

# Objectives

- To provide examples of computer science in the real world
- To provide an overview of common problem-solving strategies
- To introduce Python's numeric data types
- To show examples of simple programs
- To introduce loops and simple functions
- To introduce turtle graphics

# Computer Science

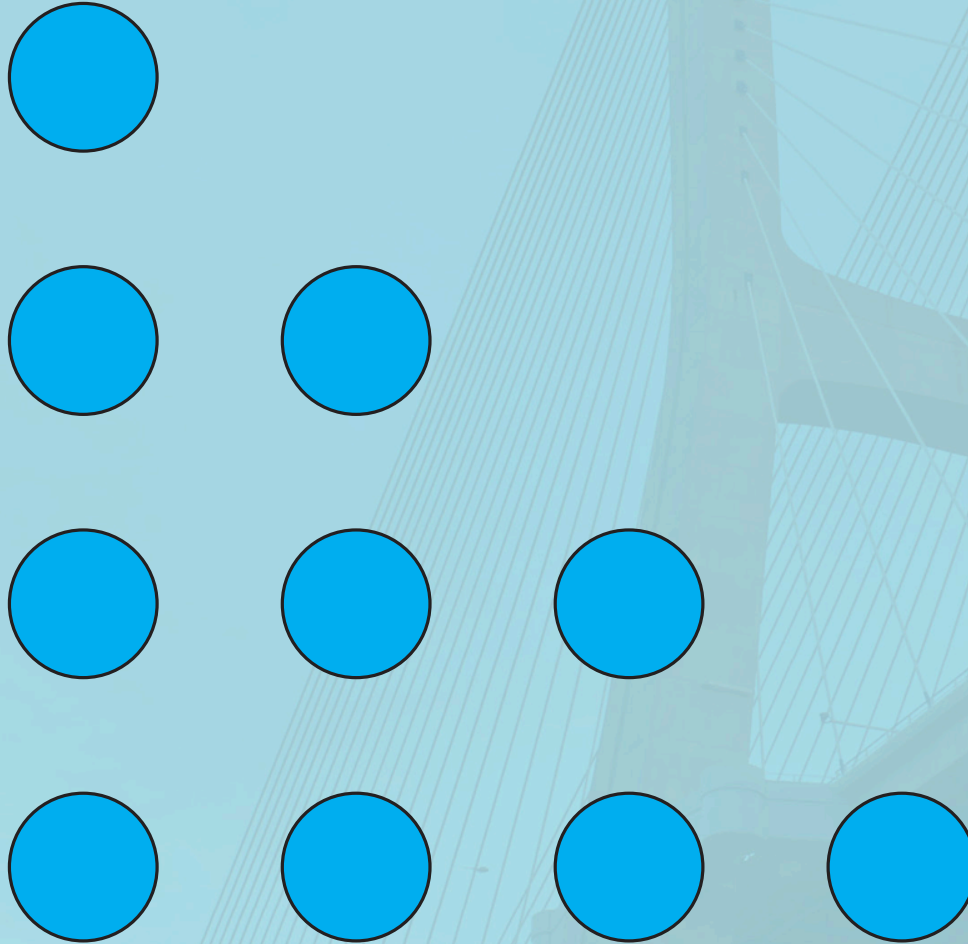
- Problem Solving
- Algorithms
- Abstraction
- Programming



