FUNCTIONS: FUNCTIONAL PROGRAMMING

Overview:

- learn lambda functions and their use in functional programming

lambda Functions

- use *def* for named functions

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

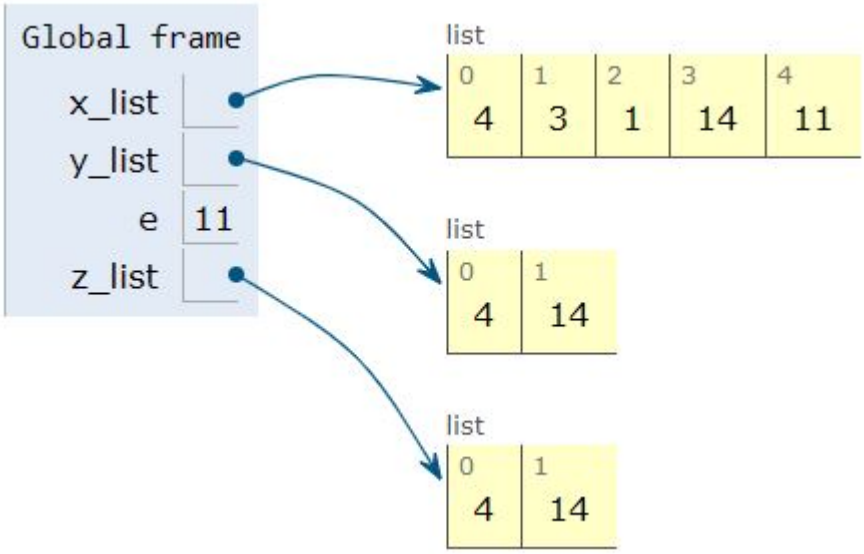
- use for anonymous functions:

```
lambda x: x + 1
```

- single expression only
- can be used with *filter*(), *map*() and *reduce*()

filter() Function

```
# extract even elements from a list
x_list = [4, 3, 1, 14, 11]
y_list = list(filter(lambda x:(x%2==0), x_list))
z_list = [e for e in x_list if e%2 == 0]
```



- items that evaluate to *True*
- syntax: *filter*(function, seq)

Exercise(s):

- consider a list of first n terms in arithmetic progression $A(a, d)$:

  $x = [a, a+d, a+2d, \ldots, a+(n-1)d]$

- compute a sub-list of x with elements divisible by 3

Exercise(s):

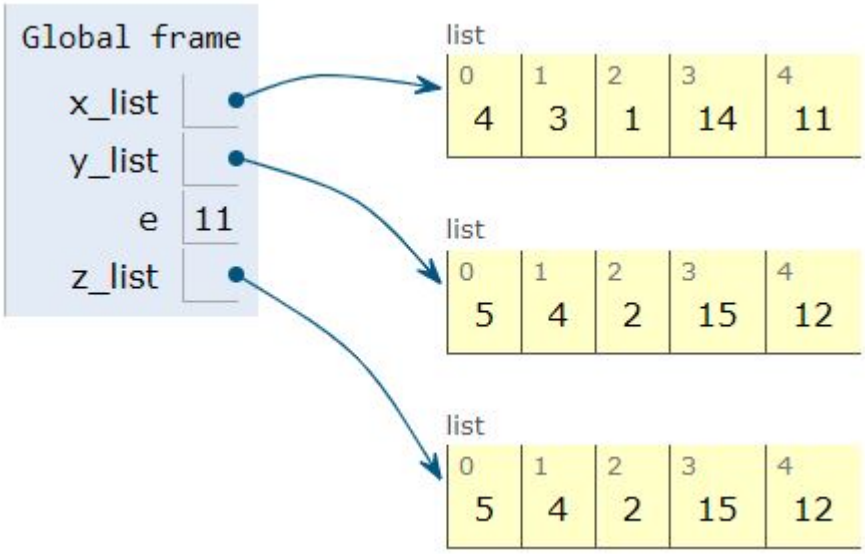
- consider a list of first n terms in geometric progression $G(b, q)$:

$$x = [b, bq, bq^2, \ldots, bq^{n-1}]$$

- compute a sub-list of x with elements divisible by 16

map() Function

```
# increment each element
x_list = [4, 3, 1, 14, 11]
y_list = list(map(lambda x: x + 1, x_list))
z_list = [e + 1 for e in x_list]
```



- apply to each element
- syntax: *map*(function, seq)

Exercise(s):

- consider a list of first n terms in arithmetic progression $A(a, d)$:

  $x = [a, a+d, a+2d, \ldots, a+(n-1)d]$

- double each element in x

Exercise(s):

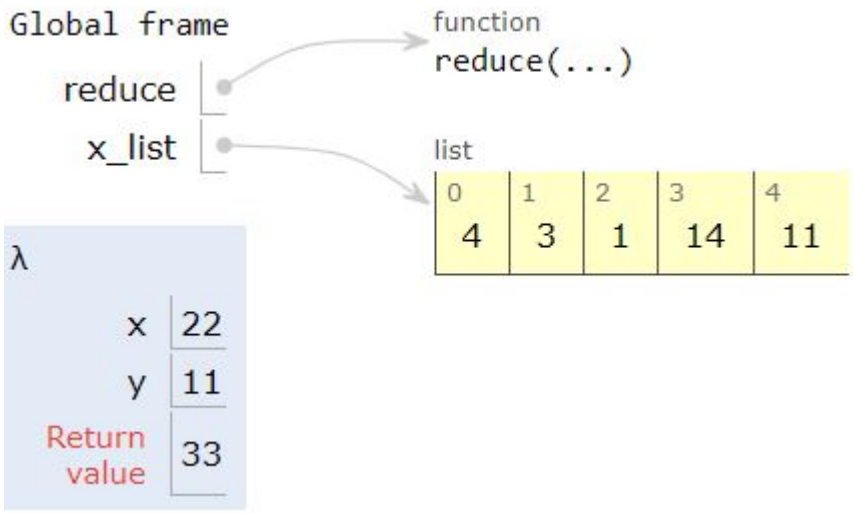
- consider a list of first n terms in geometric progression $G(b, q)$:

$$x = [b, bq, bq^2, \ldots, bq^{n-1}]$$

- increment each element in x by 5

reduce() Function

```
from functools import reduce
x_list = [4, 3, 1, 14, 11]
y = reduce(lambda x, y: x + y, x_list)
```



- iterated binary function
- no longer built-in function
- syntax: *reduce*(function, seq)