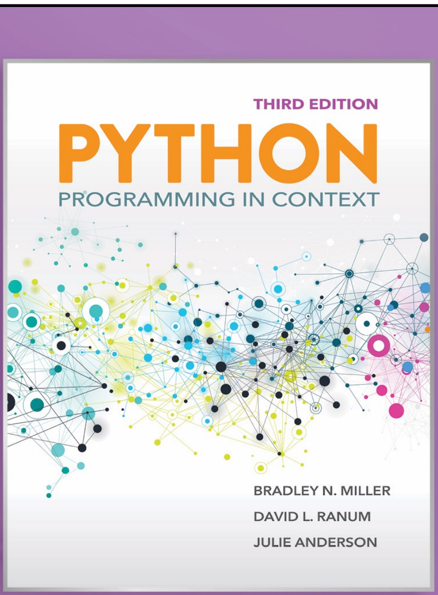
Chapter 1

Introduction to Numeric Types, Turtle Graphics, Simple Loops, and Functions



THIRD EDITION

PYTHON

PROGRAMMING IN CONTEXT

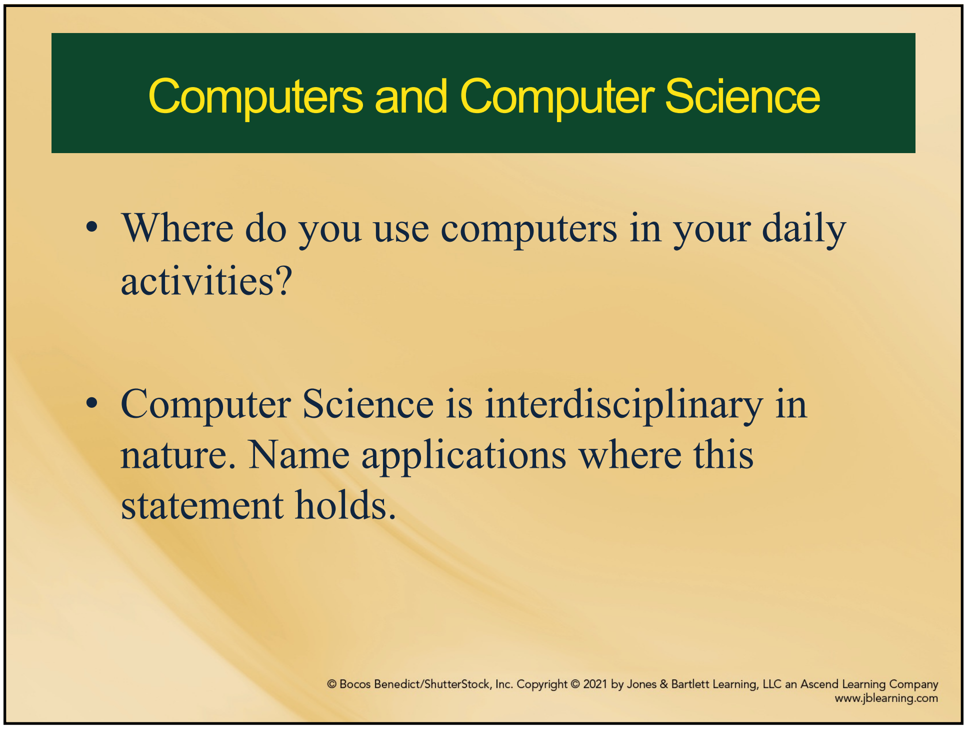
BRADLEY N. MILLER
DAVID L. RANUM
JULIE ANDERSON

1

Objectives

- To provide
  - examples of computer science in the real world
  - an overview of common problem-solving strategies
- To introduce
  - Python's numeric data types
  - Turtle graphics
  - Simple functions
  - Loops
  - Examples of simple programs

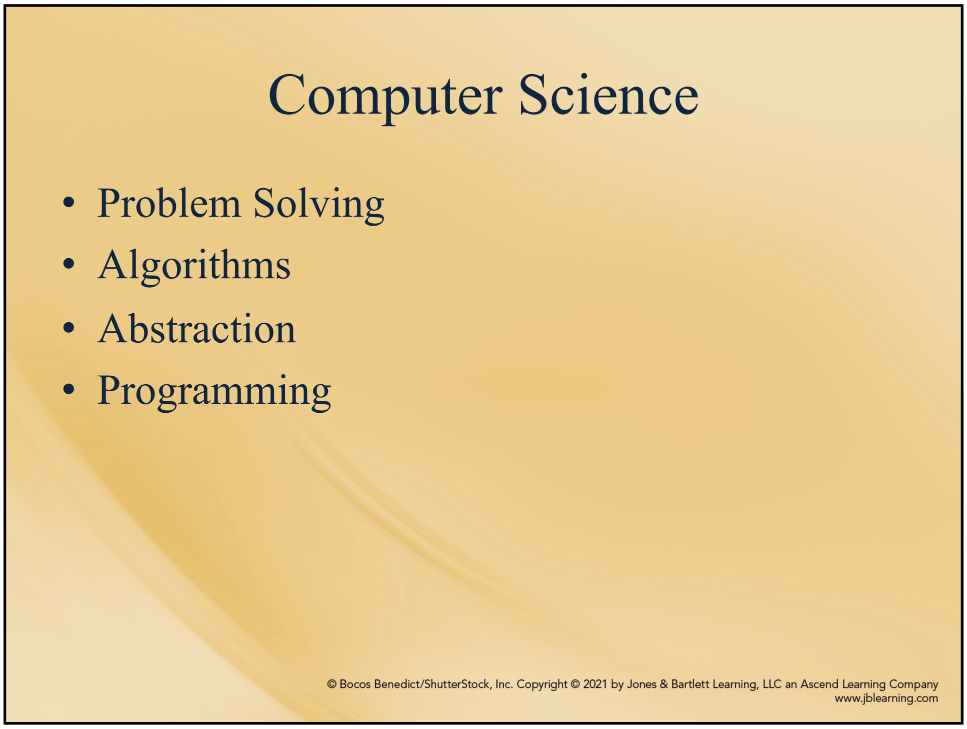
2



Computers and Computer Science

- Where do you use computers in your daily activities?
- Computer Science is interdisciplinary in nature. Name applications where this statement holds.

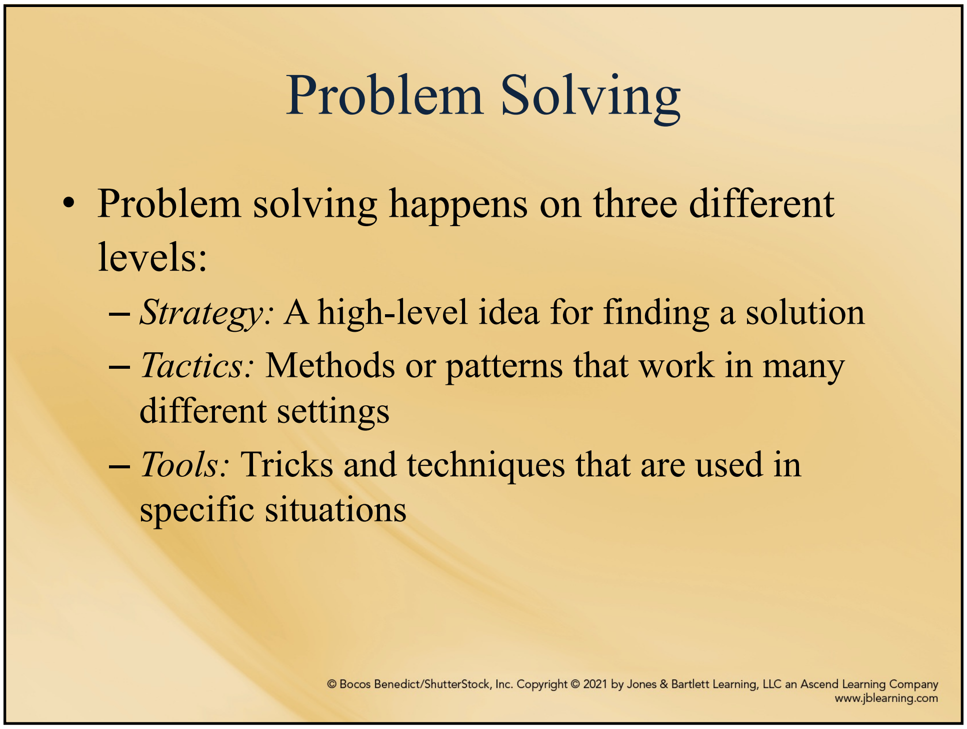
3



Computer Science

- Problem Solving
- Algorithms
- Abstraction
- Programming

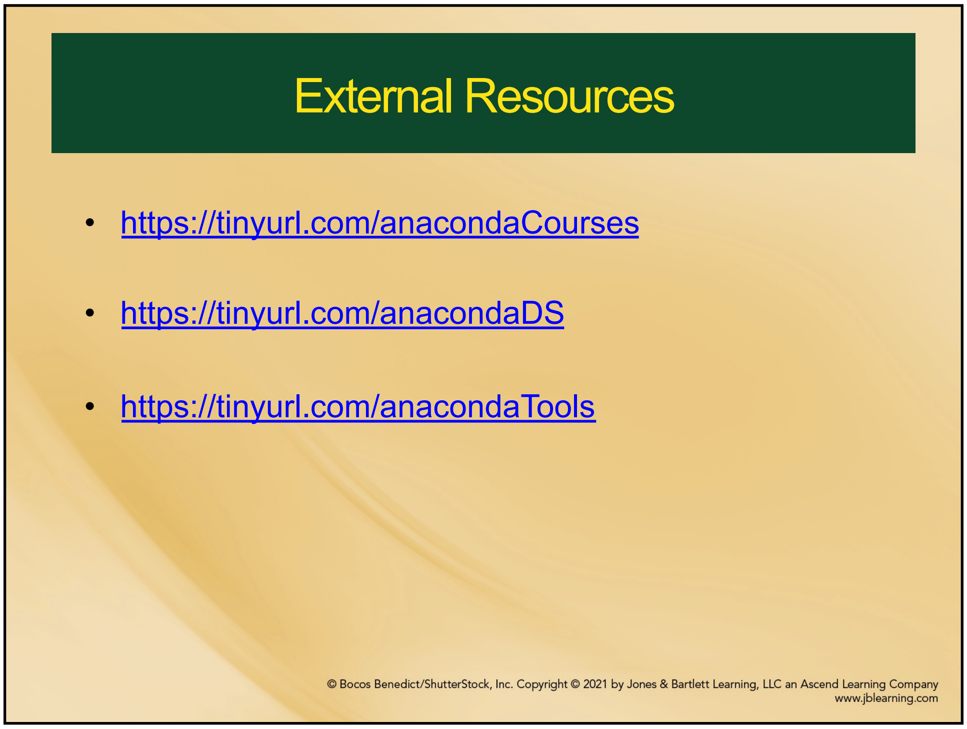
4



Problem Solving

- Problem solving happens on three different levels:
  - *Strategy:* A high-level idea for finding a solution
  - *Tactics:* Methods or patterns that work in many different settings
  - *Tools:* Tricks and techniques that are used in specific situations

5



External Resources

- https://tinyurl.com/anacondaCourses
- https://tinyurl.com/anacondaDS
- https://tinyurl.com/anacondaTools

6

Class Encore

CS 210 Leaders: Avery, Jose

Winter 2024

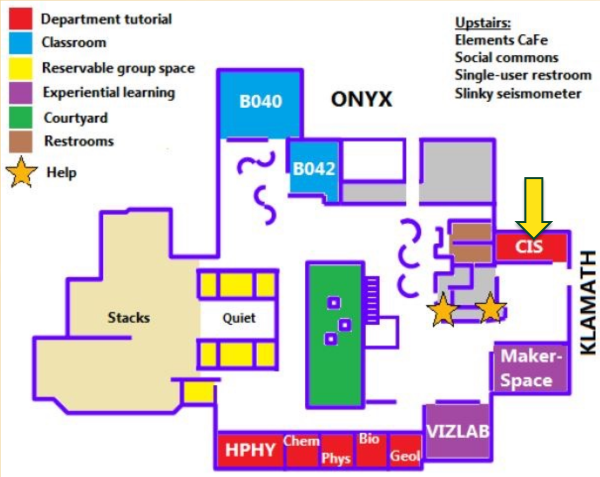
Class Encore is FREE and open to ALL students
Registration opens Friday, Jan. 12th - and stays open!
Sessions are 50 minutes & meet Weeks 2-10
Engage with peers in interactive sessions