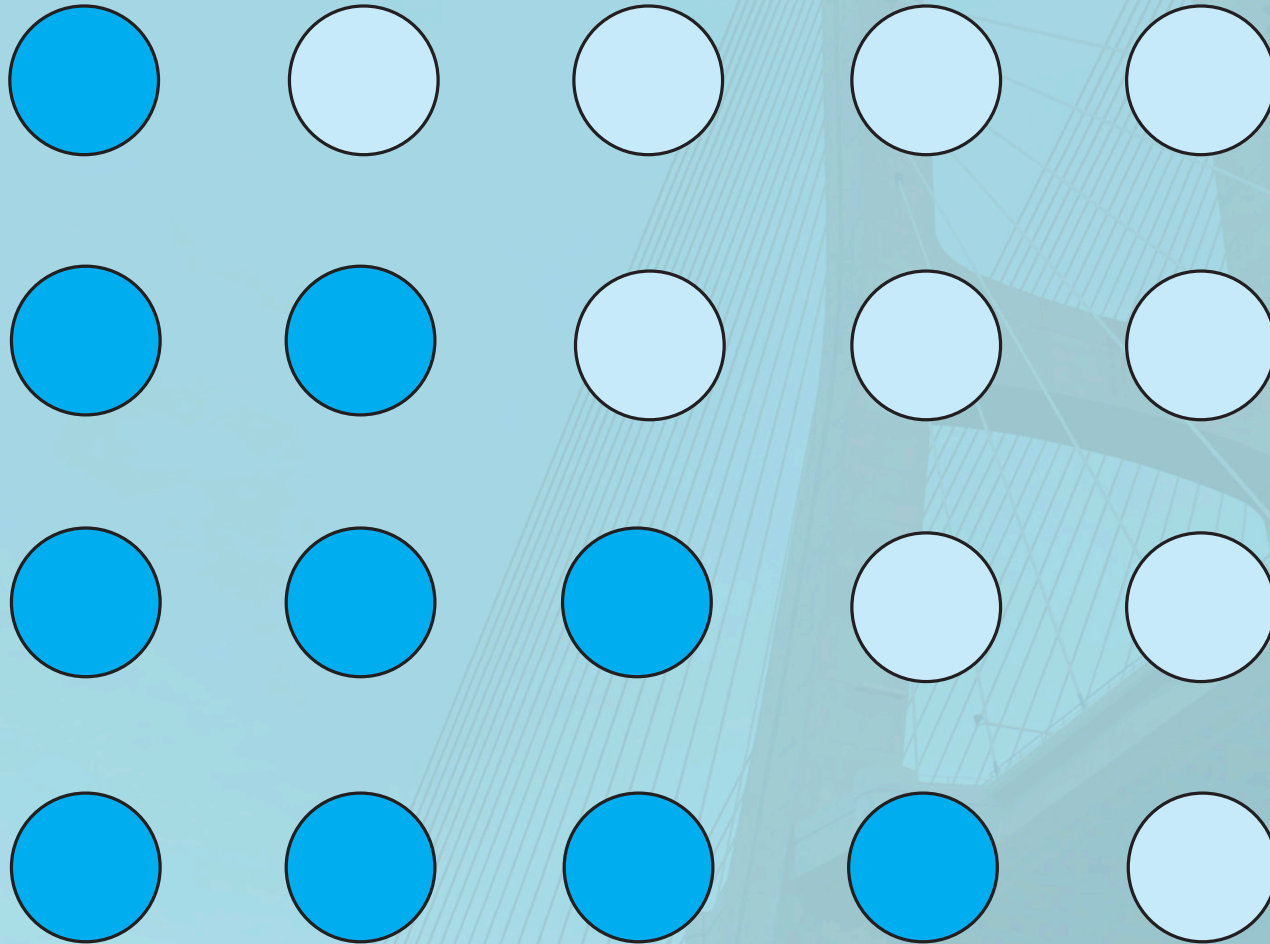
# Problem Solving

- Simplification
- Generalization

# Figure 1.1



# Figure 1.2





# Python Overview

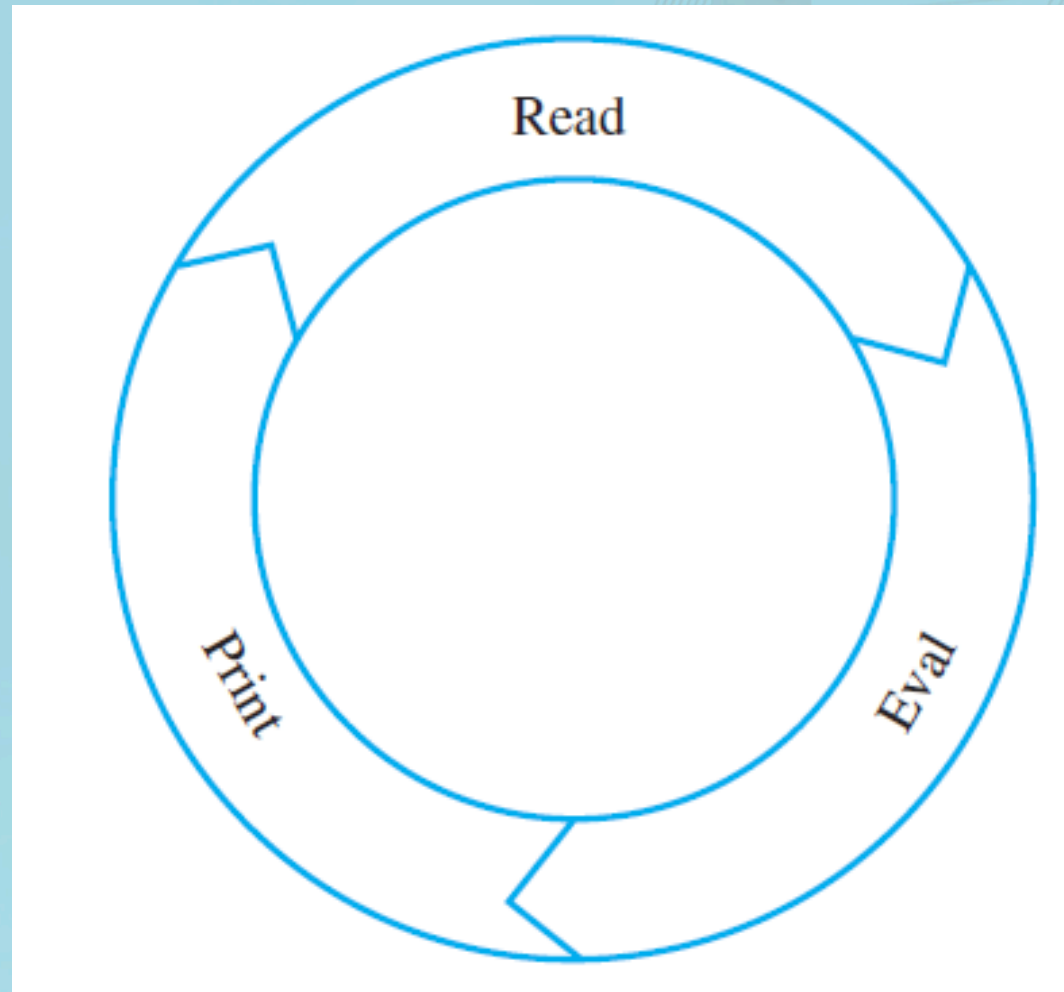
- Data Objects
- Operators
- Expressions
- Assignment Statements (variables, names)
- Python Interpreter (read, evaluate, print)

