Lecture 15

Topics: *Chapter 6. Defining Functions*

- 6.5 Functions That Return Values (continues)
- 6.6 Functions That Modify Parameters
- 6.7 Functions and Program Structure

6.5 Functions That Return Values

Functions may return more than one value.

Example:
the following program has a function that takes two parameters and returns their sum and their product.

```
def main():
   x = int(input('input a decimal number:'))
   y = int(input('input a decimal number:'))
   s,p = sum_prod(x,y)
   print("Their sum is {0}, their product is
{1}.".format(x,y,sum,prod))

def sum_prod(x,y):
   return (x+y), (x*y)

main()
```

6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
→ («original» values cannot be changed inside a function)

6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```

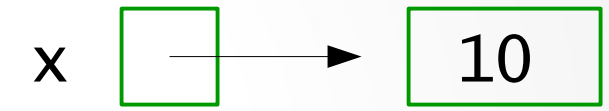
see program prog1.py

6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```

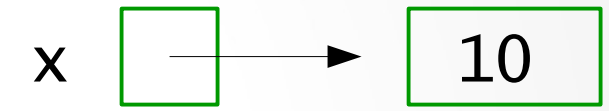


6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```

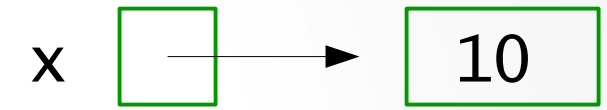


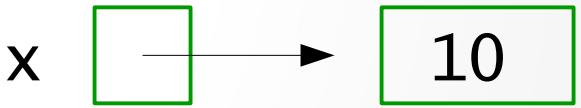
6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```



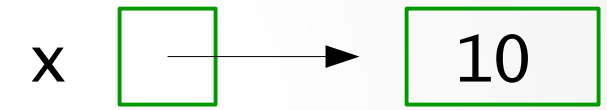


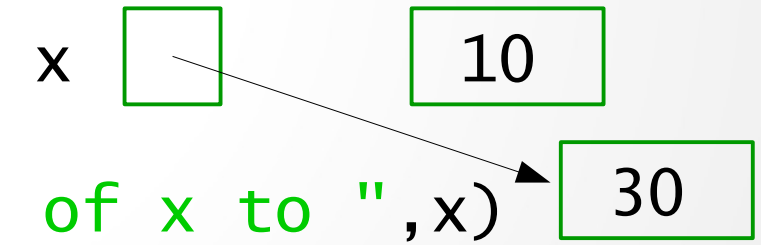
6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```



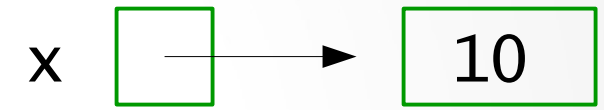


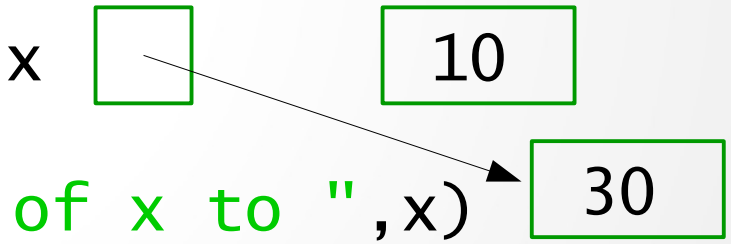
6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```





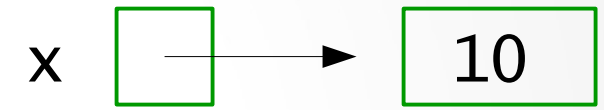

```
changing the value of x to 30
```

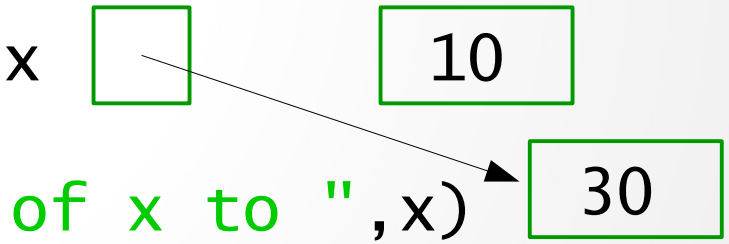
6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```





