Objective

- More Functions
- `random` Module
- Animation

Review

- What are some variations in how we use the `print` function?
- Synthesis: What are benefits of functions?
- What is a *module*?
  - What are some available Python modules? What functionality do they have?
  - How can we find out what functionality is in a module?
- How can we access the functionality defined in the modules (two ways)?
  - How does that choice affect how we use the functionality in our code?

Review: Using `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
```

`print_examples.py`

Synthesis: Benefits of Functions

- Allows us to reuse, reduce code
  - Don’t need to rewrite code that has already been defined
- Abstraction: we can call the functions and don’t need to worry about how they work
- Breaks problems into more manageable pieces
- Using functions from Python modules: know they work and are efficient

Review: Comparing Import Statements

`import <module>`

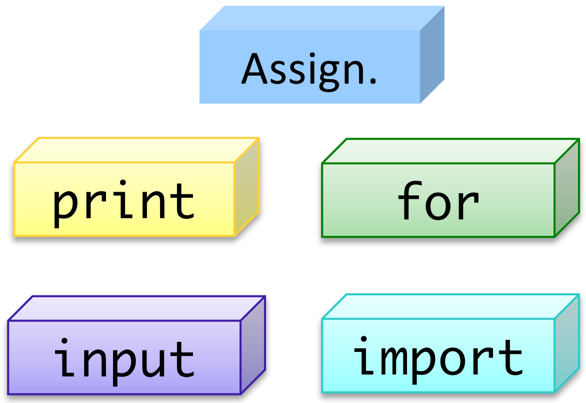
- Requires prepending constants/functions with module
  - Ex: `math.sqrt(num)`
- Benefits:
  - Helps you to know which module the constant/function is coming from
  - No problem with name clashes if two modules define the same function
    - math.aFunction()
    - os.aFunction()

`from <module> import <defn_name>`

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

Programming Building Blocks

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



Review: `random` module

- Python provides the `random` module to generate pseudo-random numbers
- What is "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

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 ≤ N ≤ b

```python
import random

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

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

random_test.py

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

`pick4.py`

Moving a Circle According to the User

- Draw a circle in the upper left-hand corner of the screen
- Tell the user to click somewhere
- Move the circle to where the user clicked

Hmm…. Some of these steps seem very different from what we've been doing.
Can we even do them?
How can we figure out if we can?

`circleShift.py`

ANIMATION

Animation

- Use combinations of the method `move` and the function `sleep`
  - Need to **sleep** so that humans can see the graphics moving
  - Otherwise, computer processes the **move**s too fast!
- `sleep` is part of the `time` module
  - takes a float representing *seconds* and pauses for that amount of time, e.g., `sleep(.5)` sleeps for .5 seconds

`animate.py`

Animate Circle Shift!

- Animate moving a circle to the position clicked by the user
  - Previously, moved in one fell swoop

```
dx = newX - circle.getCenter().getX()
dy = newY - circle.getCenter().getY()

circle.move(dx, dy)
```

  - To animate
    - Break the movement into chunks
    - Repeatedly, move one chunk, sleep
- Consider: how would you break the movement into 2 steps? 3 steps? Then, generalize

`circleShiftAnim.py`

Examples of Animation

- From Previous Classes

Looking Ahead

- Pre Lab 3 due before lab