# Figure 1.3

A screenshot of a terminal window titled "davidranum — Python — 102x27". The terminal shows the command "python3" being executed, which outputs the Python version and build information: "Python 3.2.3 (v3.2.3:3d0686d90f55, Apr 10 2012, 11:25:50)", "[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin", and "Type 'help', 'copyright', 'credits' or 'license' for more information." The prompt ">>>" is followed by a cursor.

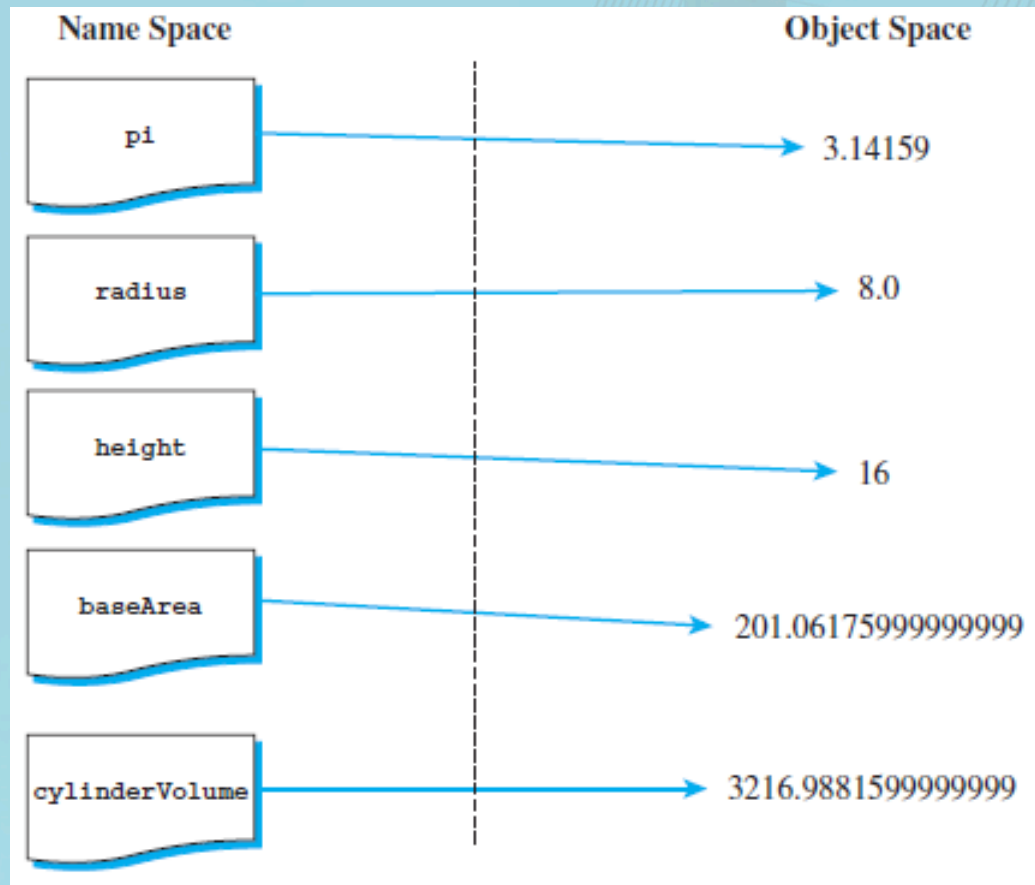