See session times and sign up at:
engage.uoregon.edu/class-encore

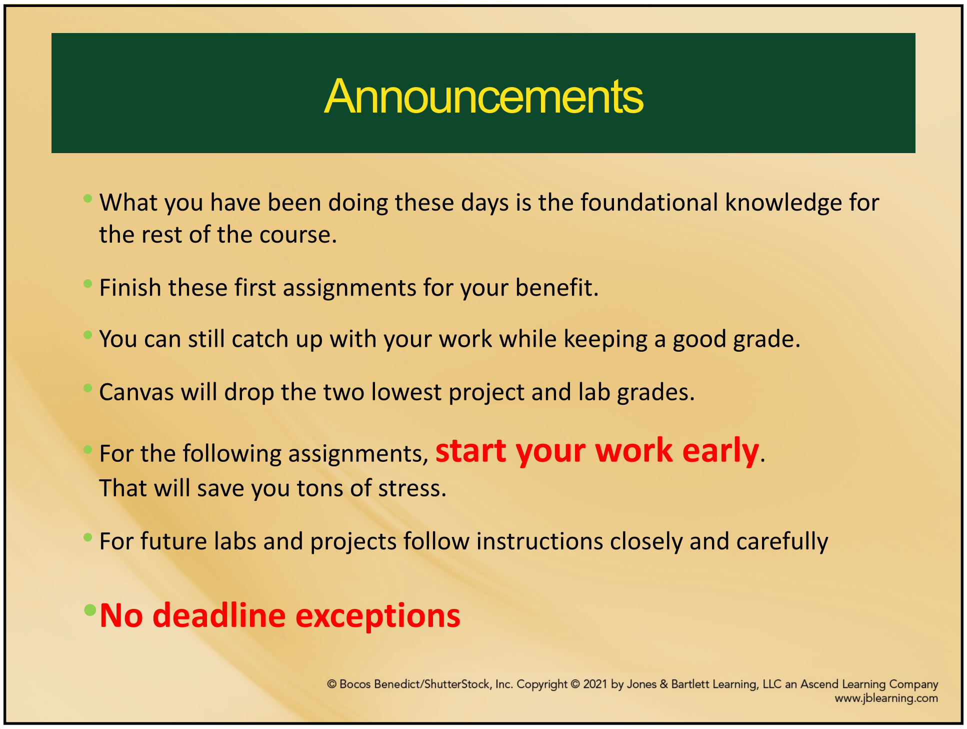
7

Announcements

- OH
- Labs



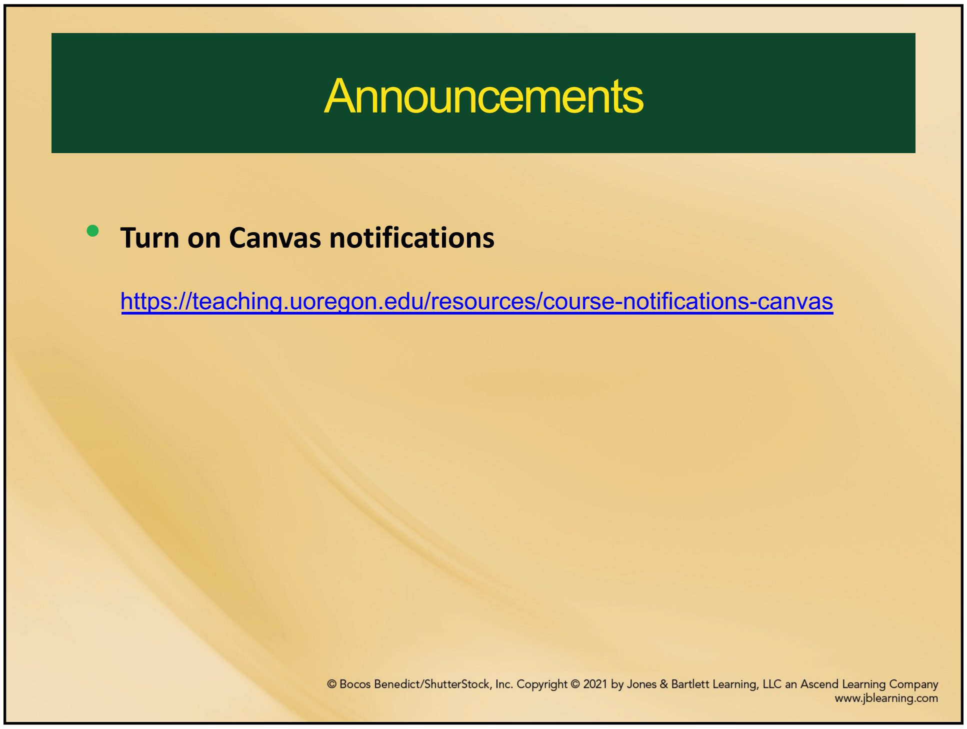
8



Announcements

- What you have been doing these days is the foundational knowledge for the rest of the course.
- Finish these first assignments for your benefit.
- You can still catch up with your work while keeping a good grade.
- Canvas will drop the two lowest project and lab grades.
- For the following assignments, **start your work early**. That will save you tons of stress.
- For future labs and projects follow instructions closely and carefully
- **No deadline exceptions**

9

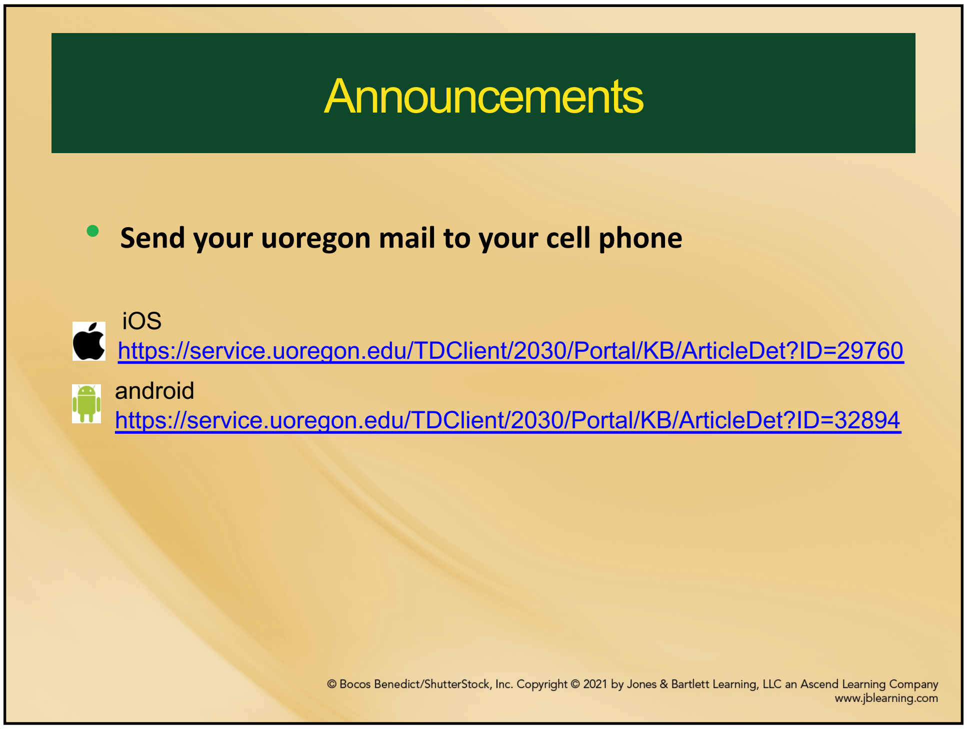


Announcements

- **Turn on Canvas notifications**

  https://teaching.uoregon.edu/resources/course-notifications-canvas

10



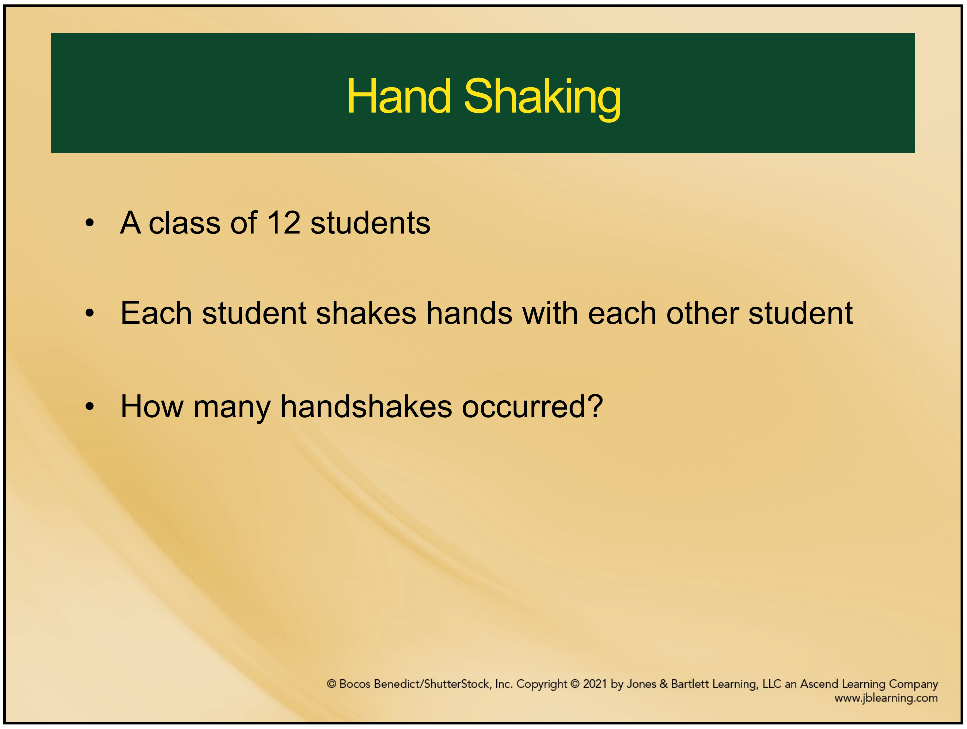
Announcements

- **Send your uoregon mail to your cell phone**

iOS
https://service.uoregon.edu/TDClient/2030/Portal/KB/ArticleDet?ID=29760

android
https://service.uoregon.edu/TDClient/2030/Portal/KB/ArticleDet?ID=32894

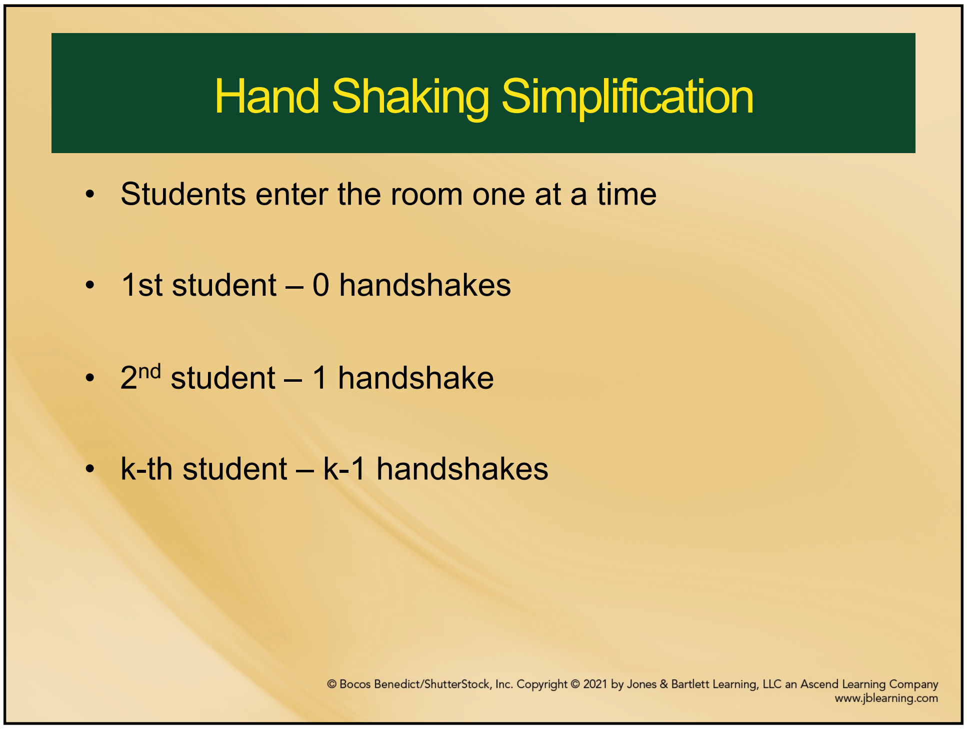
11



Hand Shaking

- A class of 12 students
- Each student shakes hands with each other student
- How many handshakes occurred?

