Objective

- More `for` loop
- Using Functions
- Broader Issue: Algorithm Bias

1

Review

- Which lab did you submit today?
  - How many have you completed?
- What statement do we use to repeat something?
- What are the possible ways to use the `range` function?
  - What do they mean?
- When we suspect we need a loop to solve a problem, what questions should we ask?
  - How do the answers to those questions inform our solution to a loop problem?
- What design pattern did the adding 5 numbers follow?
  - What are the steps of the pattern?

2

Practicing for Loops

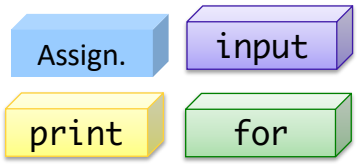
What is getting repeated? How many times?

- 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

Programming Building Blocks

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



Discussion: Programming Practice

- Problem: Add 5 numbers, inputted by the user
- We could have implemented this program last week
  - 5 separate input statements, add up the numbers
- Consider how much easier this program is to change if we want a different number of numbers added up

Review: Accumulator Design Pattern

1. Initialize accumulator variable
2. Loop until done
   - Update the value of the accumulator
3. Display result

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

7

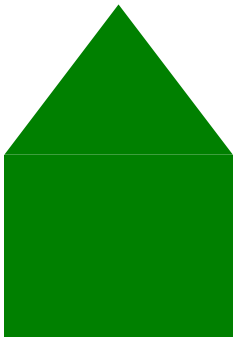
Motivating Functions

- PB&J: spreading PB, spreading jelly
  - Similar processes
  - Want to do many times
  - Rather than saying "move the knife back and forth, condiment side down, against the bread until you get X inches of …", say "spread"
- Benefits
  - Reuse, reduce code
  - Breaks problems into more manageable pieces
  - Easier to read, write

8

Example

- How would you find the area of this shape?



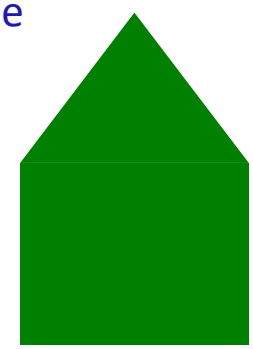
9

Example

- How would you find the area of this shape?
- Algorithm Possibilities:
  - Total Area = ½ b_t h_t + w_r*h_r
  - Total Area = Area of triangle + Area of rectangle

Which algorithm is easier to understand?

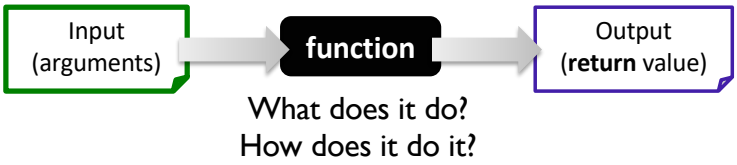
For (most) humans, words and abstraction of ideas are easier to understand



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**!

Functions

- Syntax:
  - `func_name(arg0, arg1, …, argn)`

Argument list (input)

- Depending on the function, arguments may or may not be required
  - [ ] indicate an optional argument
- Semantics: depend on the function

Built-in Functions

- Python provides some built-in functions for common tasks

  Known as function's ***signature***; *a* 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**

Description of `print`

- `print(*objects, sep=' ', end='\n', file=sys.stdout)`

  Semantics: default values for `sep` is `' '` and `end` is `'\n'`

  - 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*.

  `https://docs.python.org/3/library/functions.html#print`

Description of `print`

- `print(*objects, sep=' ', end='\n', file=sys.stdout)`

Semantics: default values for `sep` is `' '` and `end` is `'\n'`

- Examples:

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

Output:

```
Hi; there; class
Put on sameline
```


More Examples of Built-in Functions

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

Interpreter

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 (i.e., what is ***returned*** in a variable)

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


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

`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
`ceil(x)`	Return the ceiling of x as a float
`exp(x)`	Return e raised to the power of x
`sqrt(x)`	Return the square root of x

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)
```

Another Import Statement

```
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`

module_example_from_import.py

Using Definitions from Modules

```
import <module>
```

- Prepend constant or function with **`modulename`**.
  - Examples for constants:
    - `math.pi`
    - `math.e`
  - Examples for functions:
    - `math.sqrt(num)`

module_example_import.py

Comparing Import Statements

`import <module>`

- Requires prepending constants/functions with module
  - Ex: `math.sqrt(num)`
- Benefits:
  - Helps you to know which module the constant/function is coming from
  - No problem with name clashes if two modules define the same function
    - math.aFunction()
    - os.aFunction()

`from <module> import <defn_name>`

- Don't need to prepend constants/functions with module
  - Ex: `sqrt(num)`
- Benefit: Easier to write/read

Benefits of Using Python Libraries/Modules

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

Finding Modules To Use

- How do I know if functionality that I want already exists?
  - Python Library Reference: https://docs.python.org/3/library
- In the beginning, you will probably rewrite existing functionality to help you learn how it works

Broader Issue Groups

Alicia Amanda Charlie Matt Tim	Brian Claire Elizabeth Ethan Justin	Harrison Libby Michelle Sam Tyler	Elias Kyle Sambridhi Winter	David Jackson Micah Ricardo

Broader Issue: Human Bias in Algorithms

- People use the term "algorithm" to refer to different things
  - Distinguish those things
- Comment on this statement, in context of CSCI111: "Algorithms are opinions embedded in code."
- Reflect on "My department of education contact told me 'It's math and I wouldn't understand it.'"
  - Why is it beneficial to make the algorithm transparent? To keep it opaque?
- Consider the sentencing algorithm that considered likelihood of recidivism
  - What should be considered in sentencing?
  - How do we/should we "interrogate" algorithms?
- What algorithm are you questioning now?

Broader Issue: Human Bias in Algorithms

- Our definition of algorithms and the types of problems we solve are different than the ones described in the talk
  - Those algorithms: machine learning
    - Learn from data to categorize it or make predictions
  - Ours are likely not opinions
- BUT, you're learning more about programming and algorithms and it's a good idea to stop and question algorithms and results
  - You'll be a purchaser of software and I want you to be informed and ask good questions when making decisions
  - Yet another benefit of the liberal arts

Looking Ahead

- Pre Lab 3, Lab 3 next week