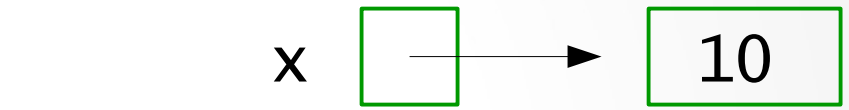

```
changing the value of x to 30
```

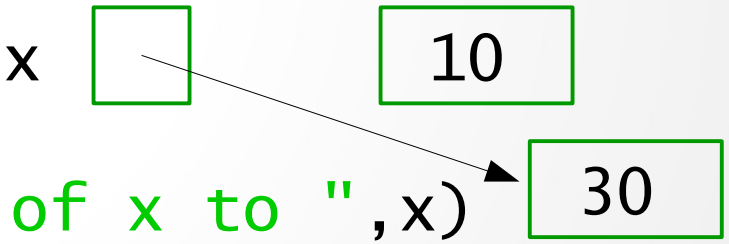
6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

```
def main():
    x = 10
    modify(x)
    print("The value of x is",x)

def modify(x):
    x = x+20
    print("changing the value of x to ",x)
main()
```






```
changing the value of x to 30
The value of x is 10
```

6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»
- («original» values cannot be changed inside a function)

Other languages have a mechanism that allows to change parameter's values, that is called «call by reference». Python doesn't have it.

we will have to return modified value

Warning: Functions may modify variables that are mutable objects (lists, graphics objects)
Recall that mutable means that the value of an item in a list can be modified with an assignment statement.

6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»

we will have to return modified value

```
def main():
      x = 10
      x = modify(x)
      print("The value of x is",x)

def modify(x):
      x = x+20
      print("changing the value of x to ",x)
      return x
main()
```

see program prog2.py

6.6 Functions That Modify Parameters

Can functions modify the parameters?

«Python passes all parameters by value»

we will have to return modified value

```
def main():
      x = 10
      x = modify(x)
      print("The value of x is",x)

def modify(x):
      x = x+20
      print("changing the value of x to ",x)
      return x
main()
```

```
changing the value of x to 30
The value of x is 30
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)




def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

6.6 Functions That Modify Parameters

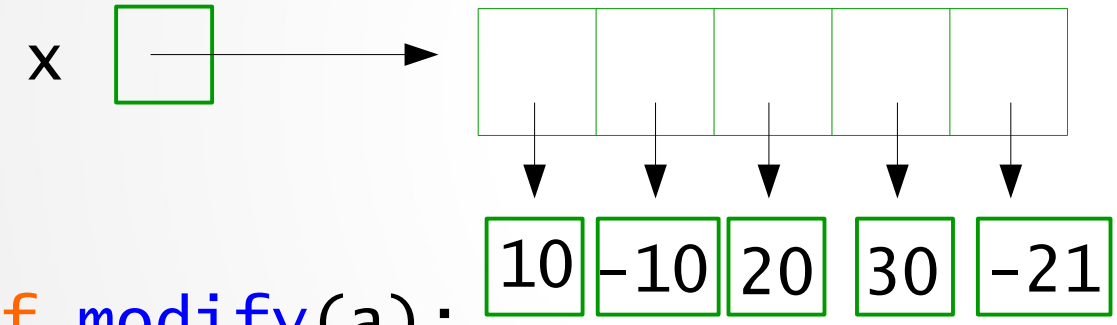
Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)




def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```

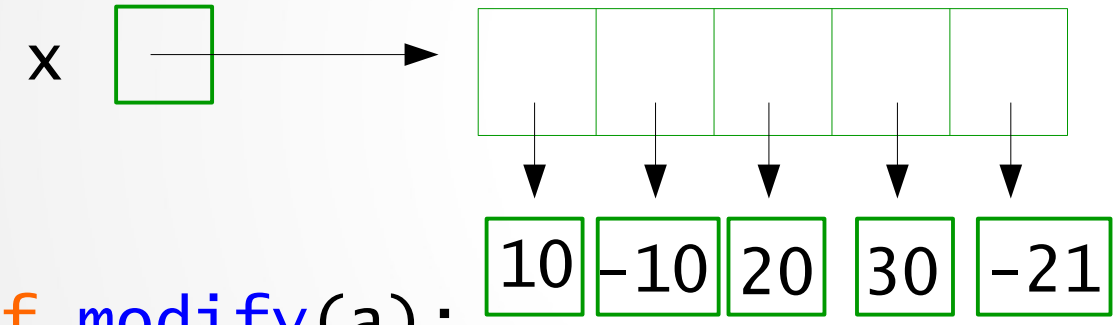