```
david-ranums-powerbook-g4-98:~ ranumday$ python3
Python 3.2.3 (v3.2.3:3d0686d90f55, Apr 10 2012, 11:25:50)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

# Figure 1.4

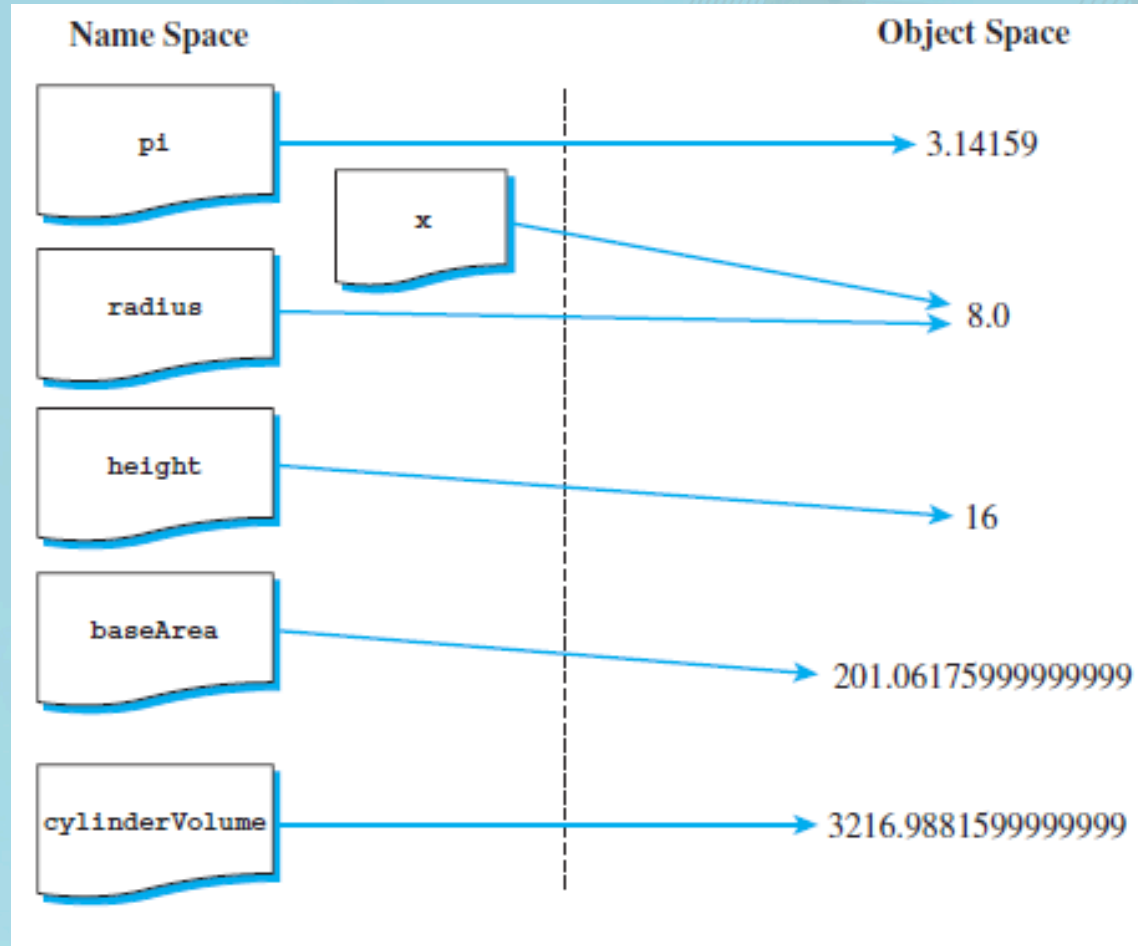




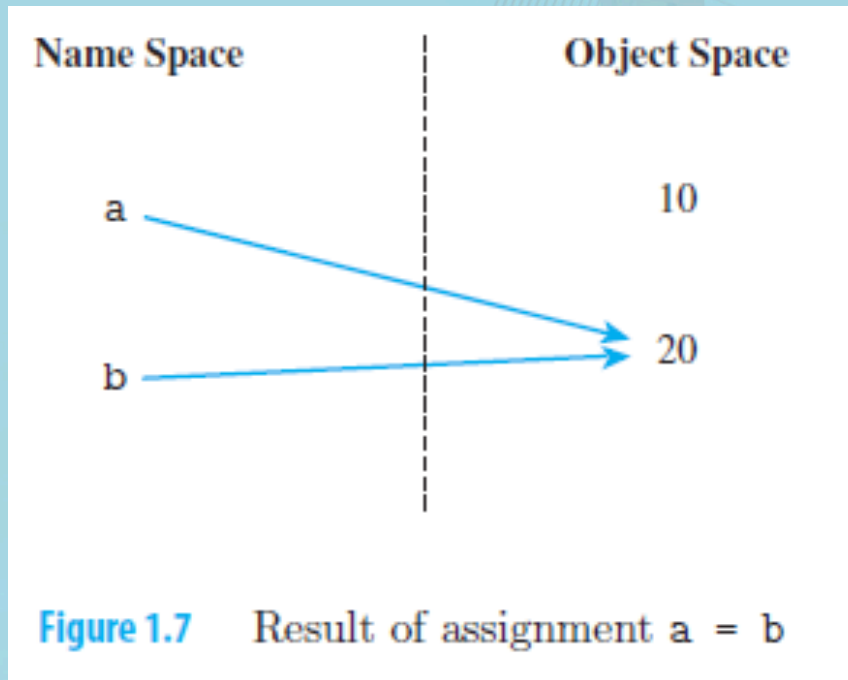
# Figure 1.5



# Figure 1.6



# Figure 1.7



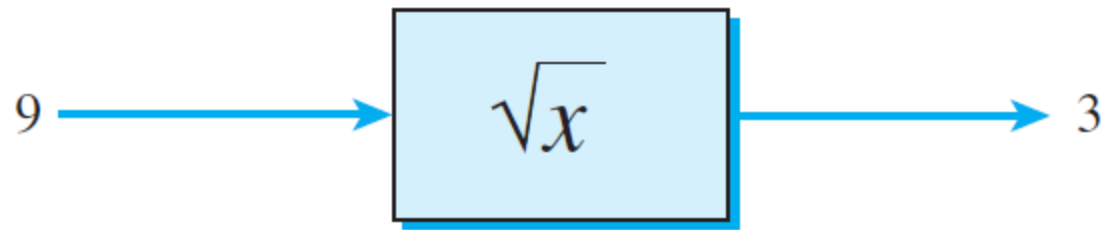
**Figure 1.7** Result of assignment `a = b`



# Abstraction and Functions

- Black Box