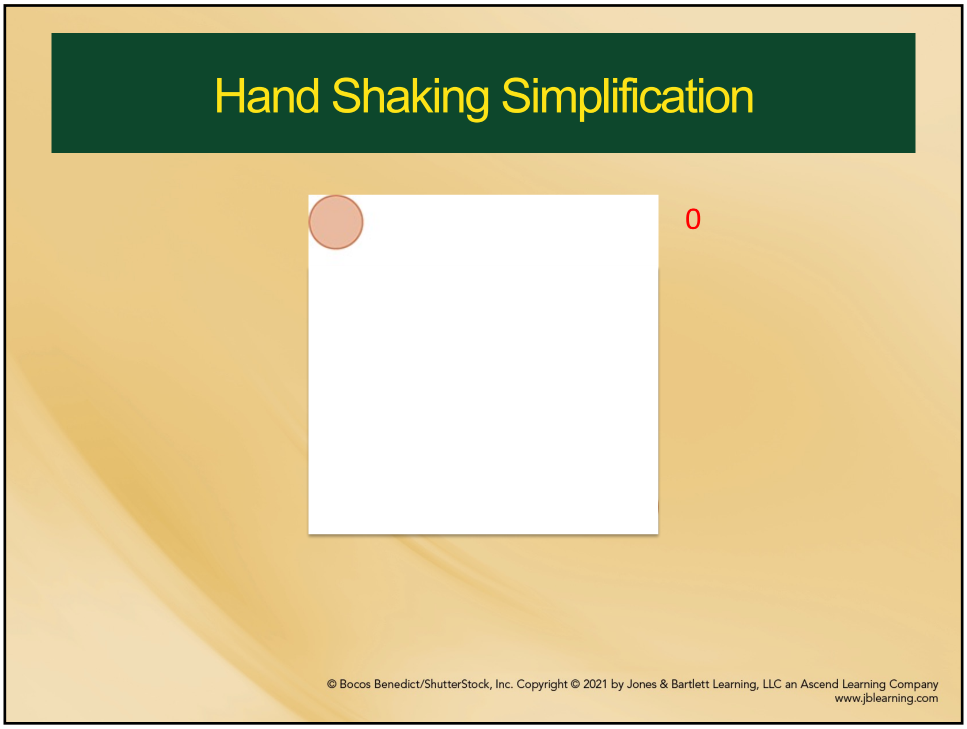
12



Hand Shaking Simplification

- Students enter the room one at a time
- 1st student – 0 handshakes
- 2^{nd} student – 1 handshake
- k-th student – k-1 handshakes

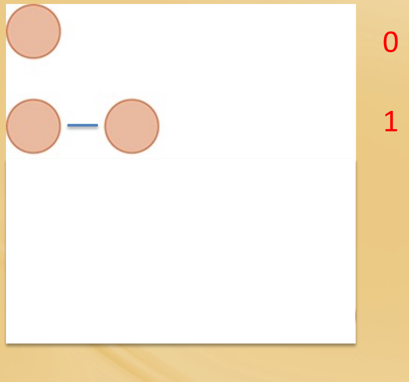
13



Hand Shaking Simplification

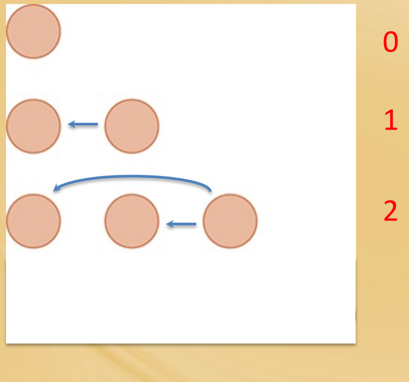
14

Hand Shaking Simplification

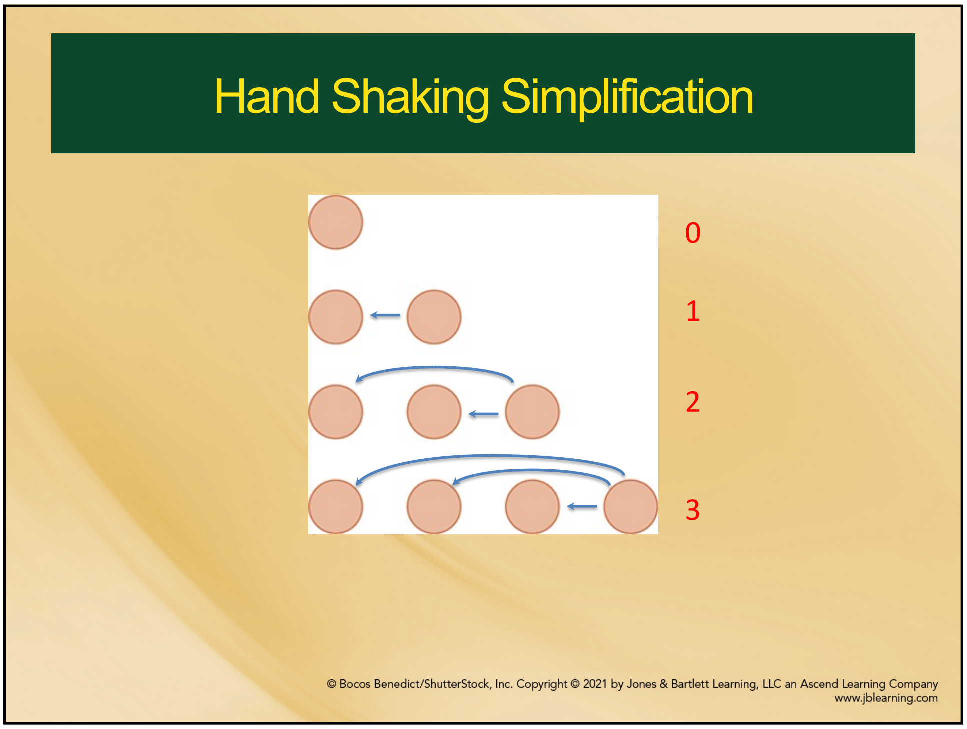


15

Hand Shaking Simplification

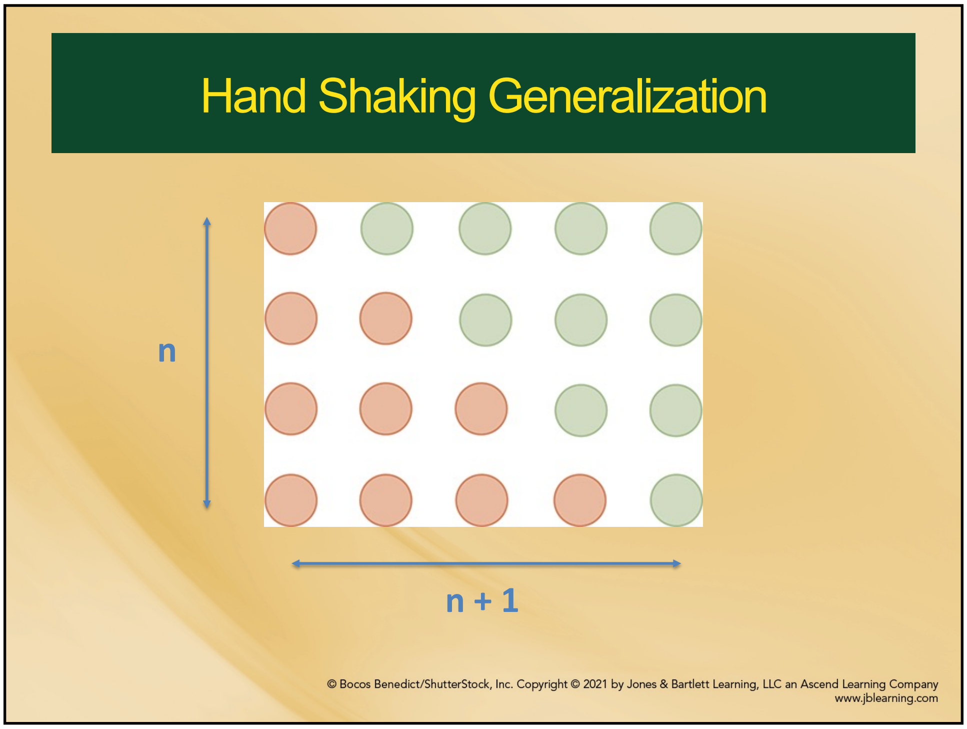


16



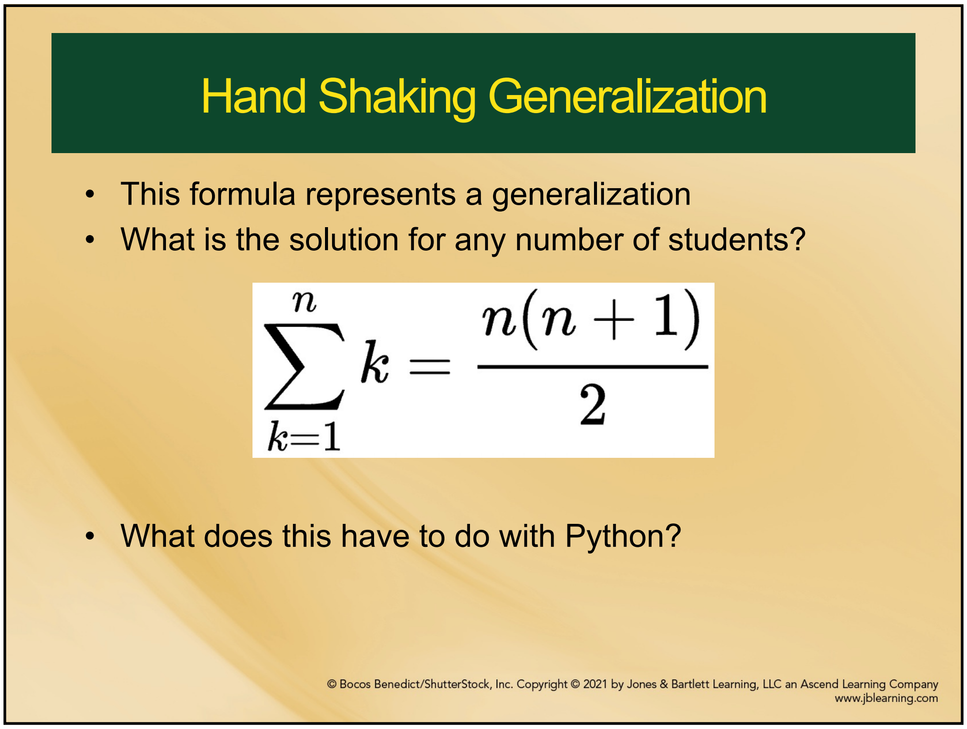
Hand Shaking Simplification

17



Hand Shaking Generalization

18



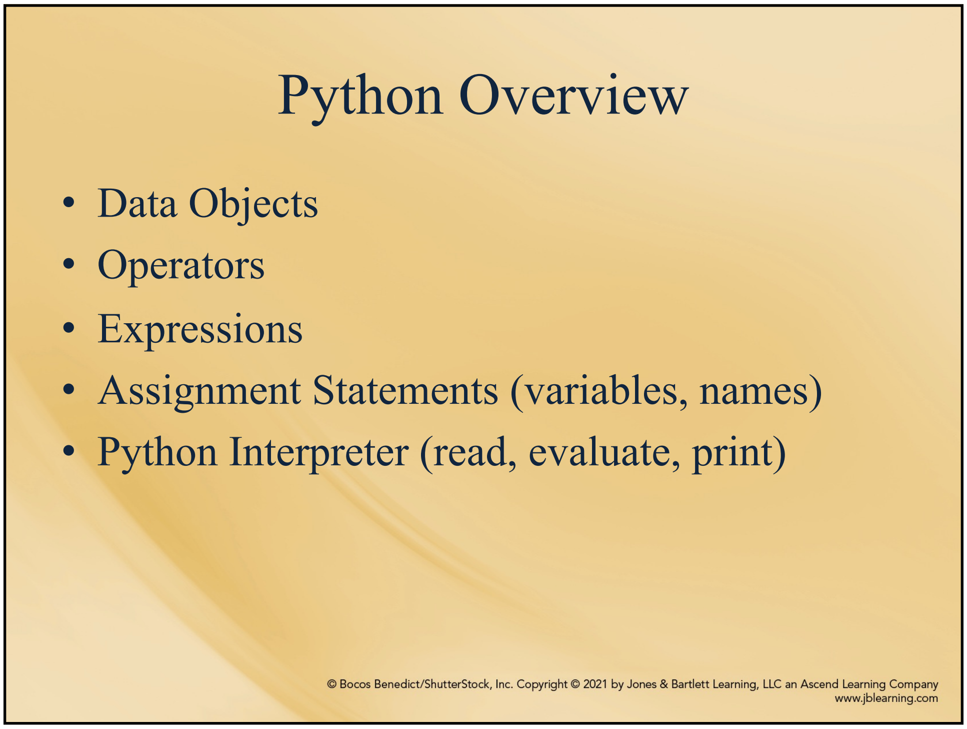
Hand Shaking Generalization

- This formula represents a generalization
- What is the solution for any number of students?

$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

- What does this have to do with Python?