```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



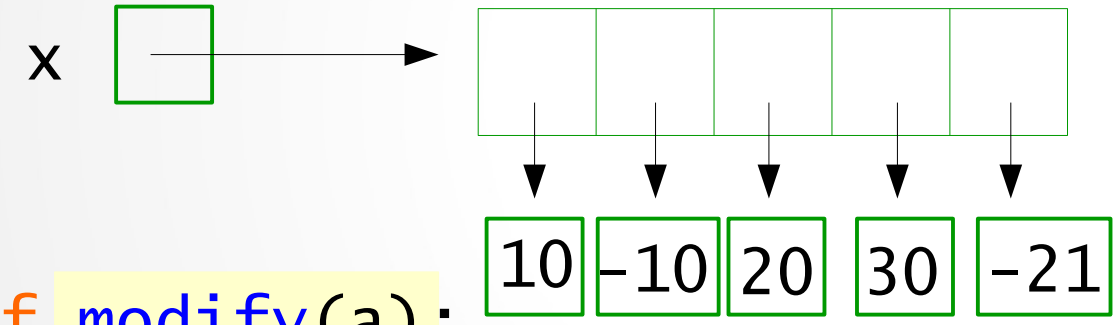

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



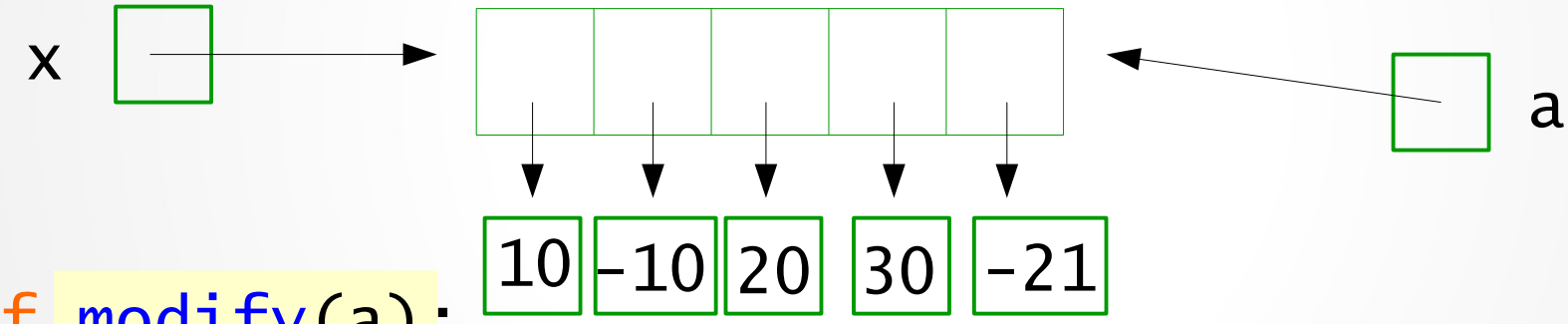

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

The initial list x: [10, -10, 20, 30, -21]

