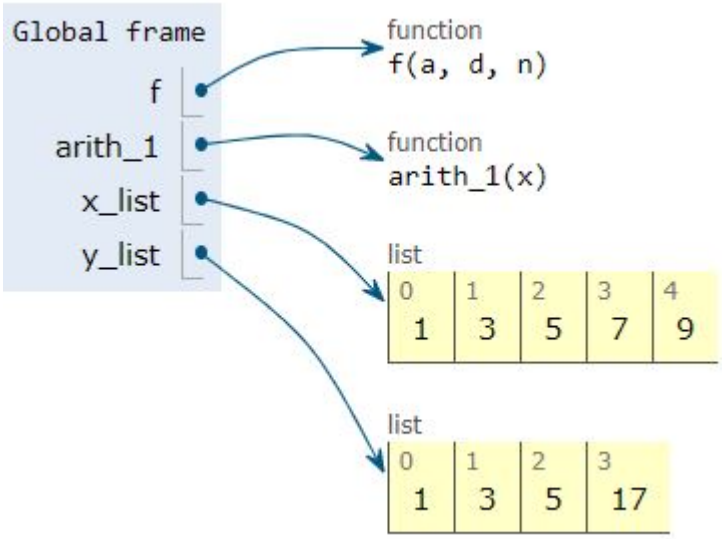

Global frame
f
check_arith
x_list
x_list_arithmetic
True
y_list
y_list_arithmetic
False
function
f(a, d, n)
function
check_arith(x)
list
0
1
3
1
2
5
3
7
list
0
1
3
1
2
5
3
17

- write a function *arith_1*() that takes *x_list* of values. If it is an arithmetic progression, it adds next value to *x_list*.

```
x_list_1 = [1, 3, 5, 7]
# input is OK, add 9
x_list_1 = [1, 3, 5, 7, 9]

x_list_2 = [1, 3, 5, 17]
# input is not OK
x_list_2 = [1, 3, 5, 17]
```

Solution:

```
def f(a,d,n):
    last = a + (n-1)*d
    return list(range(a, last+1, d))

def arith_1(x):
    n = len(x)
    if n < 2:
        return x
    else:
        a, d = x[0], x[1] - x[0]
        y = f(a, d, n)
        if (x == y):
            x.append(y[-1] + d)
            return x
        else:
            return x

x_list = [1, 3, 5, 7]
x_list = arith_1(x_list)
y_list = [1, 3, 5, 17]
y_list = arith_1(y_list)
```

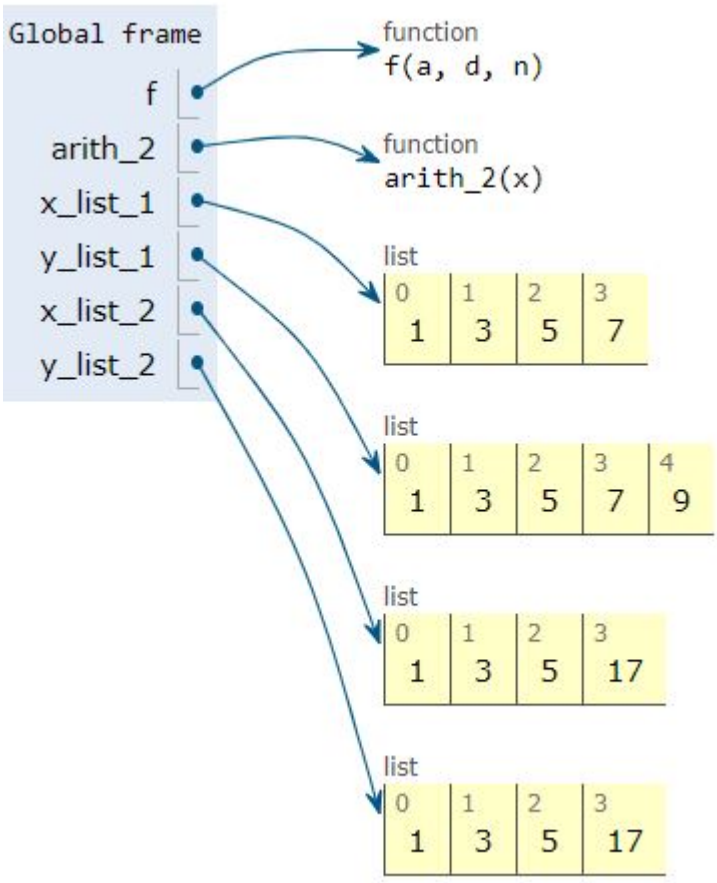

Global frame
f
arith_1
x_list
y_list
function
f(a, d, n)
function
arith_1(x)
list
0 1 2 3 4
1 3 5 7 9
list
0 1 2 3
1 3 5 17