19

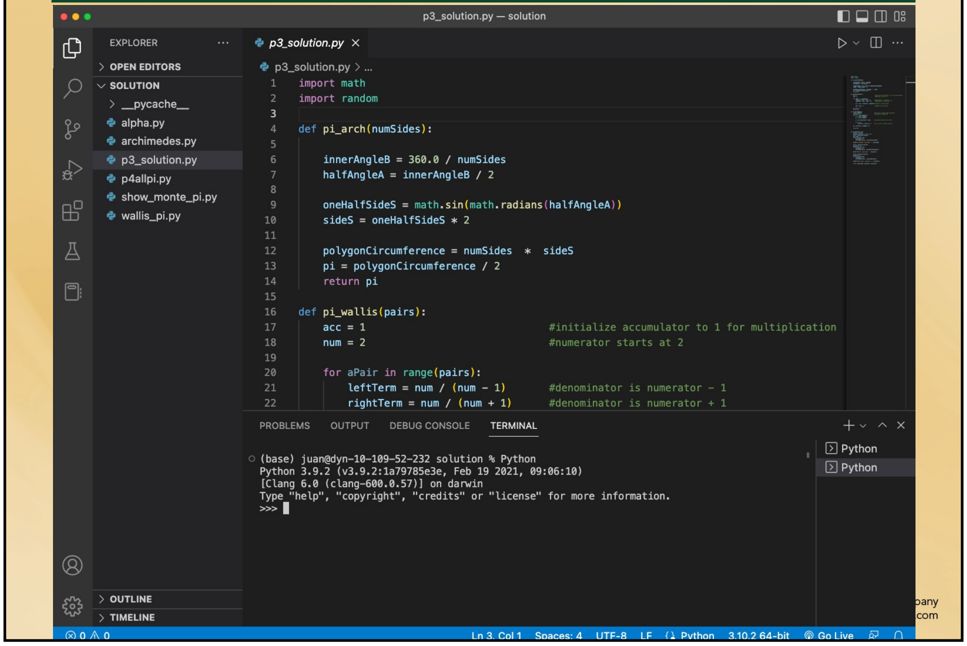


Python Overview

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

20

The VSCode Python Interpreter

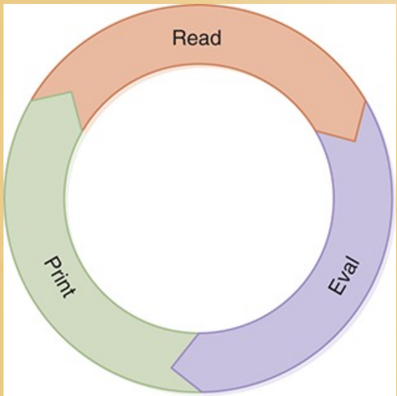


```python
import math
import random

def pi_arch(numSides):

    innerAngleB = 360.0 / numSides
    halfAngleA = innerAngleB / 2

    oneHalfSideS = math.sin(math.radians(halfAngleA))
    sideS = oneHalfSideS * 2

    polygonCircumference = numSides  *  sideS
    pi = polygonCircumference / 2
    return pi

def pi_wallis(pairs):
    acc = 1                           #initialize accumulator to 1 for multiplication
    num = 2                           #numerator starts at 2

    for aPair in range(pairs):
        leftTerm = num / (num - 1)        #denominator is numerator - 1
        rightTerm = num / (num + 1)       #denominator is numerator + 1
```

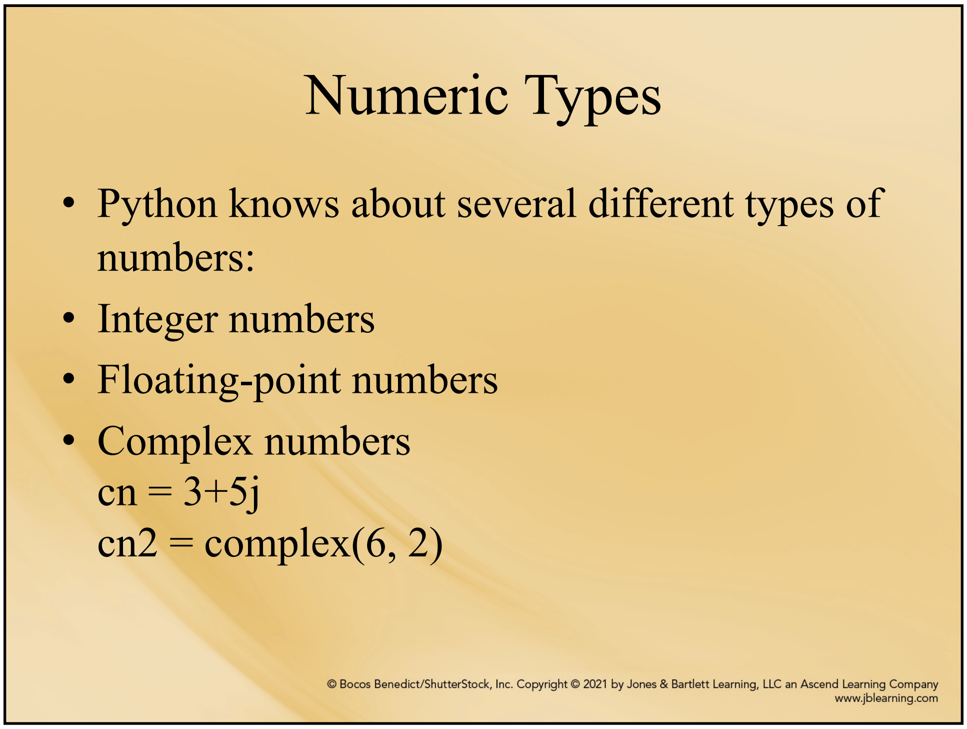
```
(base) juan@dyn-10-109-52-232 solution % Python
Python 3.9.2 (v3.9.2:1a79785e3e, Feb 19 2021, 09:06:10)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

21

The read-eval-print Loop



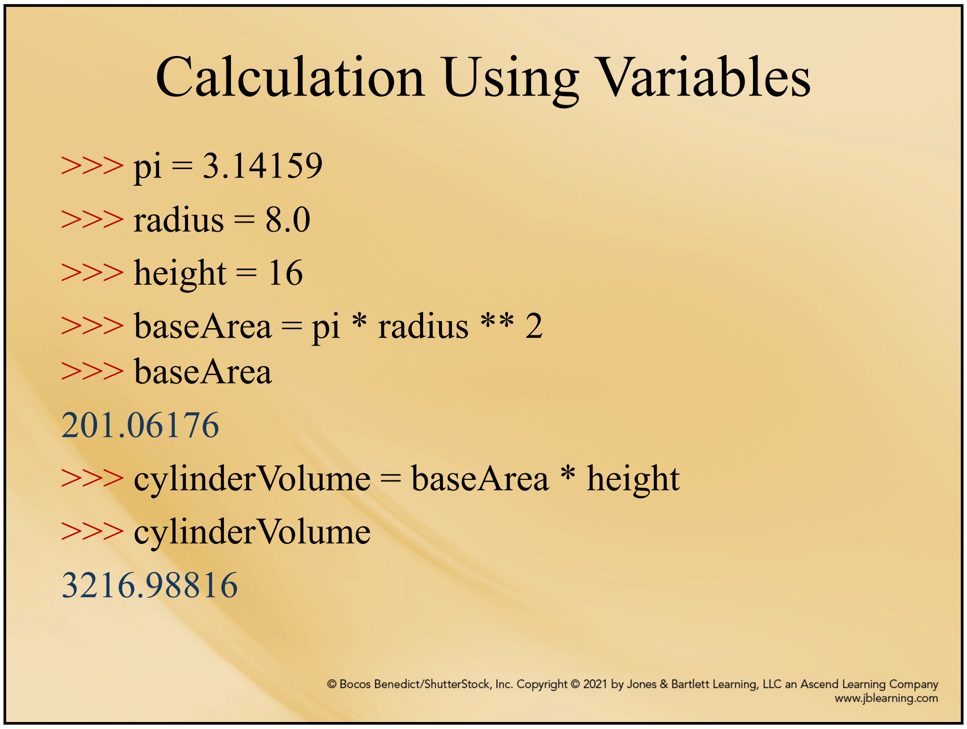
22



Numeric Types

- Python knows about several different types of numbers:
- Integer numbers
- Floating-point numbers
- Complex numbers
  cn = 3+5j
  cn2 = complex(6, 2)

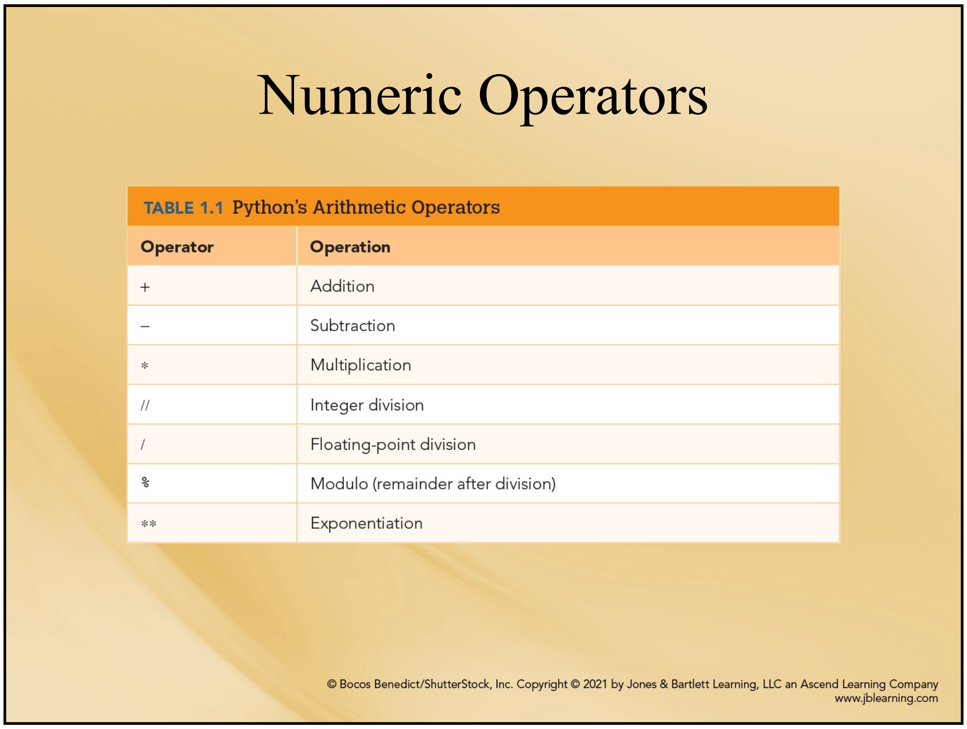
23



Calculation Using Variables