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



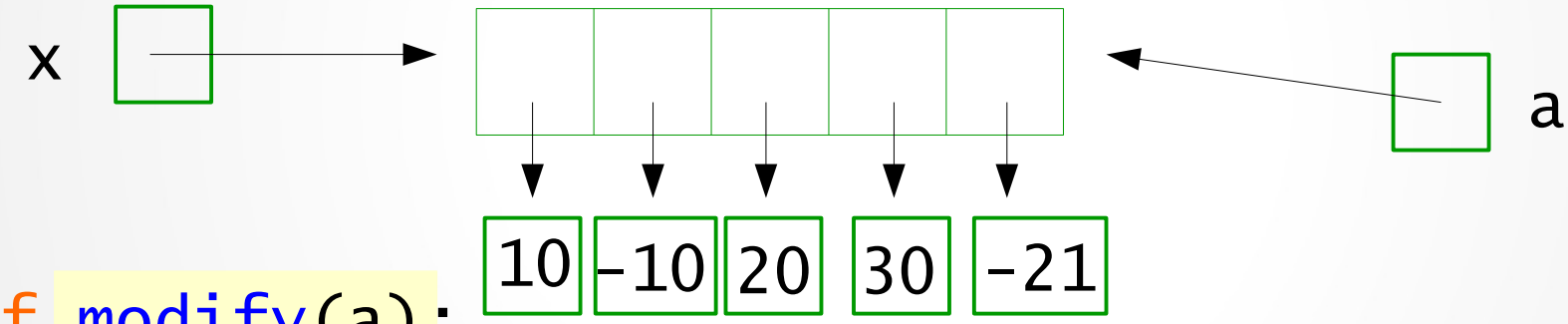

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



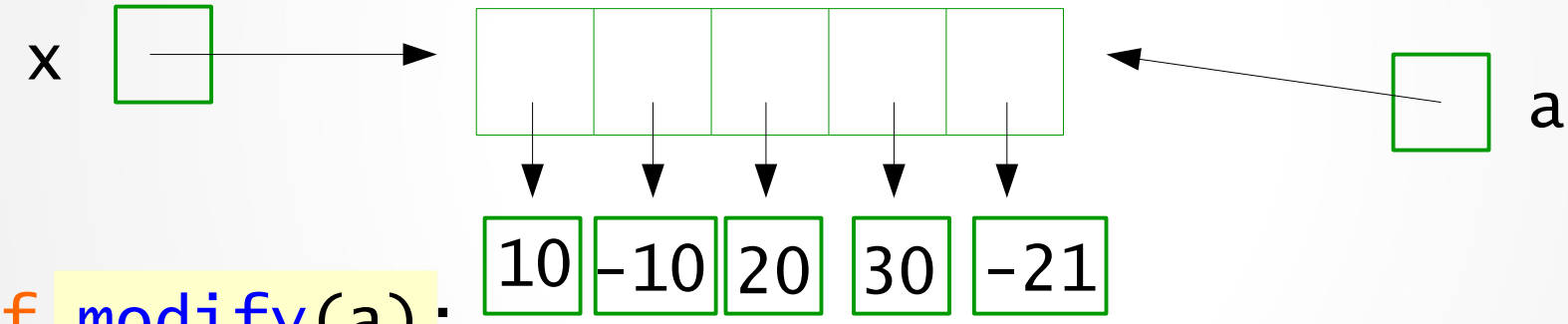

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

The initial list x: [10, -10, 20, 30, -21]

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



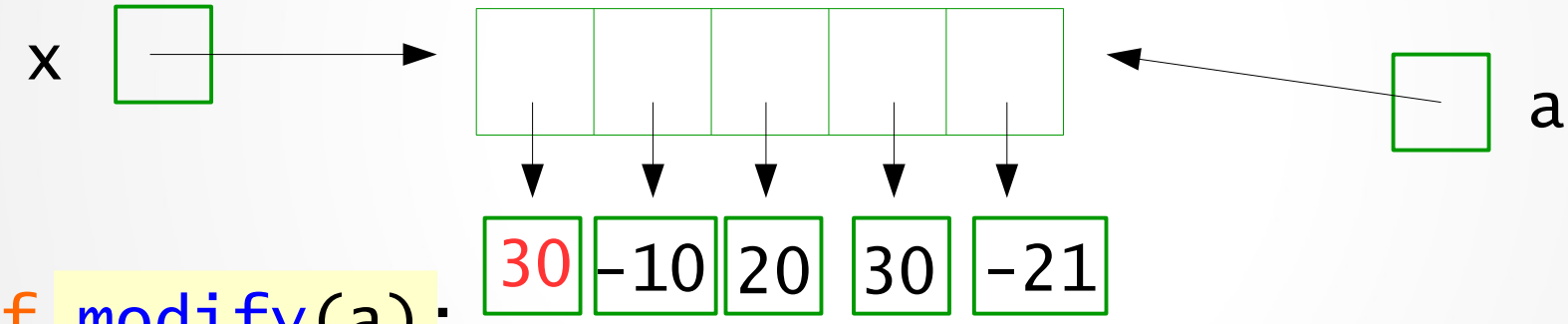

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to 10 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



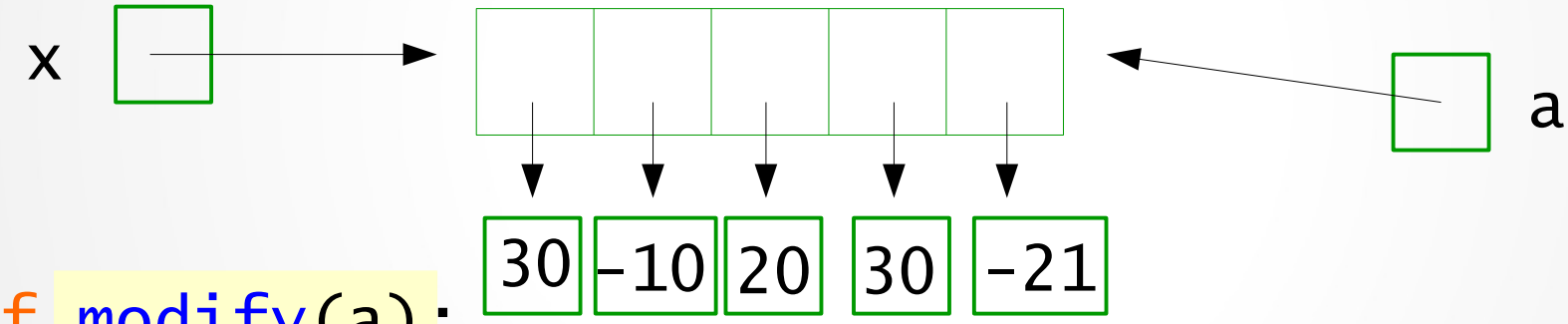

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

The initial list x: [10, -10, 20, 30, -21]
Adding 20 to 10 ...

