

# Functions and Exam Review

---

Rose-Hulman Institute of Technology

Computer Science and Software Engineering

Check out 11-WhileLoops from SVN

# Function Review

---

- Functions can take multiple parameters
  - `def distance (p1, p2): # p1, p2 are points`  
`xdist = abs(p1.getX()- p2.getX())`  
`ydist = abs(p1.getY()- p2.getY())`  
`return math.sqrt(xdist*xdist + ydist*ydist)`
- Invoke a function; must supply actual parameters:
  - `d = distance(Point(-1,2), Point(2,6))`
- Functions can return values

# Passing parameters in Python

---

- What type of information do formal parameters receive?
- If we assign new values to formal parameters, does this affect the actual parameters?
- Consider this version of square:
  - **def squareNext(x):**  
    **x = x + 1**  
    **return x \* x**

See [mutatorExample.py](#)

# Passing a mutable parameter

- Function can change contents of a mutable parameter
- What does this print?  
What actually gets passed to the function?

```
def addOneToAll(listOfNums):  
    '''Adds one to each item in  
    the given list.'''  
    for i in range(len(listOfNums)):  
        listOfNums[i] += 1  
    listOfNums = [0, 1, 2]  
  
myList = [1,3,5,7]  
addOneToAll(myList)  
print("myList is ", myList)
```

See [mutatorExample.py](#)

# Optional parameters

---

- Function definition can make some parameters *optional*

```
>>> int("37")
37
>>> int("37", 10)
37
>>> int("37", 8) # specify base 8
31
```

# Optional parameters

---

- We can declare a parameter to be optional by supplying a default value.

```
def printDate(month, day, year="2011"):
    print("{0} {1}, {2}".format(month, day, year))
```

```
>>> printDate("January", 5, 2012)
```

```
January 5, 2012
```

```
>>> printDate("January", 5)
```

```
January 5, 2011
```

# Multiple optional parameters

---

- Can use *named* actual arguments:

```
def printDate(month="Jan", day="4", year="2011"):  
    print("{0} {1}, {2}".format(month, day, year))
```

```
printDate()
```

```
printDate(26)
```

```
printDate(day=26)
```

See [mutatorExample.py](#)

# Returning Multiple Values

---

- A function can return multiple values
  - `def powers(n):`  
    `return n**2, n**3, n**4`
- What's the type of the value returned by this call?  
**powers(4)**
- Assign returned values individually, or to a tuple:  
**thePowers = powers(5)**  
**p2, p3, p4 = powers(5)**

See `mutatorExample.py`

# Exam 1

---

- Format:
  - Paper Component
  - Computer Part

# Resources for Paper Part

---

- Textbook
- Notes sheet:
  - Single 8.5 by 11 page (both sides)
  - Whatever you want on it
  - Prepare this carefully!

No  
Sharing!

Q1

# Resources for Computer Part

---

- Any printed or handwritten material you choose (notes, books, printouts, ...)
- Your computer, with power adapter and network cable
  - Your computer and anything actually on it
  - Network, but only to access your own SVN repository and any material directly reachable from the CSSE 120 ANGEL and course web sites for this term

Q2

# Possible Topics for Exam 1

---

- Zelle chapters 1-7
- algorithm
- comment
- variable, assignment
- identifier, expression
- loop
  - definite (for)
  - counted (range function)
- phases of software development
- print, input
- import, math functions
- int, float, long, conversion
- strings (basic operations)
- character codes (chr, ord)
- lists (concatenation, slices)
  - list methods
  - indexing
- reading, writing files
- formatted output using format()
- using objects, graphics
- event-driven programming

Questions

?

# Review: Definite Loops

---

- *Definite loop*: knows before the loop starts to execute the number of iterations of the loop body

# Review: Definite loop over list

---

- `def printSquares(listOfNums):`
  - `for num in listOfNums:`
    - `val = num * num`
    - `print(val)`
- Useful when:
  - Only need to read from the list
  - Don't need the index (position) in the list

# Review: Definite, *counted* loop

---

- ```
def squareEach(listOfNums):  
    for i in range(len(listOfNums)):  
        val = listOfNums[i] * listOfNums[i]  
        listOfNums[i] = val
```
- Useful when:
  - Need to write to the list
  - Need the index (position) in the list for some other reason

# Is This Loop a Definite Loop?

---

```
# Open the file
```

```
inputFile = open(inputFileName, 'r')
```

```
# Process each line of file
```

```
for line in inputFile:
```

```
    image = Image(imageCenter, line.rstrip())
```

```
    image.draw(win)
```

```
    time.sleep(delay)
```

```
win.getMouse()
```

```
inputFile.close()
```

```
win.close()
```

# *Indefinite Loops*

---

- Number of iterations is not known when loop starts
- Is a conditional loop
  - Keeps iterating as long as a certain condition remains true
  - Conditions are Boolean expressions
- Typically implemented using *while statement*
- Syntax: **while <condition> :**  
**<body>**

# While Loop Debugging

---

- A *pre-test loop*: Condition tested at the top of the loop
- Example use of while loops:

*Nadia deposits \$100 in a savings account each month. Each month the account earns 0.25% interest on the previous balance. How many months will it take her to accumulate \$10,000?*

- What's wrong with `moneyDeposit.py` ?

Q5

# Infinite loops on purpose

---

- With **for** loops, we could make the program run for a really long time, but not forever.
- Create a very simple **while** loop that runs forever.

# Break statement

---

- Lets us break out of a loop in the middle
- Here's the pattern:
  - **while True:**
    - # Do some processing**
    - if processingSaysToStop:**
    - break**
    - # Do some more processing**

Start HW11  
when done

Complete TODOs in `guessMyNumber.py`

Demo your program when you finish

# Exercise: While Loops

Q6 - 7