```
>>> pi = 3.14159
>>> radius = 8.0
>>> height = 16
>>> baseArea = pi * radius ** 2
>>> baseArea
201.06176
>>> cylinderVolume = baseArea * height
>>> cylinderVolume
3216.98816
```

24

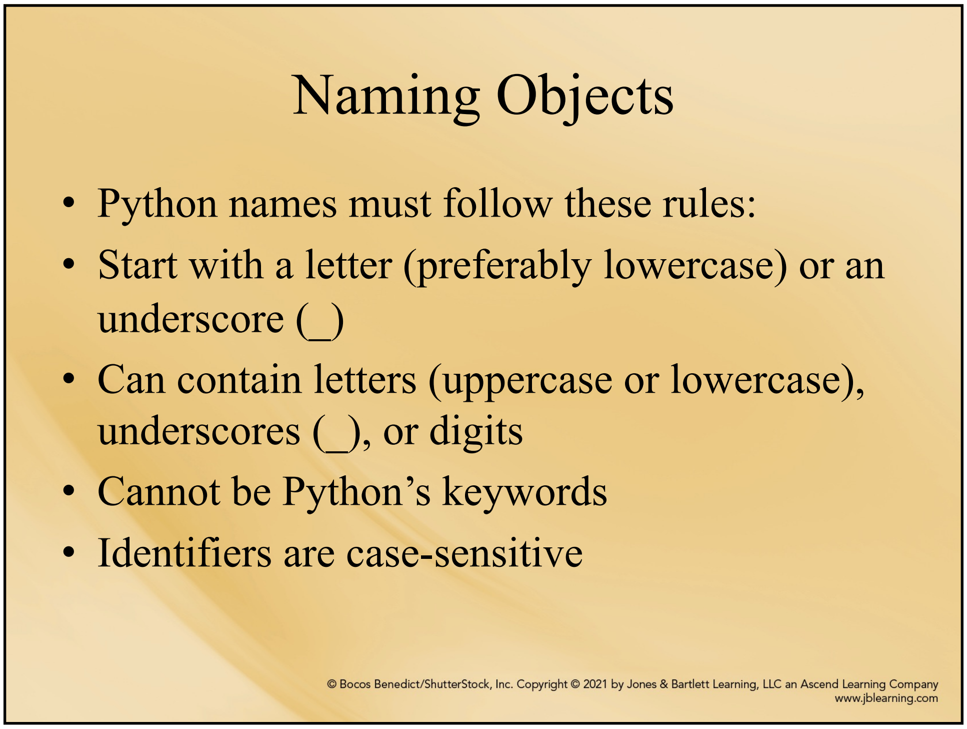


Numeric Operators

TABLE 1.1 Python's Arithmetic Operators

Operator	Operation
+	Addition
−	Subtraction
*	Multiplication
//	Integer division
/	Floating-point division
%	Modulo (remainder after division)
**	Exponentiation

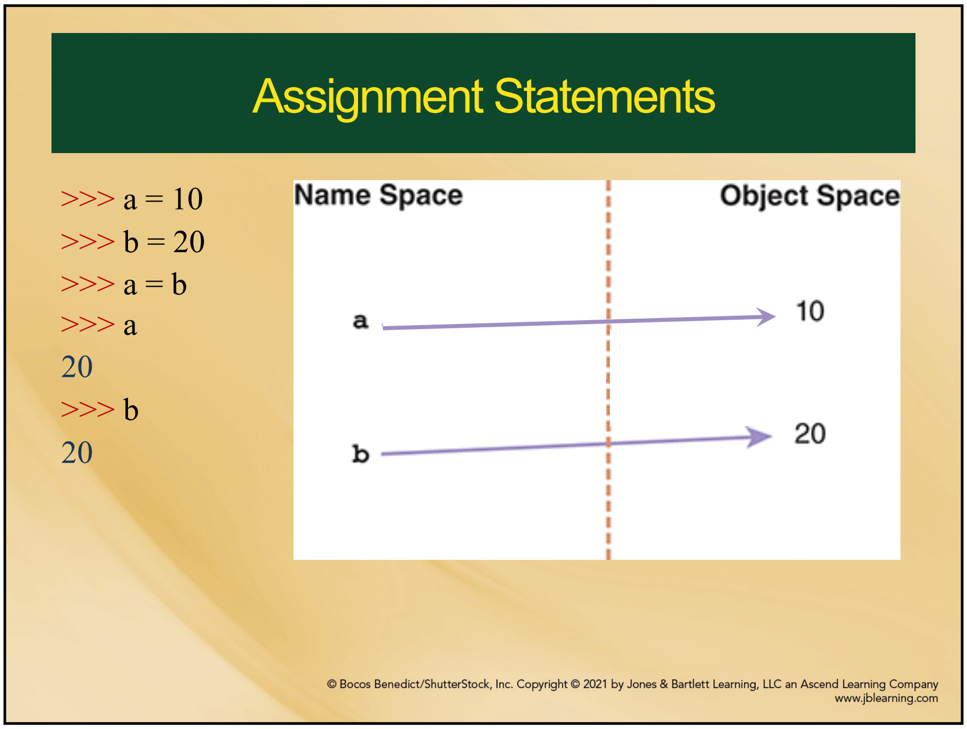
25



Naming Objects

- Python names must follow these rules:
- Start with a letter (preferably lowercase) or an underscore (_)
- Can contain letters (uppercase or lowercase), underscores (_), or digits
- Cannot be Python's keywords
- Identifiers are case-sensitive

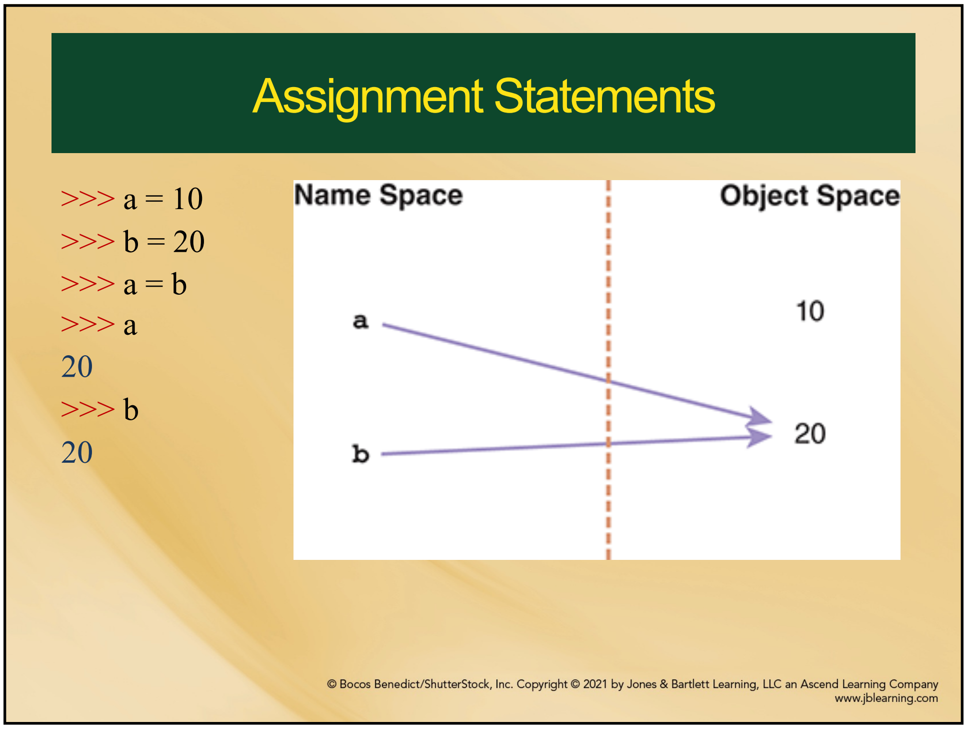
26



Assignment Statements

```
>>> a = 10
>>> b = 20
>>> a = b
>>> a
20
>>> b
20
```

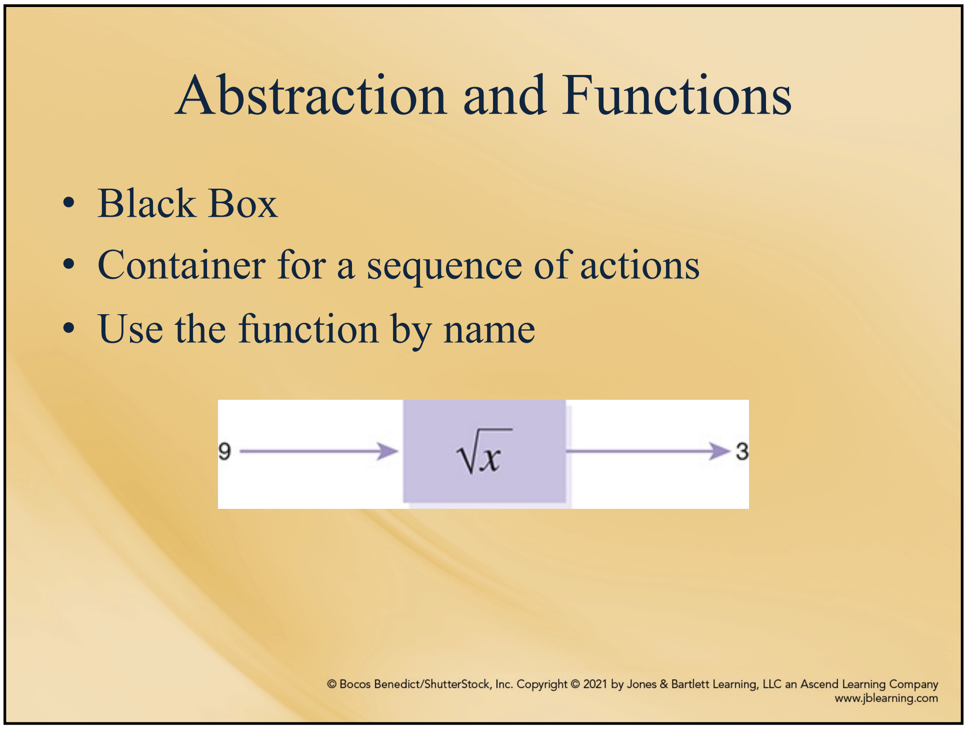
27



Assignment Statements

```
>>> a = 10
>>> b = 20
>>> a = b
>>> a
20
>>> b
20
```

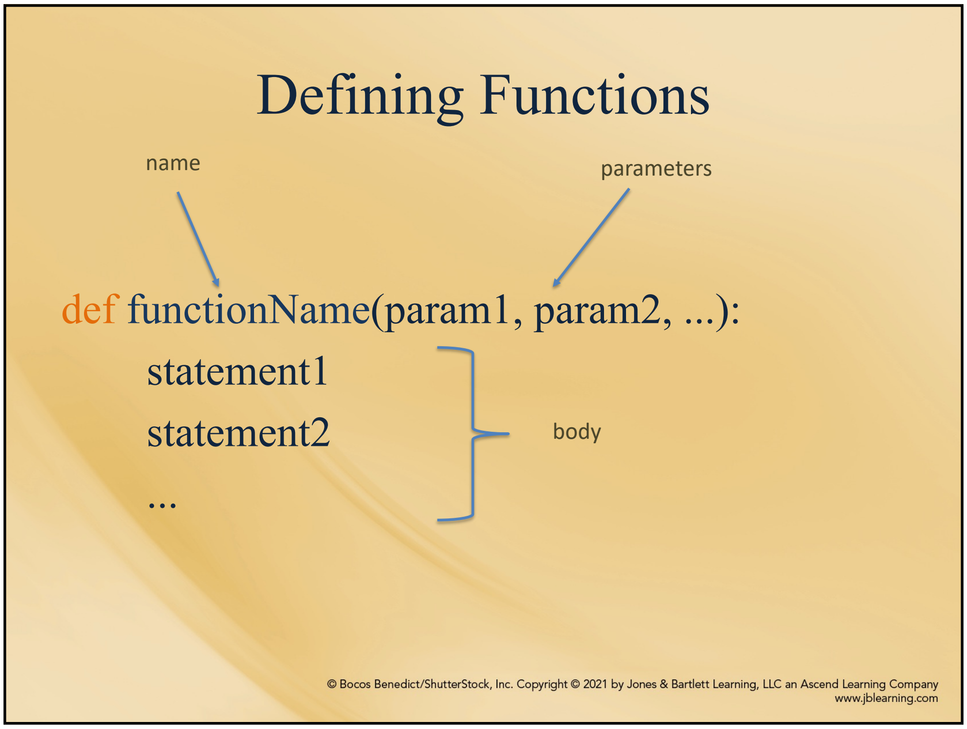
28



Abstraction and Functions

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

29



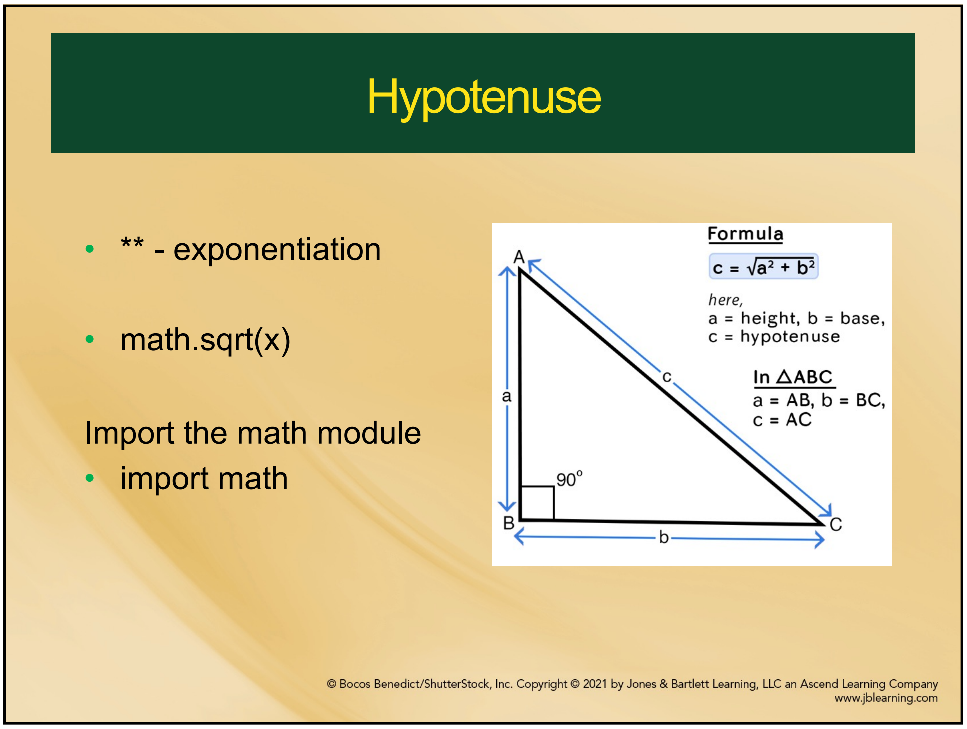
Defining Functions

name
parameters

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

body

30



Hypotenuse

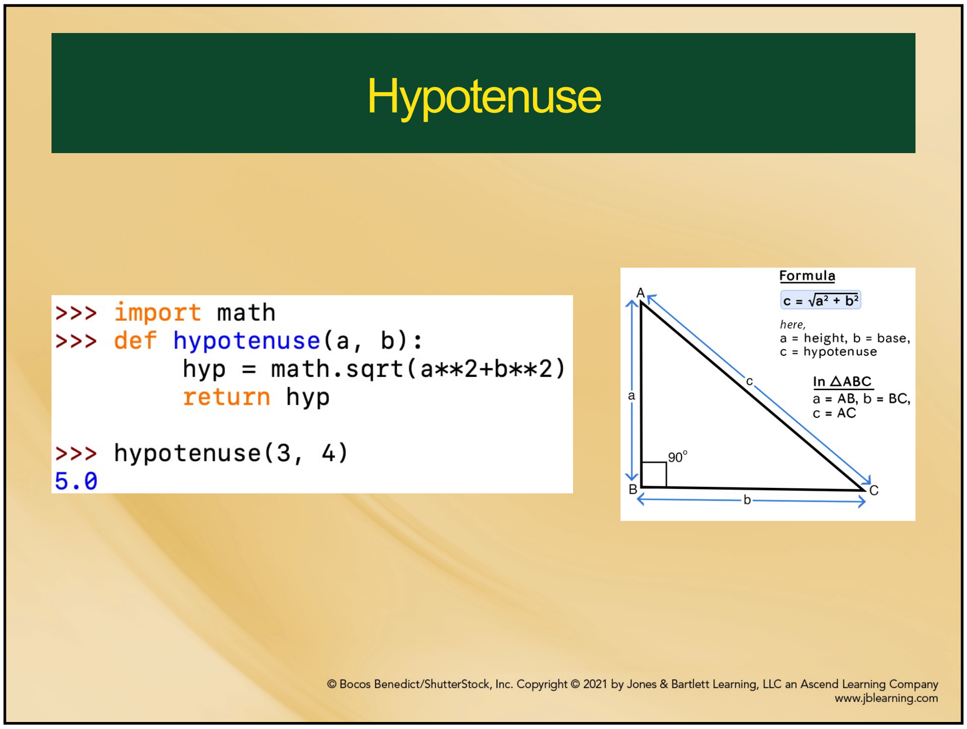
- ** - exponentiation
- math.sqrt(x)

Import the math module

- import math

Formula
c = $\sqrt{a^2 + b^2}$
here,
a = height, b = base,
c = hypotenuse
In △ABC
a = AB, b = BC,
c = AC
A
B
C
a
b
c
90°

31

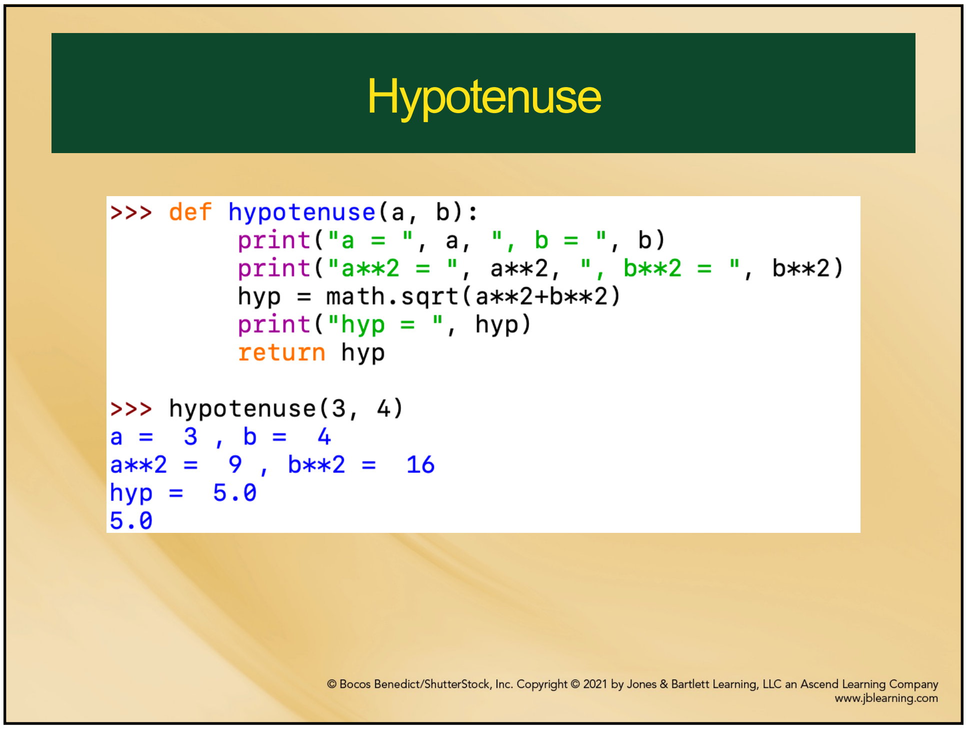


Hypotenuse