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



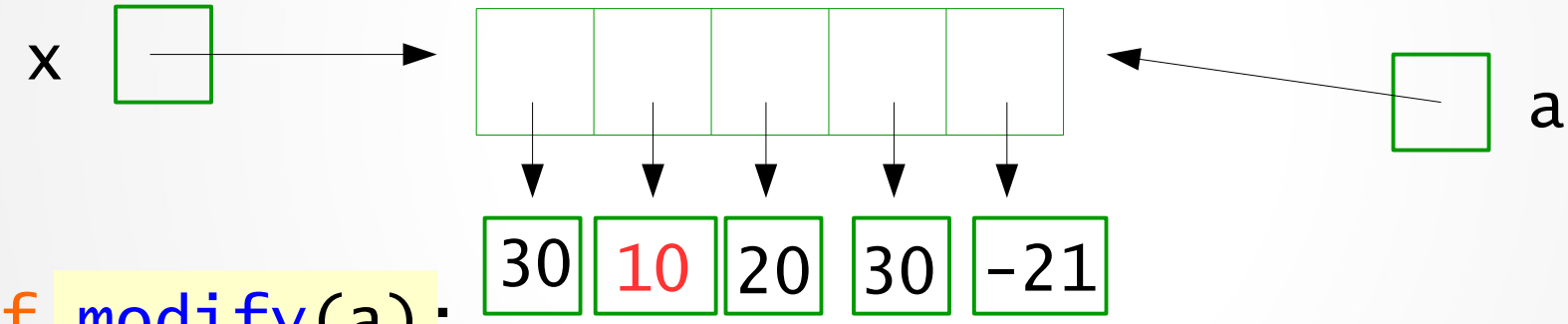

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to -10 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



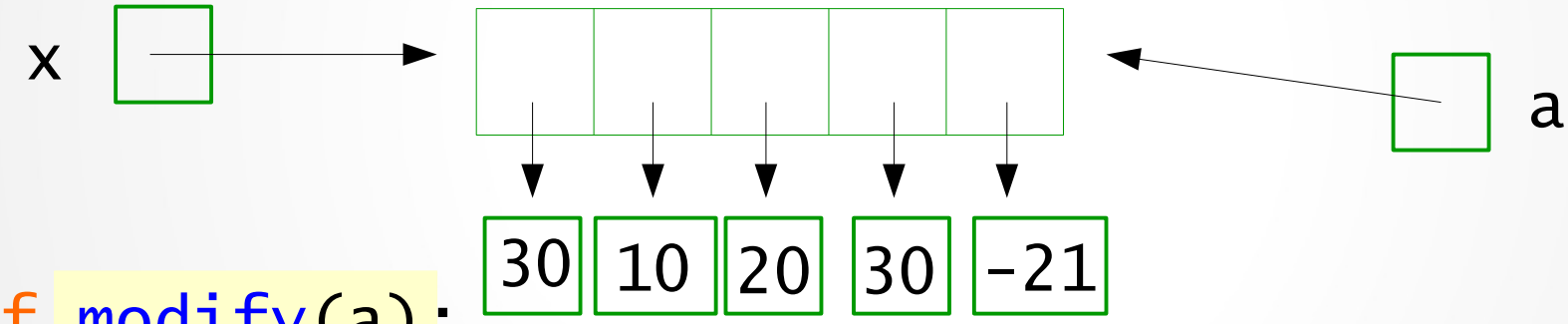

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to -10 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



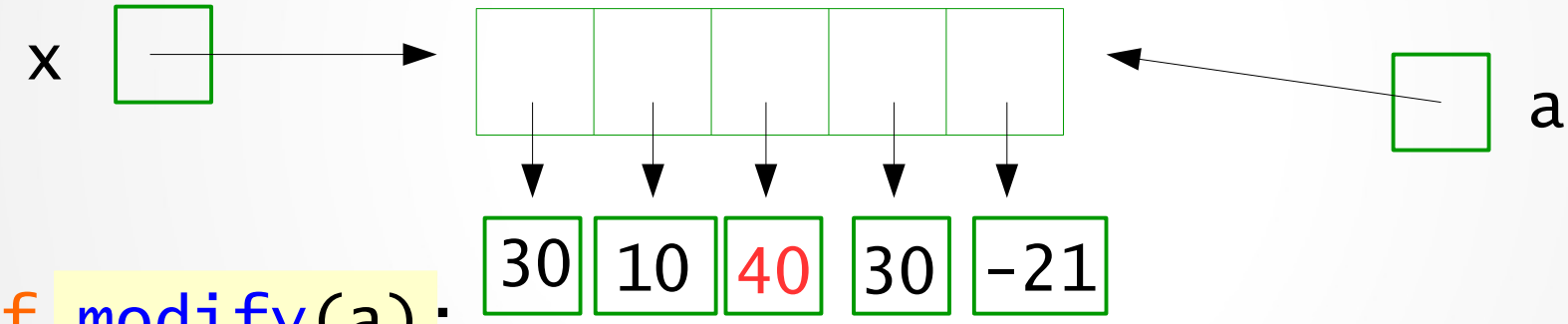

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to 20 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