- write a function *arith_2()* that takes *x_list* of values. If it is an arithmetic progression, it returns a *y_list* from *x_list* and the next value.

```
x_list_1 = [1, 3, 5, 7]
# input is OK, add 9
x_list_1 = [1, 3, 5, 7, 9]

x_list_2 = [1, 3, 5, 17]
# input is not OK
x_list_2 = [1, 3, 5, 17]
```

Solution:

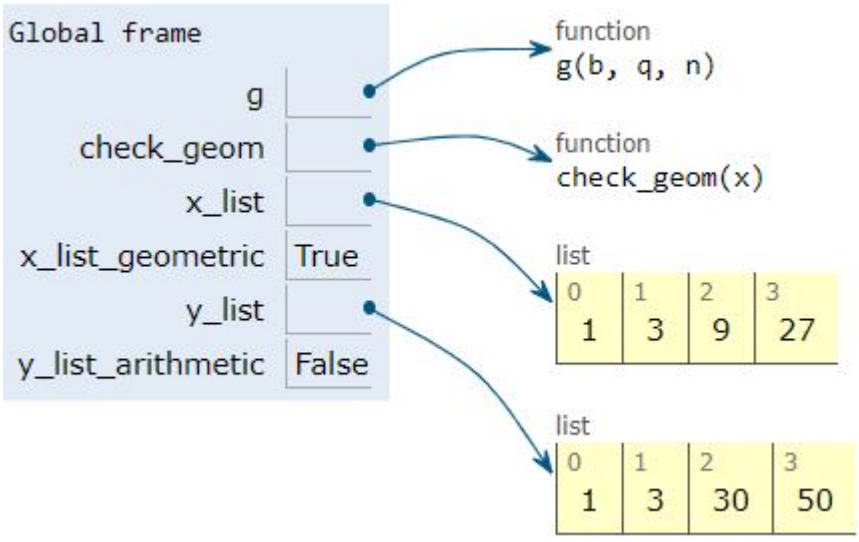
```
def f(a,d,n):
    last = a + (n-1)*d
    return list(range(a, last+1, d))

def arith_2(x):
    n = len(x)
    if n < 2:
        return x.copy()
    else:
        a, d = x[0], x[1] - x[0]
        y = f(a, d, n)
        if (x == y):
            next_element = y[-1] + d
            return x + [next_element]
        else:
            return x.copy()


x_list_1 = [1, 3, 5, 7]
y_list_1 = arith_2(x_list_1)
x_list_2 = [1, 3, 5, 17]
y_list_2 = arith_2(x_list_2)
```


Global frame
f
arith_2
x_list_1
y_list_1
x_list_2
y_list_2
function
f(a, d, n)
function
arith_2(x)
list
0 1 2 3
1 3 5 7
list
0 1 2 3 4
1 3 5 7 9
list
0 1 2 3
1 3 5 17
list
0 1 2 3
1 3 5 17

- write function *check_geom*() that takes a list of values and determines if this list is a geometric progression.

Solution:

```
def g(b, q, n):
    """ list of first n elements in geom.
        progression with start b, factor d """
    result = [b*q**(i-1) for i in range(1, n+1)]
    return result

def check_geom(x):
    """ check if x is an geom. progression """
    n = len(x)
    if n < 2:
        return True
    else:
        b, q  = x[0], x[1]/x[0]
        y = g(b, q, n)
        return (x == y)

x_list = [1, 3, 9, 27]
x_list_geometric = check_geom(x_list)

y_list = [1, 3, 30, 50]
y_list_arithmetic = check_geom(y_list)
```