```
>>> import math
>>> def hypotenuse(a, b):
        hyp = math.sqrt(a**2+b**2)
        return hyp

>>> hypotenuse(3, 4)
5.0
```

Formula
c = $\sqrt{a^2 + b^2}$
here,
a = height, b = base,
c = hypotenuse
In △ABC
a = AB, b = BC,
c = AC
A
B
C
a
b
c
90°

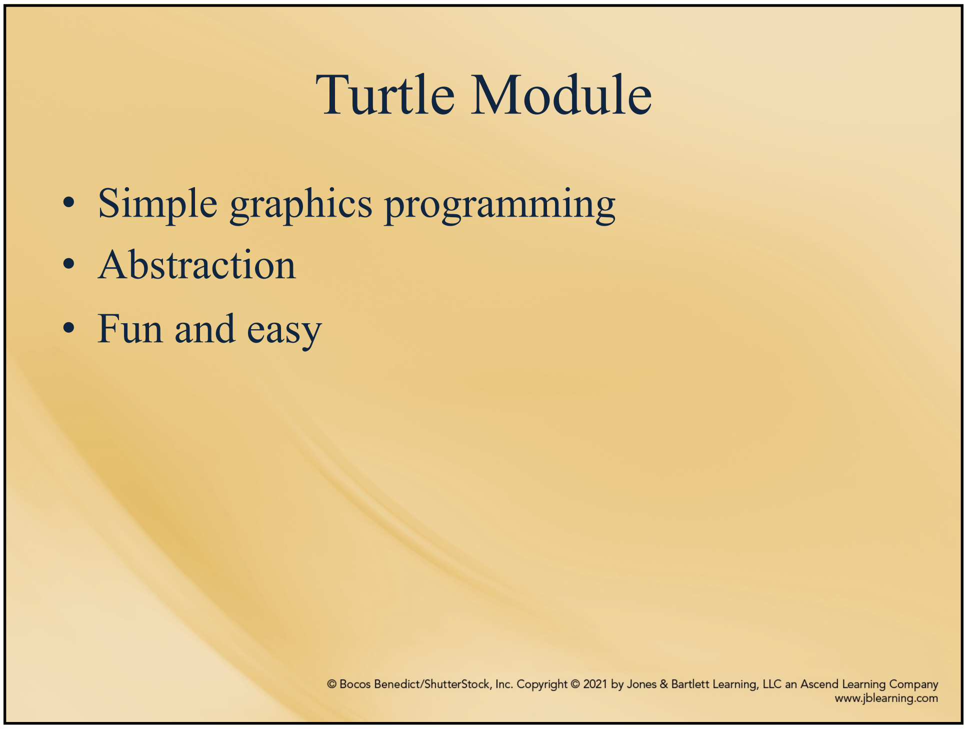
32



Hypotenuse

```
>>> def hypotenuse(a, b):
         print("a = ", a, ", b = ", b)
         print("a**2 = ", a**2, ", b**2 = ", b**2)
         hyp = math.sqrt(a**2+b**2)
         print("hyp = ", hyp)
         return hyp

>>> hypotenuse(3, 4)
a =  3 , b =  4
a**2 =  9 , b**2 =  16
hyp =  5.0
5.0
```

33



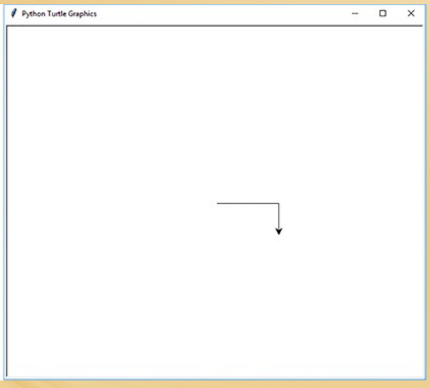
Turtle Module

- Simple graphics programming
- Abstraction
- Fun and easy

34

Using the Turtle

```
>>> import turtle
>>> t = turtle.Turtle()
>>> t.forward(100)
>>> t.right(90)
>>> t.forward(50)
>>> t.position()
(100.00,-50.00)
>>> t.heading()
270.0
```



35

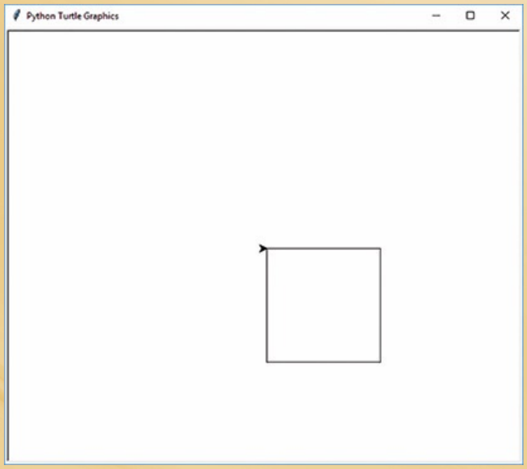
Drawing a Square

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

36

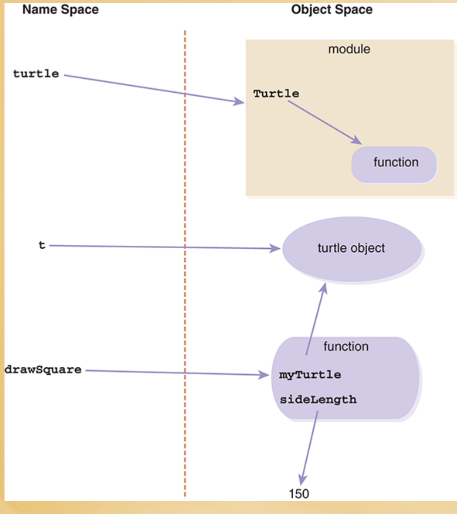
Calling the Function

```
>>> import turtle
>>> t = turtle.Turtle()
>>> drawSquare(t, 150)
>>> turtle. done()
```

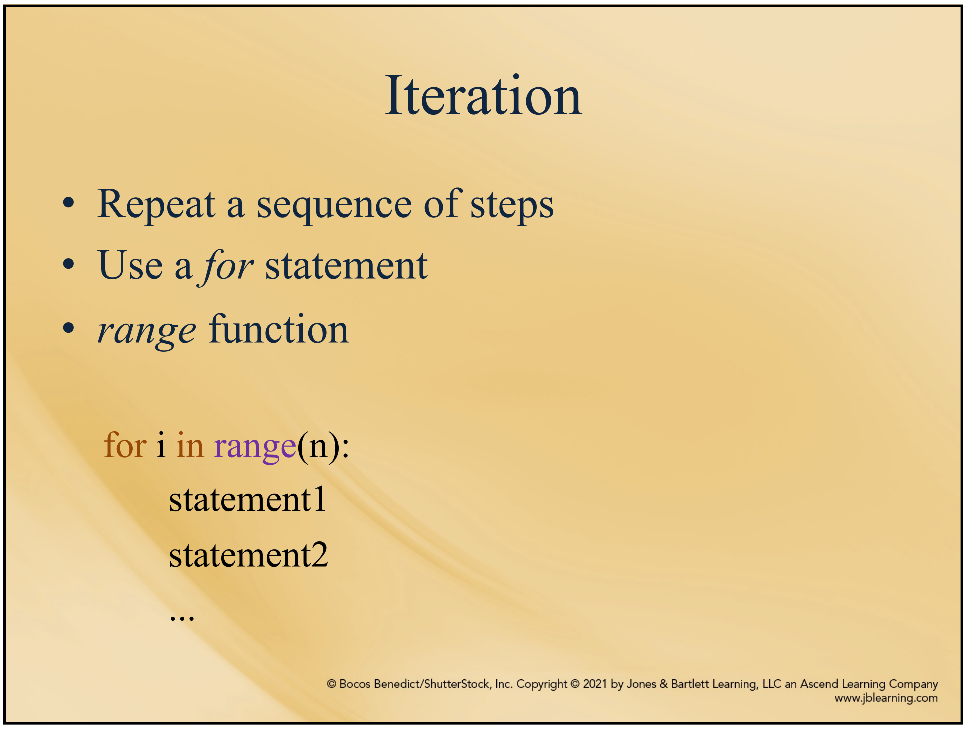


37

Reference Diagram



38

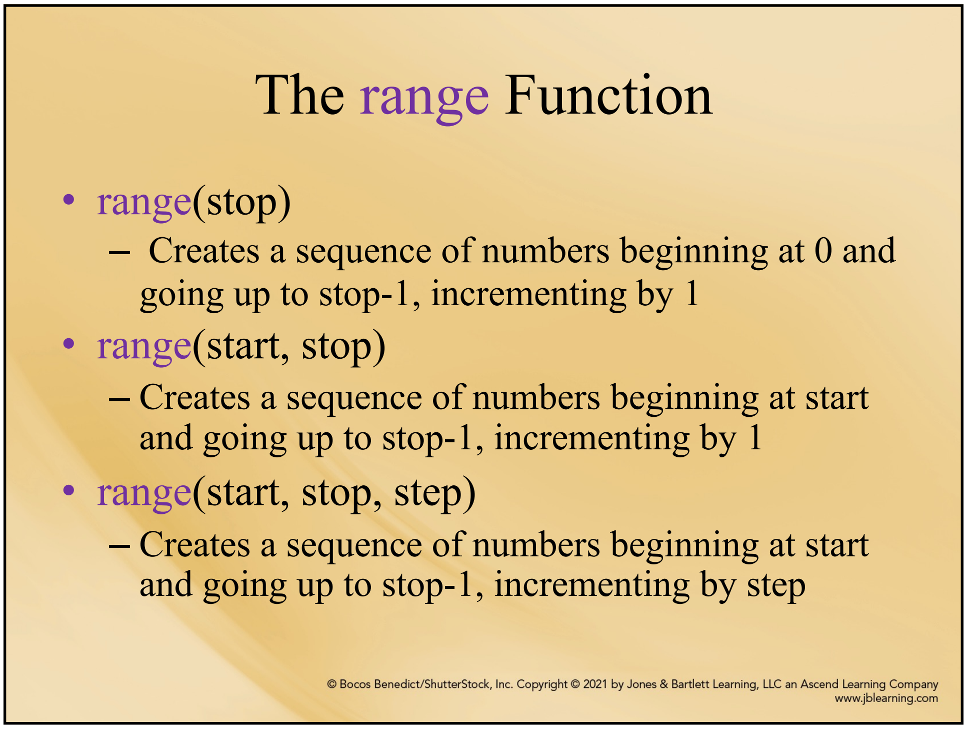


Iteration

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