- Container for a sequence of actions
- Use the function by name

# Figure 1.8

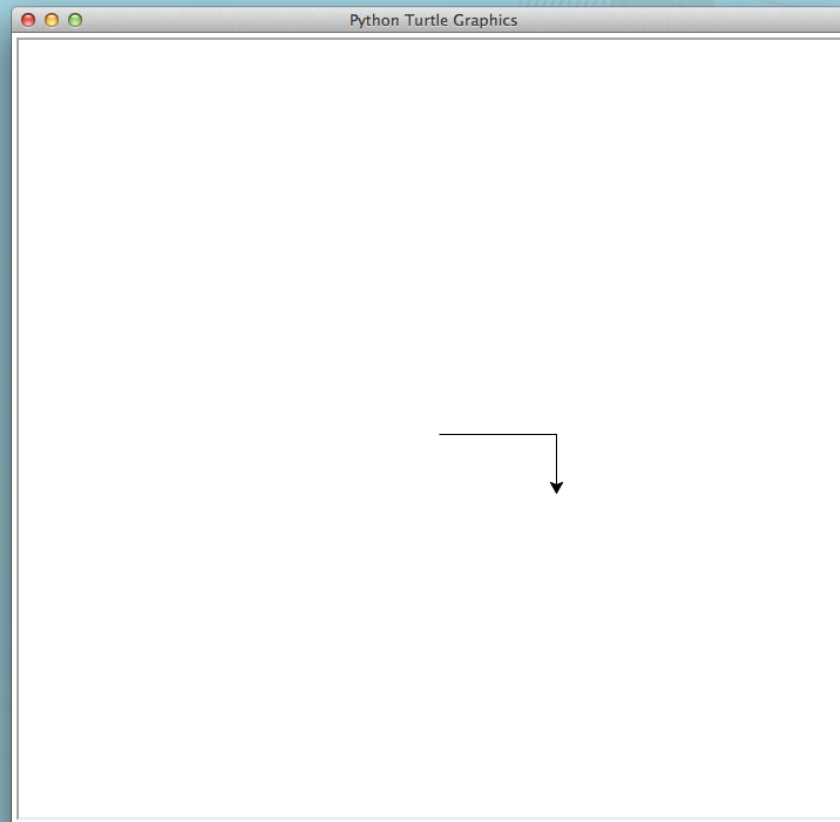


# turtle module

- Simple graphics programming
- Abstraction
- Fun and easy



# Figure 1.9



# Defining Functions

- Name
- Parameters
- Body

# Listing 1.1

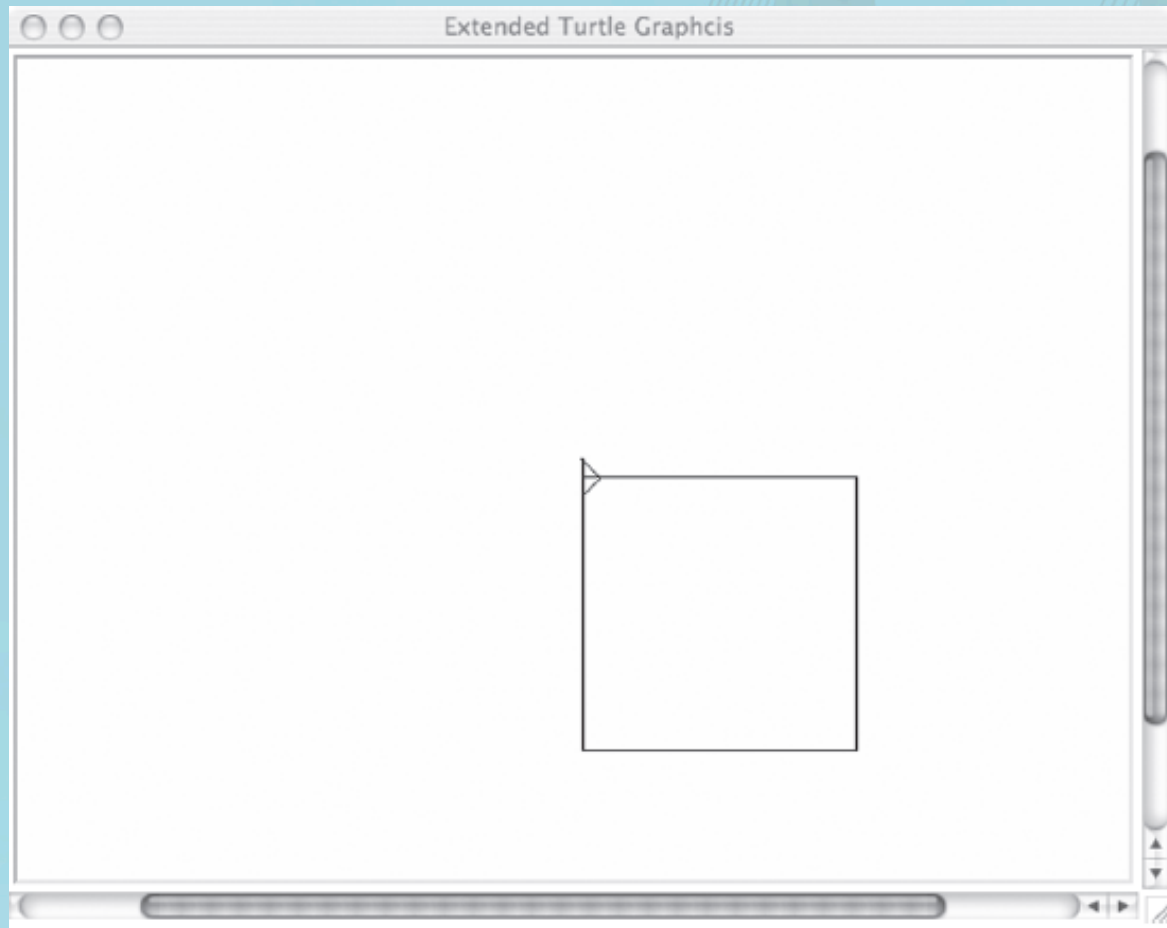
```
def functionName(param1,param2,...):  
    statement1  
    statement2  
    ...
```

## Listing 1.2

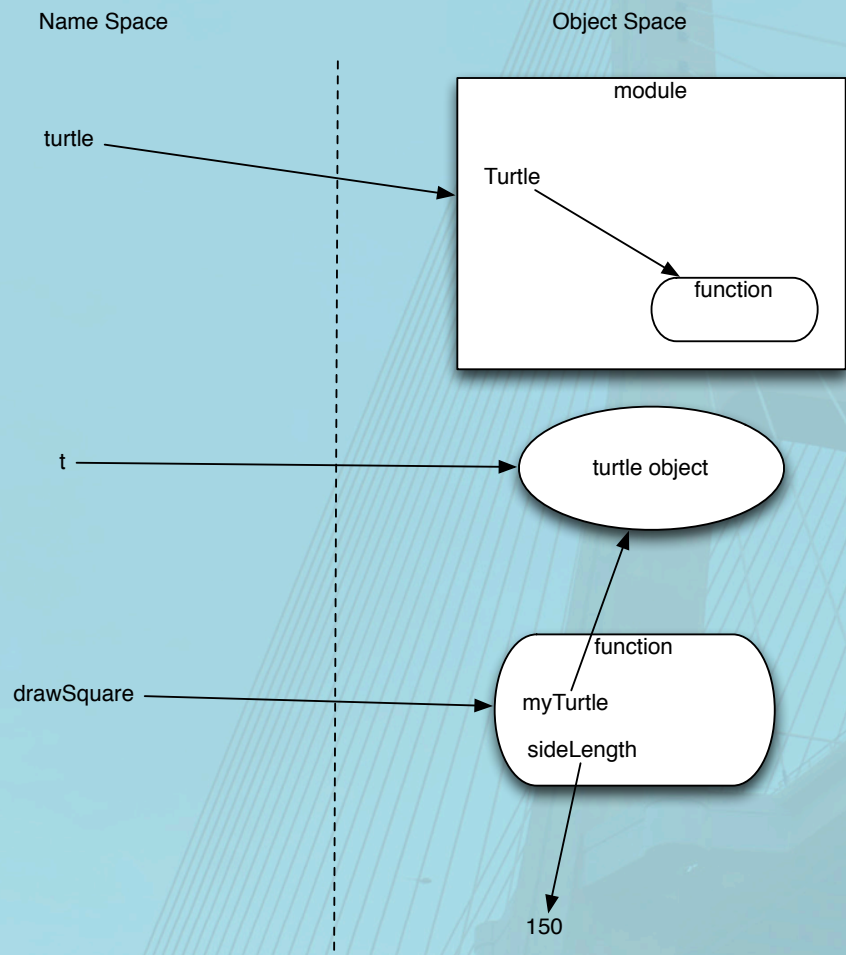
```
def drawSquare(myTurtle,sideLength):  
    myTurtle.forward(sideLength)  
    myTurtle.right(90) # side 1  
    myTurtle.forward(sideLength)  
    myTurtle.right(90) # side 2  
    myTurtle.forward(sideLength)  
    myTurtle.right(90) # side 3  
    myTurtle.forward(sideLength)  
    myTurtle.right(90) # side 4
```



# Figure 1.10



# Figure 1.11



# Iteration

- Repeat a sequence of steps
- Use a for statement
- range function

## Listing 1.3

