

## Objectives

- Using Functions
- Import

Jan 20, 2012

Sprenkle - CSCI111

1

## Review

- How can I covert one primitive type to another primitive type?
- What is the short cut for adding 1 to a variable and saving it in that variable?
- What statement do we use to repeat something?
- What is the syntax of that statement?
- What does the following statement mean/do?  
`for i in range(5):`

Jan 20, 2012

Sprenkle - CSCI111

2

## Review

- Type conversion
  - Use type's *constructor*
- Shorthands, such as `x+=1`

Jan 20, 2012

Sprenkle - CSCI111

3

## for loop review

```
for i in range(5):  
    # like assigning i values(0,1,2,3,4)  
  
    # rest of loop body ...
```

- Note: when have `range(5)`
  - `i` gets values (0, 1, 2, 3, 4)
  - Which means that loop executes 5 times
- Optional: start and step parameters

Jan 20, 2012

Sprenkle - CSCI111

4

## Practicing for Loops

What is getting repeated?  
How many times?

- A) 1  
2  
3  
4  
Tell me that you  
love me more
- C) 10  
9  
8  
7  
...  
1  
Blast off
- B) I had the time of my life  
And I never felt this way before  
And I swear this is true  
And I owe it all to you
- } 3 times,  
followed by Dirty bit

Jan 20, 2012

Sprenkle - CSCI111

sum5.py

5

## Parts of an Algorithm

- Input, Output
- Primitive operations
  - What data you have, what you can do to the data
- Naming
  - Identify things we're using
- Sequence of operations
- Conditionals
  - Handle special cases
- Repetition/Loops
- Subroutines
  - Call, reuse similar techniques



Jan 20, 2012

Sprenkle - CSCI111

6

# FUNCTIONS

Jan 20, 2012

Sprenkle - CSCI111

7

## Motivating Functions

- PB&J: spreading PB, spreading jelly
  - Similar processes
  - Want to do many times
  - Simplify by saying "spread" rather than saying "move the knife back and forth, condiment side down, against the bread until you get X inches of ..."
- Benefits
  - Reuse, reduce code
  - Easier to read, write

Jan 20, 2012

Sprenkle - CSCI111

8

## Example

- How would you find the area of this shape?



Jan 20, 2012

Sprenkle - CSCI111

9

## Example

- How would you find the area of this shape?
- Algorithm Possibilities:
  - Total Area =  $\frac{1}{2} b_t h_t + w_r * h_r$
  - Total Area = Area of triangle + Area of rectangle

We already solved these

Which algorithm is easier to understand?



Jan 20, 2012

Sprenkle - CSCI111

10

## Functions

- Functions perform some task
  - May take **arguments/parameters**
  - May **return** a value that can be used in assignment



What does it do?  
How does it do it?

We don't know **how** it does it,  
but it's okay because it doesn't matter  
→ as long as it **works!**

Jan 20, 2012

Sprenkle - CSCI111

11

## Functions



Argument list (input)

- Syntax:
  - `func_name(arg0, arg1, ..., argn)`
- Depending on the function, arguments may or may not be required
  - `[]` indicate an optional argument
- Semantics: depend on the function

Jan 20, 2012

Sprenkle - CSCI111

12

## Built-in Functions

- Python provides some built-in functions for common tasks

Known as function's **signature**  
 Template for how to "call" function  
 Optional argument

- input([prompt])**
  - If prompt is given as an argument, prints the prompt without a newline/carriage return
  - If no prompt, just waits for user's input
  - Returns user's input (up to "enter") as a **string**

Jan 20, 2012

Sprenkle - CSCI111

13

## Description of print

- print([object, ...], \*, sep=' ', end='\n', file=sys.stdout)** Important later

Meaning: default values for sep and end are ' ' and '\n', respectively

- Print *object(s)* to the stream *file*, separated by *sep* and followed by *end*.
- Both *sep* and *end* must be strings; they can also be None, which means to use the default values. If no *object* is given, *print()* will just write *end*.

<http://docs.python.org/py3k/library/functions.html#print>

Jan 20, 2012

## Description of print

- print([object, ...], \*, sep=' ', end='\n', file=sys.stdout)** Important later

Meaning: default values for sep and end are ' ' and '\n', respectively

- Examples

```
print("Hi", "there", "class", sep=' ', ' )
print("Put on same", end='')
print("line")
```

Output: Hi, there, class  
Put on sameline

Jan 20, 2012

Sprenkle - CSCI111

15

## More Examples of Built-in Functions

Function Signature	Description
<b>round(x[,n])</b>	Return the float <i>x</i> rounded to <i>n</i> digits after the decimal point. If no <i>n</i> , round to nearest <i>int</i> .
<b>abs(x)</b>	Returns the absolute value of <i>x</i> .
<b>type(x)</b>	Return the type of <i>x</i> .
<b>pow(x, y)</b>	Returns <i>x</i> <sup><i>y</i></sup> .

