Built-in power supply? Check. Ability to survive anything? Check. Easy to control? Okay, anyone who's had a cockroach as an uninvited houseguest knows that's not the case. So, rather than re-inventing the biological wheel with a robotic version, North Carolina State university researchers have figured out a way to remotely control a real Madagascar hissing cockroach. They used an off-the-shelf microcontroller to tap in to the roach's antennae and abdomen, then sent commands that fooled the insect into thinking danger was near, or that an object was blocking it. That let the scientists wirelessly prod the insect into action, then guide it precisely along a curved path, as shown in the video below the break. The addition of a sensor could allow the insects to one day perform tasks, liking searching for trapped disaster victims -- something to think about the next time you put a shoe to one.

http://www.engadget.com/2012/09/06/researchers-guide-cockroach-wirelessly/
Functions

def fact(n):
    prod = 1
    for factor in range(n, 1, -1):
        prod = prod * factor
    return prod

def main():
    target = eval(input("n! for what n?"))
    answer = fact(target)
    print( target, "! = ", answer)
def firstCalc(a, b):
    return a + b

def secondCalc(a, b, c):
    return (a + b - c)

def getInput():
    return (eval(input("number?")))

def main:
    a = getInput()
    b = getInput()
    c = getInput()
    d = firstCalc(a, b)
    e = secondCalc(a, c, d)
    print("The answer is: ", e)
• Write a function to print out 5 stars
• Write a main() to call that function

• Write a function to print out n stars
• Write a main() to call that function 5 times with n = 5

• Write a main() to call that function 5 times, where n goes from 1 to 5
Overview

• Each data type can represent certain set of values
• Each data type has a set of associated operations
  (e.g., + is different for float and for string)

• “Traditional programming view” is that data is passive – it’s manipulated and combined with active operations

• More modern: object-oriented approach
• Graphical User Interfaces (GUIs) provide windows, icons, etc.
  – Graphics library (graphics.py) written specifically for this book
  – Uses object-oriented programming (OOP) approach
The Object of Objects

• Basic idea – view a complex system as the interaction of simpler objects
• An object is a sort of active data type that combines data and operations
• Objects know stuff (contain data) and they can do stuff (have operations)
• Objects interact by sending each other messages
Objects Example

- Data processing system for the college
- Keep records on students who attend
- Each student can be represented as an object
The Object of Objects

• The student object would contain data like:
  – Name
  – ID number
  – Courses taken
  – Campus Address
  – Home Address
  – GPA
  – Etc.
The Object of Objects

• Student object should also respond to requests
• May want to send out a campus-wide mailing: need a campus address for each student
• We could send the `printCampusAddress` message to each student object. When the student object receives the message, *the object prints its own address*
Object of Objects

• Objects may reference other objects

• Each course might be represented by an object:
  – Student roster: keep track of student objects
  – Instructor: have a new object type of instructor
  – Where the class meets: string? New object type?
  – Etc.

• Course objects might be asked to
  – List all students in the class
  – List addresses of all students in class
  – Etc.
Discussion

• Goal: Enable an advisor to check if his student can take CM 151

• What data / methods should student object contain?
• What data / methods should a course object contain?
Simple Graphics Programming

• This chapter uses the `graphics.py` library

• Two location choices
  — In the same folder as your graphics program
    • For Linux or your own computer
  — In Python’s Lib directory with other libraries
    • See page 488 for your own computer
    • Can download from: mcsp.wartburg.edu/zelle/python
Simple Graphics Programming

• This is a library: need to import the graphics commands
  >>> import graphics

• *Graphics window* is a place on screen where graphics will appear
  >>> win = graphics.GraphWin()

• This command creates new window titled “Graphics Window”
Simple Graphics Programming

• *GraphWin* is an object
  – assigned to the variable *win*
  – can manipulate the window object through this variable

• Windows can be closed/destroyed by issuing the command

```python
>>> win.close()
```