```
def drawSquare(myTurtle,sideLength):  
    for i in range(4):  
        myTurtle.forward(sideLength)  
        myTurtle.right(90)
```

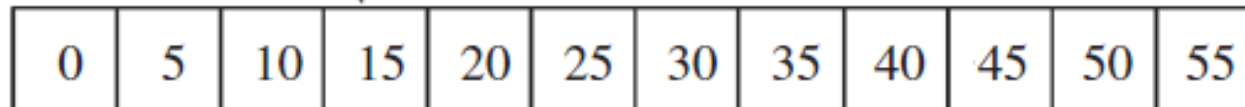


# Figure 1.12

```
for item in range(0,60,5):
```

item

fourth time through the loop



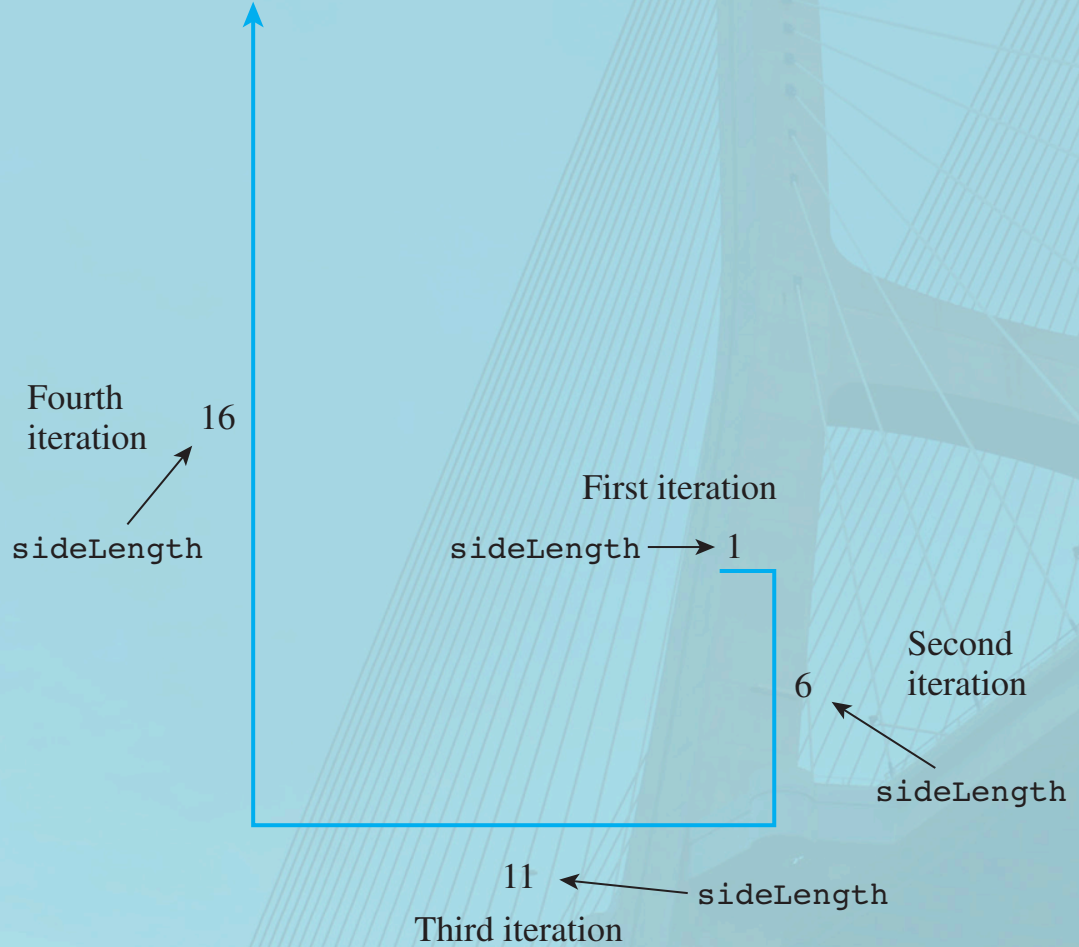
0	5	10	15	20	25	30	35	40	45	50	55
---	---	----	----	----	----	----	----	----	----	----	----

sequence produced by: `range(0,60,5)`

## Listing 1.4