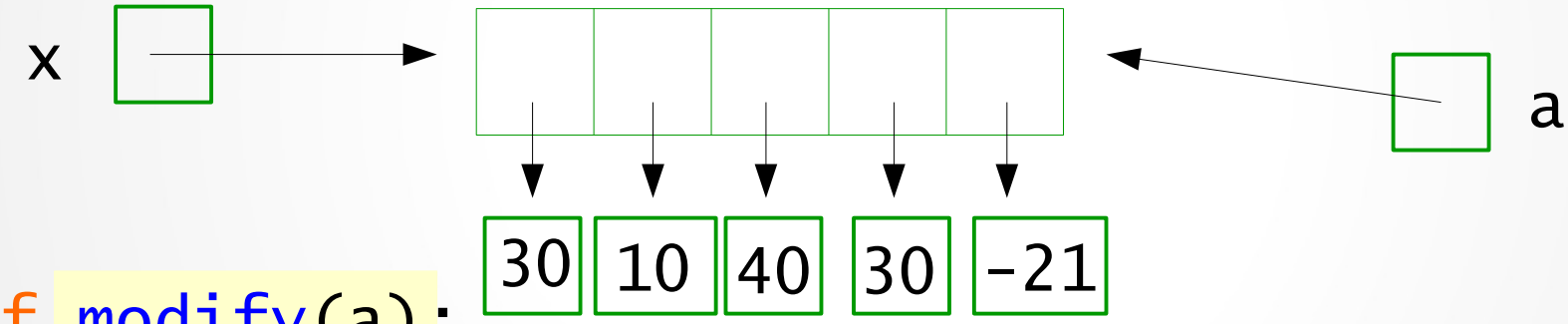

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

The initial list x: [10, -10, 20, 30, -21]
Adding 20 to 20 ...

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



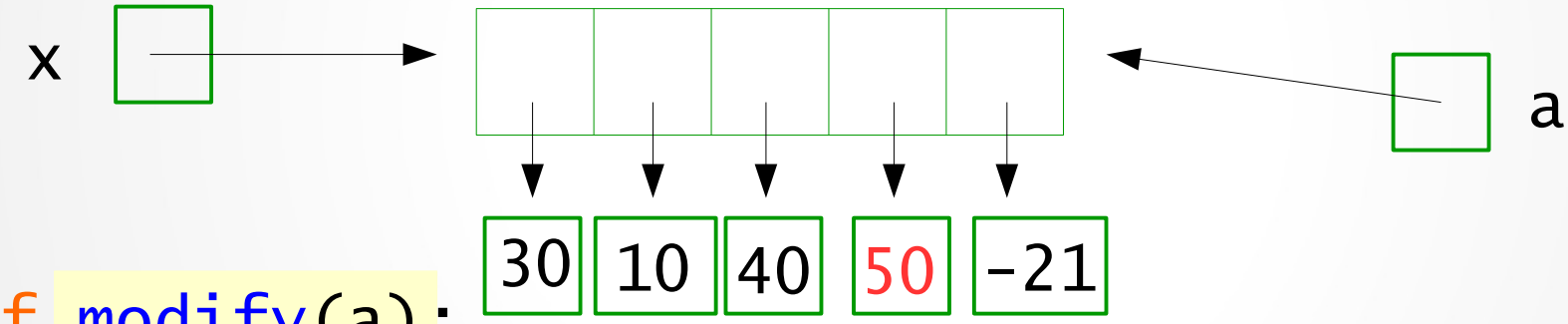

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to 30 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