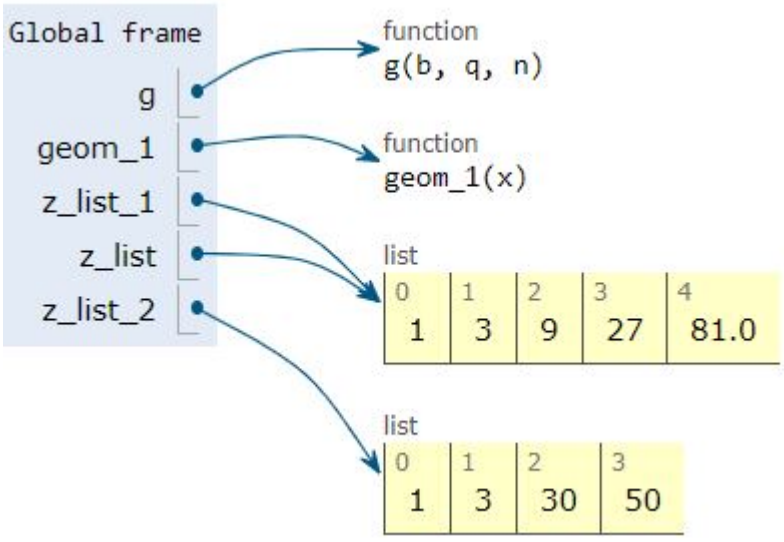

Global frame
g
check_geom
x_list
x_list_geometric
True
y_list
y_list_arithmetic
False
function
g(b, q, n)
function
check_geom(x)
list
0
1
1
3
2
9
3
27
list
0
1
1
3
2
30
3
50

- write a function *geom_1*() that takes *z_list* of values. If it is a geometric progression, it adds next value to *z_list*.

```
z_list = [1, 3, 9, 27]
# input is OK, add 81
z_list = [1, 3, 9, 27, 81]

z_list = [1, 3, 30, 50]
# input is not OK
z_list = [1, 3, 30, 50]
```

Solution:

```
def g(b, q, n):
    result = [ b*q**(i-1) for i in range(1, n+1)]
    return result

def geom_1(x):
    n = len(x)
    if n < 2:
        return x
    else:
        b, q  = x[0], x[1]/x[0] # x[0] is not 0
        y = g(b, q, n)
        if (x == y):
            x.append(y[-1] * q)
            return x
        else:
            return x

z_list_1 = [1, 3, 9, 27]
z_list = geom_1(z_list_1)

z_list_2 = [1, 3, 30, 50]
z_list_2 = geom_1(z_list_2)
```

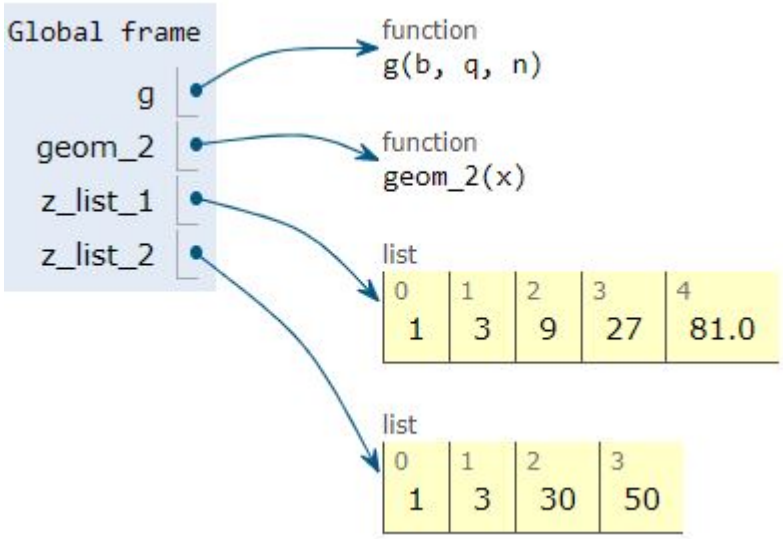

Global frame
g
geom_1
z_list_1
z_list
z_list_2
function
g(b, q, n)
function
geom_1(x)
list
0 1
1 3
2 9
3 27
4 81.0
list
0 1
1 3
2 30
3 50

- write a function *geom_2()* that takes *z_list* of values. If it is a geometric progression, it returns a *w_list* from *z_list* and the next value.

```
z_list = [1, 3, 9, 27]
# input is OK, add 81
w_list = [1, 3, 9, 27, 81]

z_list = [1, 3, 30, 50]
# input is not OK
w_list = [1, 3, 30, 50]
```

Solution:

```
def g(b, q, n):
    result = [b*q**(i-1) for i in range(1, n+1)]
    return result

def geom_2(x):
    n = len(x)
    if n < 2:
        return x
    else:
        b, q  = x[0], x[1]/x[0] # x[0] not 0
        y = g(b, q, n)
        if (x == y):
            next_element = y[-1] * q
            return x + [next_element]
            return x
        else:
            return x

z_list_1 = [1, 3, 9, 27]
z_list_1 = geom_2(z_list_1)
z_list_2 = [1, 3, 30, 50]
z_list_2 = geom_2(z_list_2)
```


Global frame
g
geom_2
z_list_1
z_list_2
function
g(b, q, n)
function
geom_2(x)
list
0 1
1 3
2 9
3 27
4 81.0
list
0 1
1 3
2 30
3 50