```
def drawSpiral(myTurtle,maxSide):  
    for sideLength in range(1,maxSide+1,5):  
        myTurtle.forward(sideLength)  
        myTurtle.right(90)
```

# Figure 1.13



# Drawing a Circle

- Simplify and Generalize
- Polygon with more and more sides



## Listing 1.5

```
def drawTriangle(myTurtle,sideLength):  
    for i in range(3):  
        myTurtle.forward(sideLength)  
        myTurtle.right(120)
```

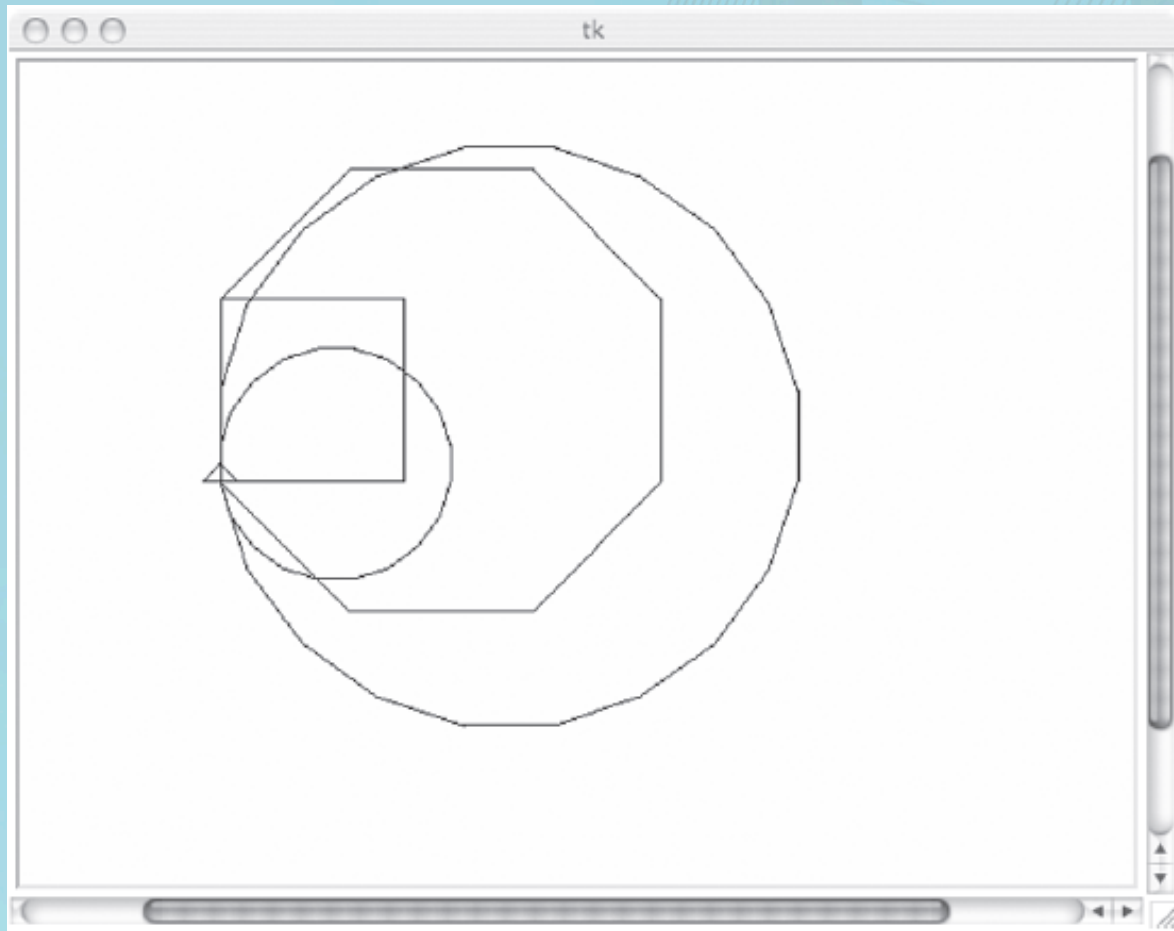
# Generalize

- 3 sides – 120 degrees
- 4 sides – 90 degrees
- 5 sides – 72 degrees
- 8 sides – 45 degrees
- N sides - ? Degrees

# Listing 1.6

```
def drawPolygon(myTurtle,sideLength,numSides):  
    turnAngle = 360 / numSides  
    for i in range(numSides):  
        myTurtle.forward(sideLength)  
        myTurtle.right(turnAngle)
```

# Figure 1.14





## Listing 1.7

```
def drawCircle(myTurtle,radius):  
    circumference = 2 * 3.1415 * radius  
    sideLength = circumference / 360  
    drawPolygon(myTurtle,sideLength,360)
```

# Figure 1.15