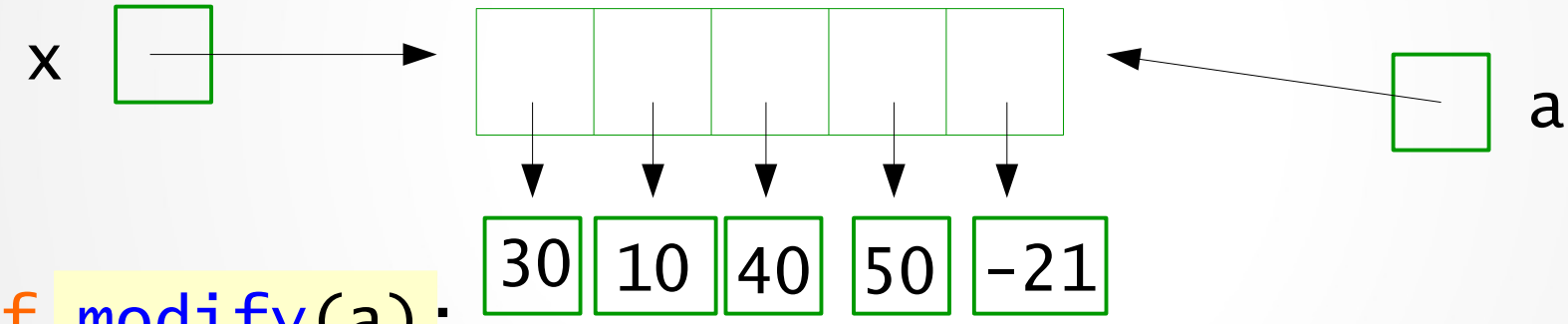

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to 30 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



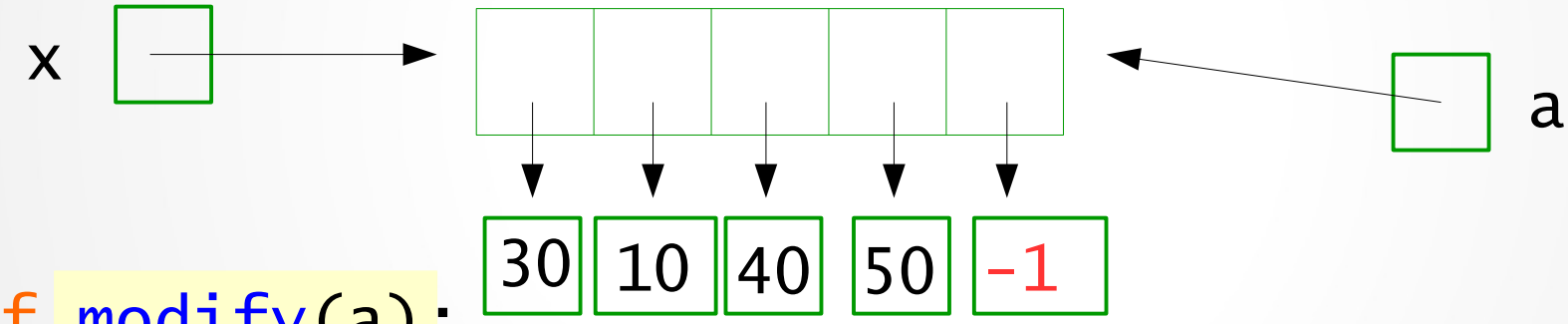

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to -21 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



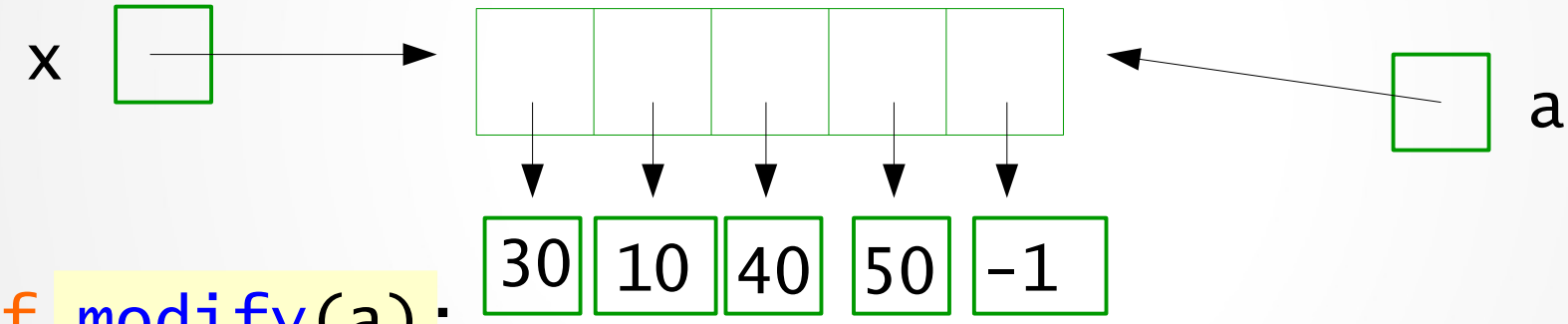

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
Adding 20 to -21 ...
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```



