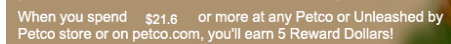


Objectives

- Designing for Change
- Using Functions
- Import

I see bugs and show them to you:



When you spend \$21.6 or more at any Petco or Unleashed by Petco store or on petco.com, you'll earn 5 Reward Dollars!

We'll learn how to fix this in Python next week

Jan 25, 2017

Sprenkle - CSCI111

1

Lab Review

- New tips, tricks, suggestions?

Jan 25, 2017

Sprenkle - CSCI111

2

DESIGNING FOR CHANGE

Jan 25, 2017

Sprenkle - CSCI111

3

Designing for Change

- What are we likely to change in the program?
- How can we make the program easier to change?

Jan 25, 2017

Sprenkle - CSCI111

4

Constants

- Special variables whose values are defined once and never changed
 - By convention, not enforced by interpreter
- By convention
 - A constant's name is all caps
 - Typically defined at top of program → easy to find, change
- Examples:

```
NUM_INPUTS = 5  
MIN_VALUE = 0
```

Never assigned values in remainder of program

Jan 25, 2017

Sprenkle - CSCI111

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 25, 2017

Sprenkle - CSCI111

6

FUNCTIONS

Jan 25, 2017

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

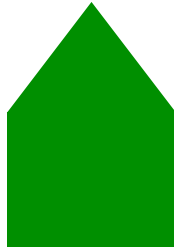
Jan 25, 2017

Sprenkle - CSCI111

8

Example

- How would you find the area of this shape?



Jan 25, 2017

Sprenkle - CSCI111

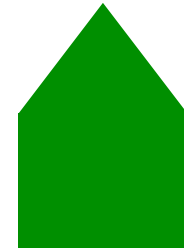
9

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

Which algorithm is easier to understand?

For (most) humans,
words and abstraction of ideas
are easier to understand



Jan 25, 2017

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 25, 2017

Sprenkle - CSCI111

11

Functions



- Syntax:
 - $\text{func_name}(\text{arg}_0, \text{arg}_1, \dots, \text{arg}_n)$
- Depending on the function, arguments may or may not be required
 - `[]` indicate an optional argument
- Semantics: depend on the function

Jan 25, 2017

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 25, 2017

Sprenkle - CSC1111

13

Description of print

- `print(value, ..., 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 25, 2017

Description of print

- `print(value, ..., 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 25, 2017

Sprenkle - CSC1111

`print_examples.py` 15

More Examples of Built-in Functions

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

Jan 25, 2017

Sprenkle - CSC1111

Interpreter

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)

```
roundedX = round(x)
```

Jan 25, 2017

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 25, 2017

Sprenkle - CSCI111

18

math Module

- Defines constants (variables) for `pi` (i.e., π) 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 25, 2017

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 25, 2017

Sprenkle - CSCI111

20

Using Definitions from Modules

- Prepend constant or function with **modulename**.
 - 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 25, 2017

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 25, 2017

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 25, 2017

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 25, 2017

Sprenkle - CSCI111

24

RANDOM MODULE

Jan 25, 2017

Sprenkle - CSC1111

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 25, 2017

Sprenkle - CSC1111

26

List of Lists of Random Numbers

Seed	List of Random Numbers				
1	0.1343642441	0.8474337369	0.763774619	0.2550690257	...
2	0.9560342719	0.9478274871	0.0565513677	0.0848719952	...
3	0.2379646271	0.5442292253	0.3699551665	0.6039200386	...
4	0.2360480897	0.1031660342	0.3960582426	0.1549722708	...
...

Jan 25, 2017

Sprenkle - CSC1111

27

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 25, 2017

Sprenkle - CSC1111

random_test.py 28

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 25, 2017

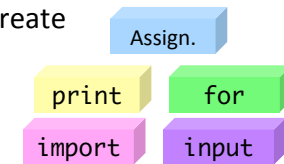
Sprenkle - CSCI111

`pick4.py`

29

Programming Building Blocks

- Adding to your tool set
- We can combine them to create more complex programs
 - Solutions to problems



Jan 25, 2017

Sprenkle - CSCI111

30

Looking Ahead

- Lab 2 due Friday
- Broader Issue: Google Search

Jan 25, 2017

Sprenkle - CSCI111

31