```
for i in range(n):
    statement1
    statement2
    …
```

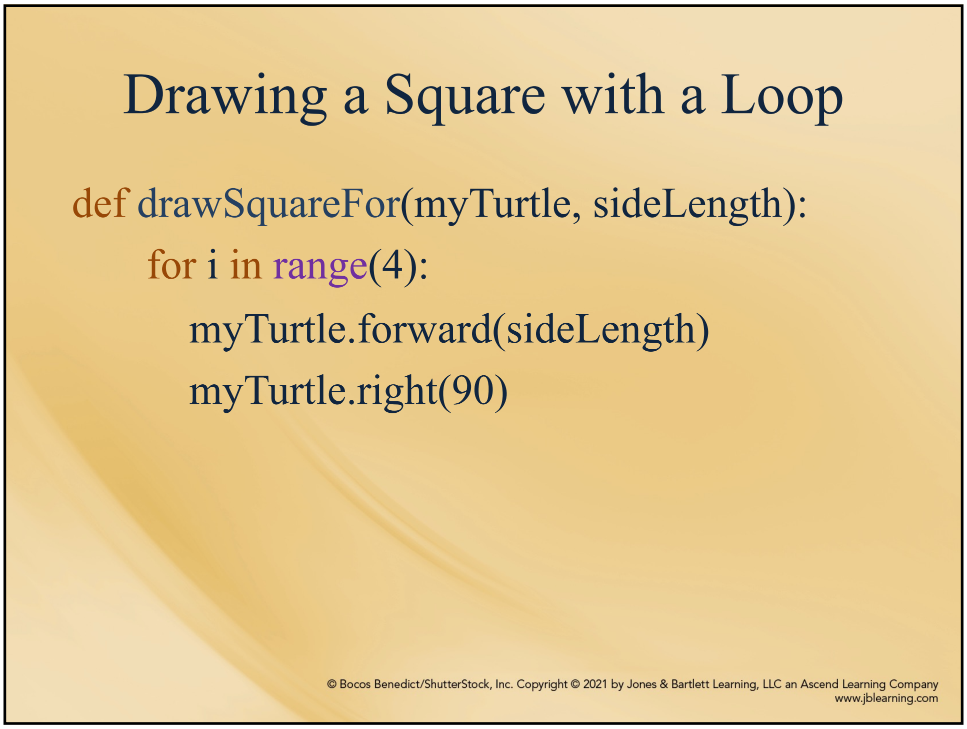
39



The range Function

- range(stop)
  - Creates a sequence of numbers beginning at 0 and going up to stop-1, incrementing by 1
- range(start, stop)
  - Creates a sequence of numbers beginning at start and going up to stop-1, incrementing by 1
- range(start, stop, step)
  - Creates a sequence of numbers beginning at start and going up to stop-1, incrementing by step

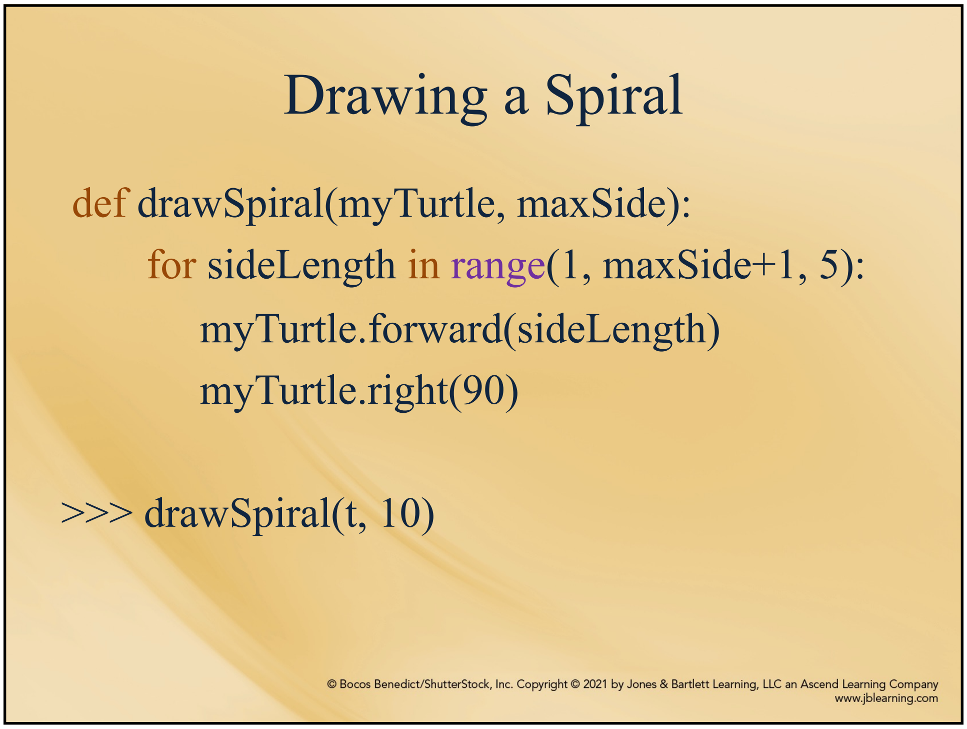
40



Drawing a Square with a Loop

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

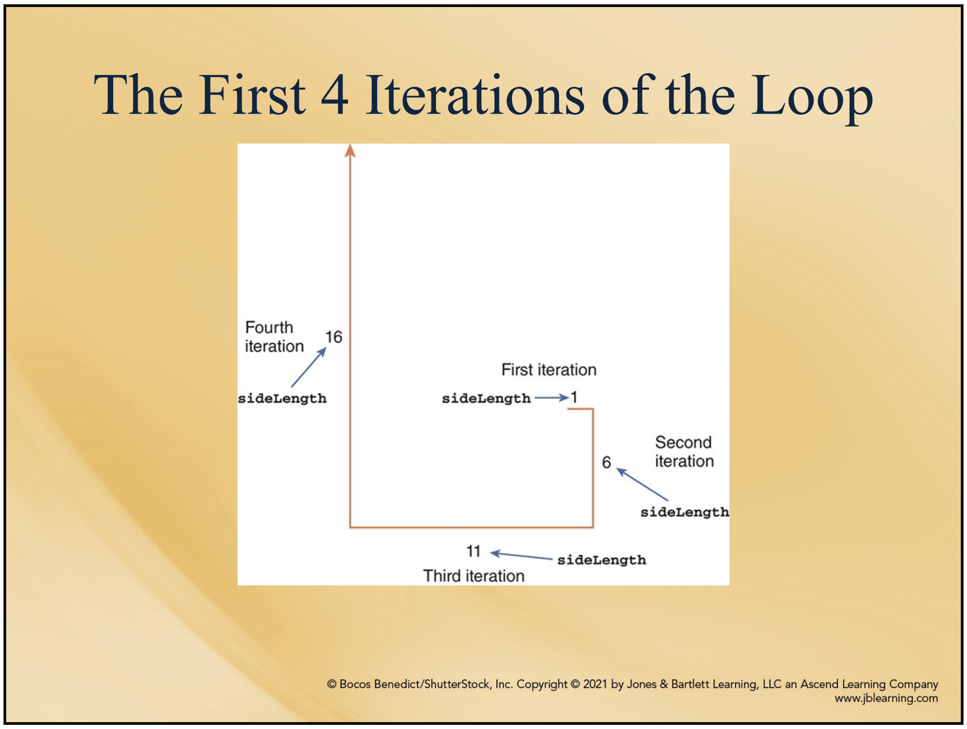
41



Drawing a Spiral

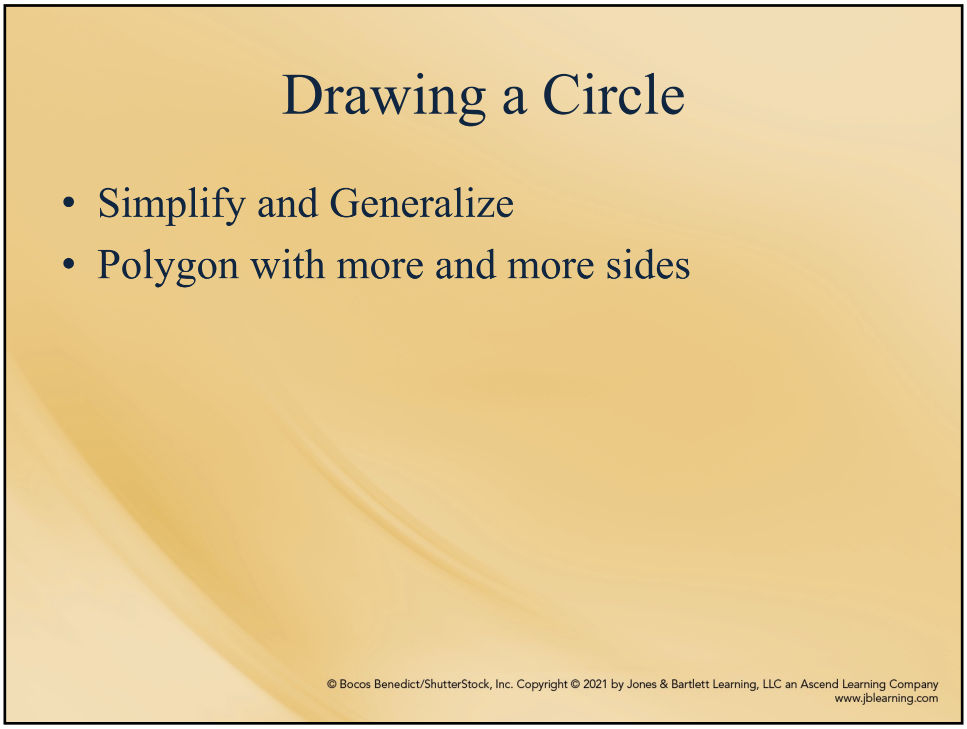
```
def drawSpiral(myTurtle, maxSide):
    for sideLength in range(1, maxSide+1, 5):
        myTurtle.forward(sideLength)
        myTurtle.right(90)