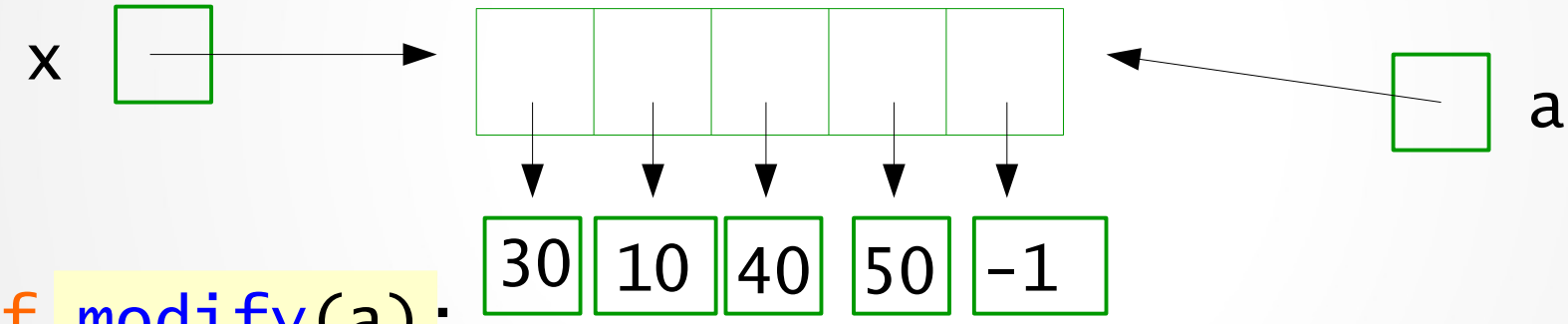

```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
```

6.6 Functions That Modify Parameters

Functions may modify variables that are mutable objects

```
def main():
    x=[10,-10,20,30,-21]
    print("The initial list x: \t \t ",x)
    modify(x)
    print("The list x after modify(x) call:",x)
```




```
def modify(a):
    for i in range(5):
        print("Adding 20 to",a[i],"...")
        a[i]=a[i]+20

main()
```

```
The initial list x:          [10, -10, 20, 30, -21]
The list x after modify(x) call: [30, 10, 40, 50, -1]
```

6.7 Functions and Program Structure

Why do we use functions?

- To reduce code duplication *(already discussed)*
  easier to write, read, test, and to update/modify

- To make programs more *modular:*
  As the algorithms we design get more complex, it gets more and more difficult to make sense of programs. One way to deal with this complexity is to break an algorithm into smaller subprograms.

6.7 Functions and Program Structure

Local variables and scope of variables

```
def main():
    x = [10,-10,20,30,-21]
    a = 10
    print("x = ",x)
    modify(a,x)
    print("after modify(a,x) call:",a,x)
    c = x.append(a)

def modify(m,x):
    p = m
    m += 10
    x.append(m)
    x = [1,2,3]
```

x, a and c are *local* to function main

m, x and p are *local* to function modify

6.7 Functions and Program Structure

Local variables and scope of variables

scope of variable x

```
def main():
    x = [10,-10,20,30,-21]
    a = 10
    print("x = ",x)
    modify(a,x)
    print("after modify(a,x) call:",a,x)
    c = x.append(a)

def modify(m,x):
    p = m
    m += 10
    x.append(m)
    x = [1,2,3]
```

6.7 Functions and Program Structure

Local variables and scope of variables

scope of variable a

```
def main():
    x = [10,-10,20,30,-21]
    a = 10
    print("x = ",x)
    modify(a,x)
    print("after modify(a,x) call:",a,x)
    c = x.append(a)

def modify(m,x):
    p = m
    m += 10
    x.append(m)
    x = [1,2,3]
```

6.7 Functions and Program Structure

Local variables and scope of variables

```
def main():
    x = [10,-10,20,30,-21]
    a = 10
    print("x = ",x)
    modify(a,x)
    print("after modify(a,x) call:",a,x)
    c = x.append(a)

def modify(m,x):
    p = m
    m += 10
    x.append(m)
    x = [1,2,3]
```

scope of variable c

6.7 Functions and Program Structure

Local variables and scope of variables

```
def main():
    x = [10,-10,20,30,-21]
    a = 10
    print("x = ",x)
    modify(a,x)
    print("after modify(a,x) call:",a,x)
    c = x.append(a)

def modify(m,x):
    p = m
    m += 10
    x.append(m)
    x = [1,2,3]
```

scope of variables m and x

6.7 Functions and Program Structure

Local variables and scope of variables

```
def main():
    x = [10,-10,20,30,-21]
    a = 10
    print("x = ",x)
    modify(a,x)
    print("after modify(a,x) call:",a,x)
    c = x.append(a)

def modify(m,x):
    p = m
    m += 10
    x.append(m)
    x = [1,2,3]
```

scope of variable p

Testing and debugging

Testing is an essential part of a program developing process.
A program cannot be released unless it passes some testing.

An error in a program is called a bug.
The process of finding and eliminating the bugs is called debugging.

Here is an example of how to write small test functions: see tests.py