Functions

Institute for Personal Robots in Education (IPRE)
A function is a piece of code you can use over and over again
  • Treat it like a black box
You pass it (optional) values, it does some work, and it (optionally) returns values
You “call it”, ”invoke it”, or “use it” by using its name and parentheses
  • The things you pass it go inside the parentheses
  • function( input ) => output
Using Simple Functions

- Functions that interact with the robot
  - forward (speed, duration)
  - beep(time, frequency)
- Pass in arguments
- Execute in sequential order
  - flow of execution

  forward(1, 1)
  beep(1, 440)
Writing Simple Functions

• **Defining** functions
  • Creates function
  • Does not execute/run them
• Indentation indicates a “block” of code
• Call functions from top-level or other functions

```python
def nudge():
    print "going forward"
    forward(1,1)
    print "now stopped"

nudge()
```

No Indention
“Top Level”
Format of a function definition

```python
def function-name():
    statement
    statement
    statement
    ...
    statement
```
def function-name(series, of, params):
    statement
    statement
    ...
    statement

function-name(series, of, params)
def nudge(speed):
    print "Going forward with speed", speed
    forward(speed,1)
    print "stopped"
    stop()

nudge(0.2)
nudge(0.9)
nudge(1)
Parameters are Variables

- When you pass values into functions as parameters, they get assigned to the variable names declared in the definition line of the function.

- For example, when you call `nudge(0.2)` The speed variable is assigned (points to) the value 0.2

- When the code in the function refers to the speed variable, it evaluates to the number 0.2

- So, when you call `nudge(0.2)` and the nudge function calls `forward(speed, 1)`, it's the same as if it called `forward(0.2, 1)`
Passing variables to a function

- If you pass a variable to a function, the function gets the value that the variable is pointing at.

```python
userInput = raw_input("Enter a Name")
setName(userInput)
print "The Robots new Name is: ", userInput
```
def beepA(length, octave):
    beep(length, 440 * (2**octave))

beepA(1,0)  # A4
beepA(2,1)  # A5
beepA(3,2)  # A6

A4 : 440 Hz
A5:  880 Hz
A6: 1760 Hz
A7: 3520 Hz
We've already seen examples:

```python
name = getName()
print "Hello, your robot is", name

print "Robot battery voltage", getBattery()

p = takePicture()
show(p)
```
Return Statements

• The return statement is used to return a value from a function

• The return statement also affects the flow of execution
  • Whenever the flow of execution hits a return statement it jumps back to the place where the function was called

• All functions have an implicit return statement at the end of the block of indented code, even if you do not specifically place one at the end of your function
def function-name(series, of, params):
    statement
    statement
    ...
    statement
    return value

output = function-name(series, of, params)
def area(radius):
    return 3.14 * radius**2

def circumference(diameter):
    return 3.14 * diameter

print "Area of a 3 ft circle", area(3)
print "Circumference", circumference(2*3)
Local Variables

- Local variables are variables which are defined inside a function.
- These variables exist only inside the function.
- We refer to this concept as the “scope” in which a variable is defined.
def area(radius):
    a = 3.14 * radius**2
    return a

def circumference(diameter):
    c = 3.14 * diameter
    return c

print “Area of a 3 ft circle”, area(3)
print “Circumference”, circumference(2*3)
Variables in a Function are Local

• Variables in a function are private
  – Including the parameters
  – Means other code can't "touch" them

• Each function has its own variables
  – Even when the names are the same

• Allows you to write functions independently without worrying about using the same name
def area(radius):
    a = 3.14 * radius**2
    return a

def circumference(radius):
    a = 3.14 * 2 * radius
    return a

a = 20
print "Area of a 3 ft circle", area(3)
print "Circumference", circumference(3)
print a
Composing/Composite Functions

• You can use the output (return value) of one function as the input (parameter) to another function.

\[\text{show( takePicture() )}\]

• In this example, the takePicture() function executes first (things inside parenthesis execute before things outside parenthesis)
• The takePicture() function returns a picture, which is then given to the show() function as a parameter.