>>> drawSpiral(t, 10)
```

42



The First 4 Iterations of the Loop

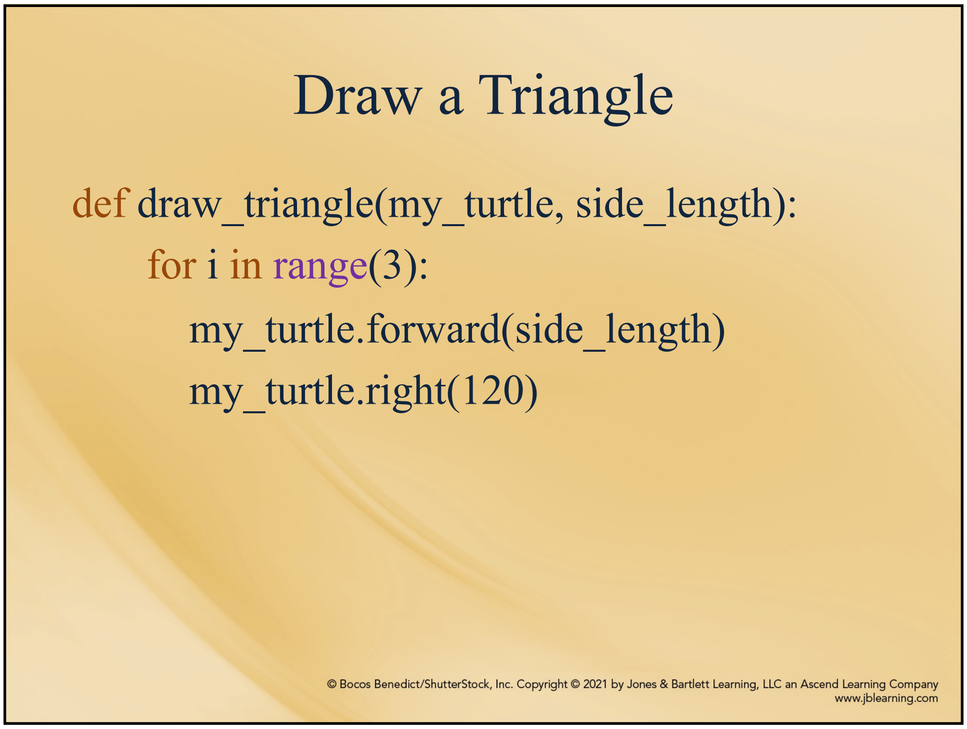
43



Drawing a Circle

- Simplify and Generalize
- Polygon with more and more sides

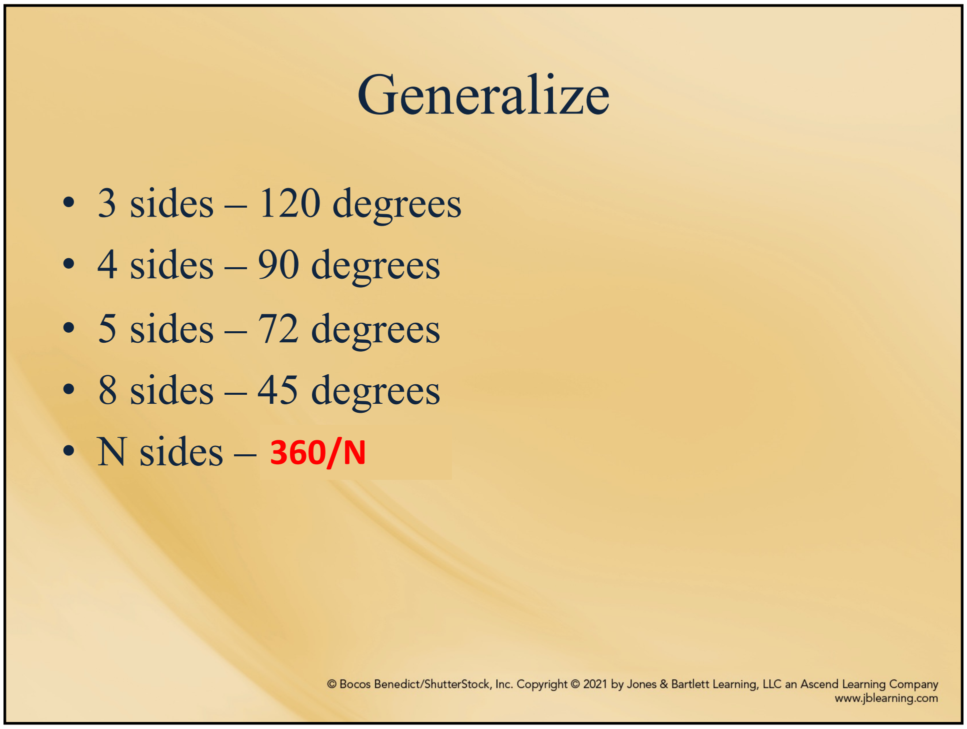
44



Draw a Triangle

```
def draw_triangle(my_turtle, side_length):
        for i in range(3):
            my_turtle.forward(side_length)
            my_turtle.right(120)
```

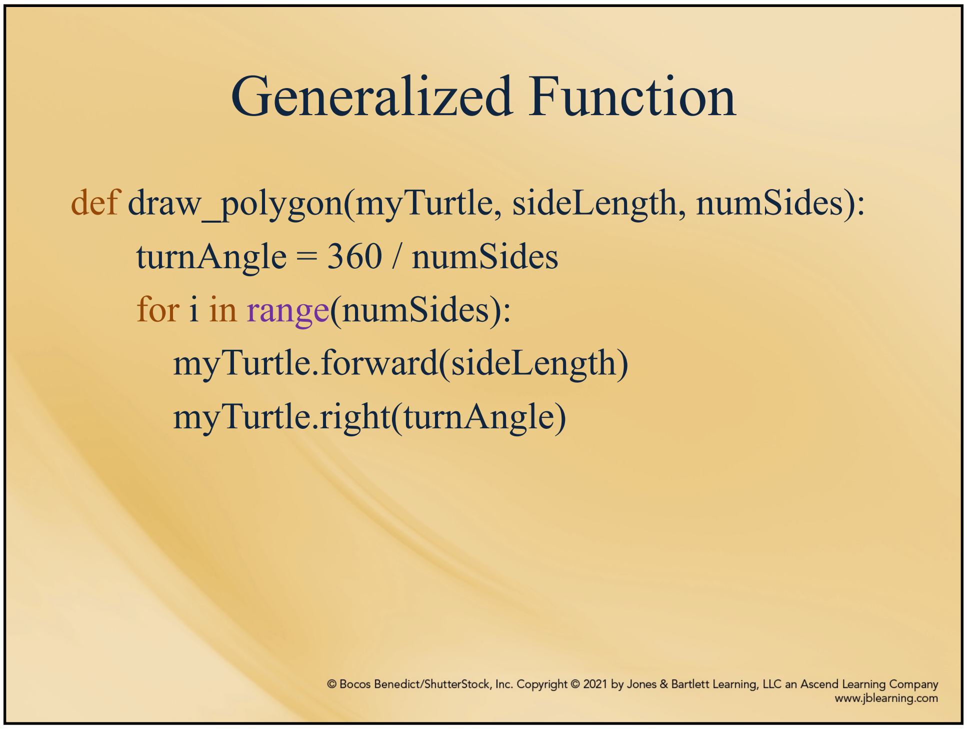
45



Generalize

- 3 sides – 120 degrees
- 4 sides – 90 degrees
- 5 sides – 72 degrees
- 8 sides – 45 degrees
- N sides – **360/N**

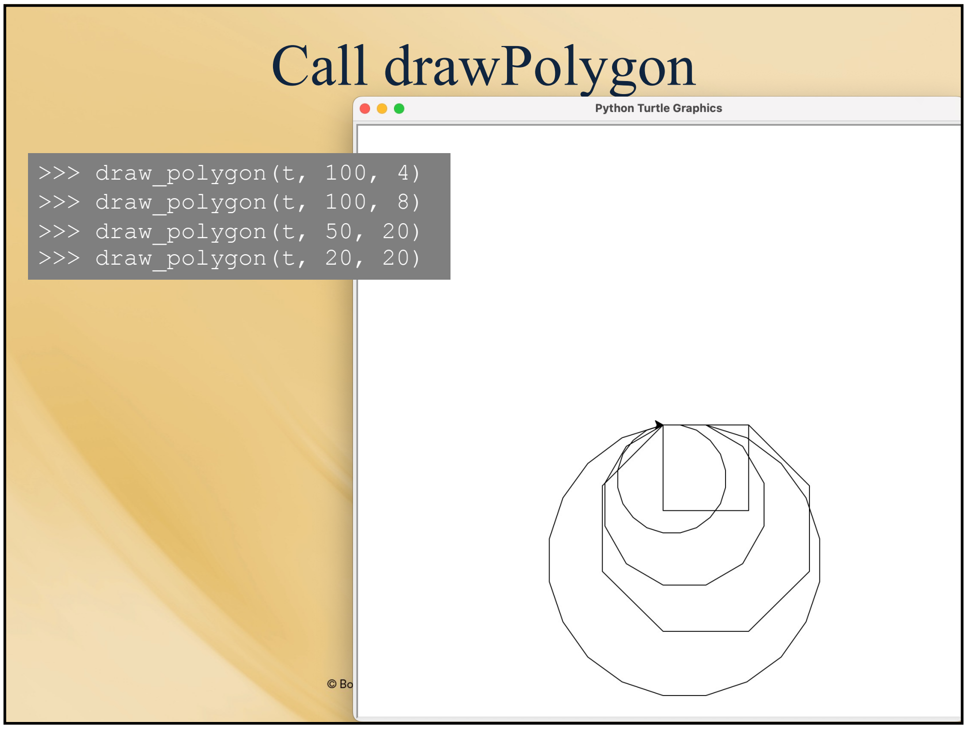
46



Generalized Function

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

47



Call drawPolygon

```
>>> draw_polygon(t, 100, 4)
>>> draw_polygon(t, 100, 8)
>>> draw_polygon(t, 50, 20)
>>> draw_polygon(t, 20, 20)
```

48

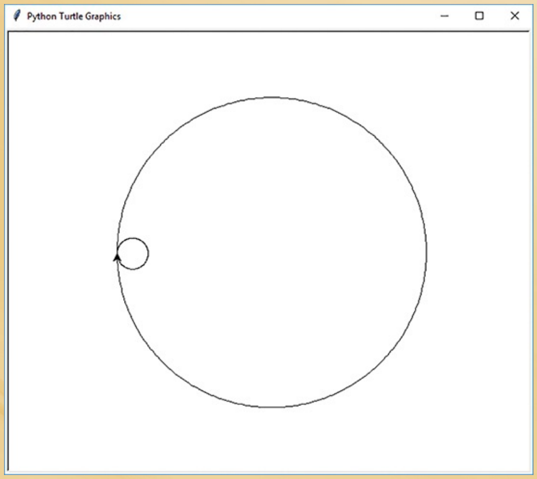
drawCircle uses drawPolygon

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

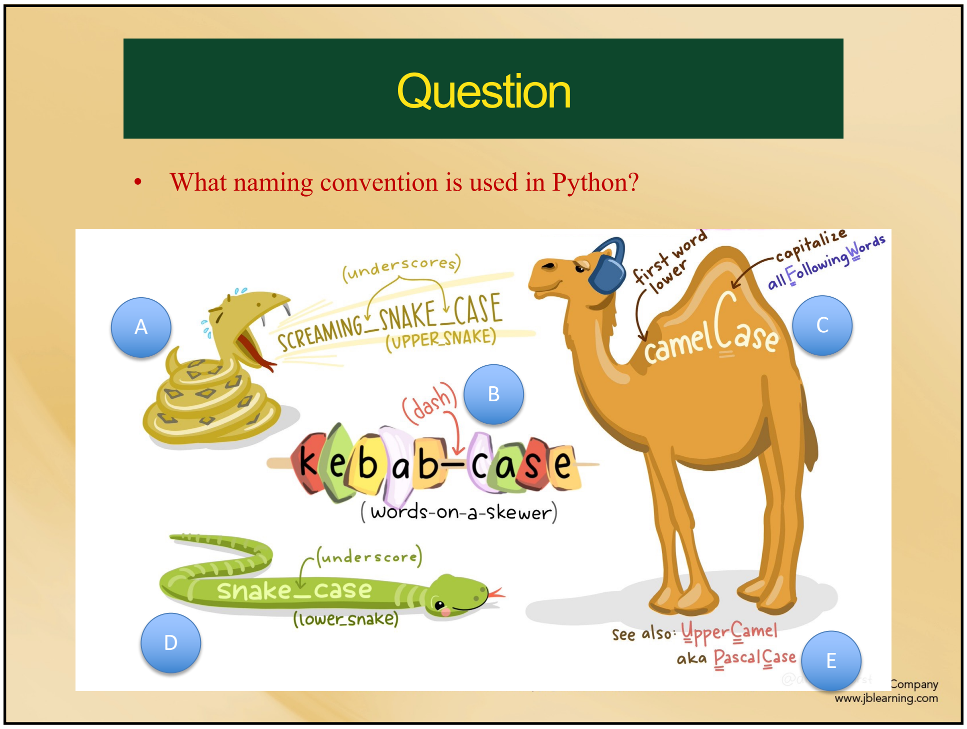
49

Draw Two Circles

```
>>> drawCircle(t, 20)
>>> drawCircle(t, 200)
```



50



Question

- What naming convention is used in Python?

Company
www.jblearning.com

51