Terminal

Jan 20, 2012

Sprenkle - CSCI111

16

## Using Functions

- Example use: Alternative to exponentiation
  - Objective: compute  $-3^2$
  - Python alternatives:
    - pow**(-3, 2)
    - $(-3) ** 2$
- We often use functions in assignment statements
  - Function does something
  - Save the output of function (what is **returned** in a variable)

```
roundx = round(x)
```

Jan 20, 2012

Sprenkle - CSCI111 [function\\_example.py](#)

17

## Python Libraries

- Beyond built-in functions, Python has a rich **library** of functions and definitions available
  - The library is broken into **modules**
  - A **module** is a file containing Python definitions and statements
- Example modules
  - math** — math functions
  - random** — functions for generating random numbers
  - os** — operating system functions
  - network** — networking functions

Jan 20, 2012

Sprenkle - CSCI111

18

## math Module

- Defines constants (variables) for `pi` (i.e.,  $\pi$ ) and `e`
  - These values never change, i.e., are **constants**
  - Recall: **we** name constants with all caps
- Defines functions such as

Function	What it Does
<code>ceil(x)</code>	Return the ceiling of x as a float
<code>exp(x)</code>	Return e raised to the power of x
<code>sqrt(x)</code>	Return the square root of x

Jan 20, 2012

Sprenkle - CSCI111

19

## Using Python Libraries

- To use the definitions in a module, you must first **import** the module
  - Example: to use the `math` module's definitions, use the import statement: `import math`
  - Typically import statements are at **top** of program
- To find out what a module contains, use the **help** function
  - Example within Python interpreter  
`import math`  
`help(math)`

Jan 20, 2012

Sprenkle - CSCI111

20

## Using Definitions from Modules

- Prepend constant or function with "**module**name."
  - Examples for constants:
    - `math.pi`
    - `math.e`
  - Examples for functions:
    - `math.sqrt`
- Practice
  - How would we write the expression  $e^{i\pi} + 1$  in Python?

Jan 20, 2012

Sprenkle - CSCI111 `module_example.py`

21

## Alternative Import Statements

```
from <module> import <defn_name>
```

- Examples:
  - `from math import pi`
    - Means "import pi from the math module"
  - `from math import *`
    - Means "import *everything* from the math module"
- With this **import** statement, don't need to prepend module name before using functions
  - Example: `e**(1j*pi) + 1`

Jan 20, 2012

Sprenkle - CSCI111

22

## Benefits of Using Python Libraries/Modules

- Don't need to rewrite code
- If it's in a module, it is very **efficient** (in terms of computation speed and memory usage)

Jan 20, 2012

Sprenkle - CSCI111

23

## Finding Modules To Use

- How do I know if functionality that I want already exists?
  - Python Library Reference:  
<http://docs.python.org/py3k/library/>
- For the most part, in the beginning you will write most of your code from scratch

Jan 20, 2012

Sprenkle - CSCI111

24

## RANDOM MODULE

Jan 20, 2012

Sprenkle - CSCI111

25

## random module

- Python provides the **random** module to generate pseudo-random numbers
- Why “pseudo-random”?
  - Generates a list of random numbers and grabs the next one off the list
  - A “seed” is used to initialize the random number generator, which decides which list to use
    - By default, the current time is used as the seed

Jan 20, 2012

Sprenkle - CSCI111

26

## Some random Functions

- **random()**
  - Returns the next random floating point number in the range [0.0, 1.0)
- **randint(a, b)**
  - Return a random integer N such that  $a \leq N \leq b$

```
import random

#random.seed(1)      # module.function()

for x in range(10):
    print(random.random())
```

Jan 20, 2012

Sprenkle - CSCI111

random\_test.py 27

## VA Lottery: Pick 4

- To play: pick 4 numbers between 0 and 9
- To win: select the numbers that are selected by the magic ping-pong ball machine
- Your job: Simulate the magic ping-pong ball machines
  - Display the number on one line

Jan 20, 2012

Sprenkle - CSCI111

pick4.py

28

## Programming Building Blocks

- Each type of statement is a building block
  - Initialization/Assignment
    - So far: Arithmetic, functions
  - Print
  - For
  - Import
- We can combine them to create more complex programs
  - Solutions to problems

Jan 20, 2012

Sprenkle - CSCI111

29

## Four Puzzles in Cyberspace

- Context: Book *Code* v2 by Lawrence Lessig
- You read Chapter 2
  - Presents the problems, not the author's proposed solutions

Cory  
Kari  
Sam  
Jonathan  
Gaurav

Gabrielle  
John G  
Josh  
Shannon  
Trang

Will  
Colby  
Koven  
Haley

Mary  
Hang  
Deirdre  
Connor

John K  
Luke  
Emily  
Drew

Jan 20, 2012

Sprenkle - CSCI111

30

## Four Puzzles from Cyberspace

- What are the four puzzles of cyberspace?
  - Favorite puzzle? Most interesting?
  - The most important to solve?
  - The hardest to solve?
- Do you consider Facebook to be “cyberspace”? If so, why? If not, how close is it?
- W&L administrators tried to regulate students use of juicycampus and acb
  - Why did they feel it should be regulated?
  - What types of regulation did they do? Was it effective?

Jan 20, 2012

Sprenkle - CSCI111

31