

# **Intro to Programming (No Prior Experience)**

## **Class 14 – Functions Cont'd + Midterm Review**

Emily Zhao

T/R 4:55PM–6:10PM

# Agenda

- Functions (continued)
- Midterm Survey
- Midterm Review Game

# Functions pt. 2

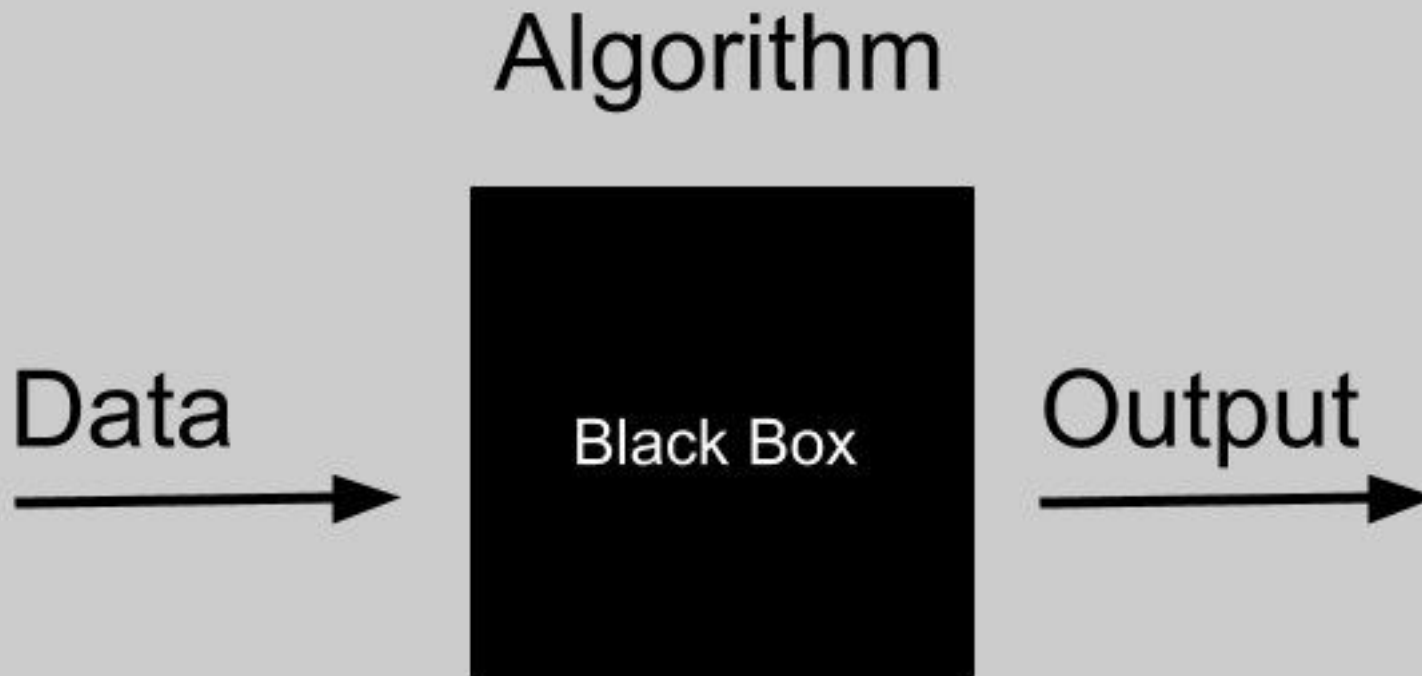
# Modules

- All programming languages come pre-packaged with a standard library of functions that are designed to make your job as a programmer easier
- Some of these functions are built right into the “core” of Python (print, input, range, etc)
- Other more specialized functions are stored in a series of files called “modules” that Python can access upon request by using the “import” statement
  - `import random`
  - `import time`

# Modules

- The import statement tells Python to load the functions that exist within a specific module into memory and make them available in your code
- Because you don't see the inner workings of a function inside a module we sometimes call them "black boxes"
- A "black box" describes a mechanism that accepts input, performs an operation that can't be seen using that input, and produces some kind of output

## “Black Box” model



## **More information about a module**

- To see information about a module, you can do the following in IDLE:
  - `help("modulename")`
- The `help()` function takes one argument (a string that represents the name of the module) and returns the user manual for that module

AirDrop

Recents

Applications

My Drive

Google Drive

webby-tellies

intro-to-computer-prog...

TBQ-WhatHappensinC...

writetonbro

Editorial

Documents

Downloads

Desktop

code-folder

PYTHON

myfunctions.py

PYTHON

myprogram.py

\_\_pycache\_\_

myfunctions.py - /Volumes/GoogleDrive/My Drive/teaching/intro-to-computer-program...

```
def sayHello(name):  
    return "Hello, " + name
```

Ln: 3 Col: 0

IDLE Shell 3.10.7

```
Python 3.10.7 (v3.10.7:6cc6b13308, Sep 5 2022, 14:02:52) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin  
Type "help", "copyright", "credits" or "license()" for more information.  
  
= RESTART: /Volumes/GoogleDrive/My Drive/teaching/intro-to-computer-programming/fall-22/code-folder/myprogram.py  
Hello, Emily  
>>>
```

Ln: 9 Col: 0

myprogram.py - /Volumes/GoogleDrive/My Drive/teaching/intro-to-computer-programmin...

```
import myfunctions  
  
print(myfunctions.sayHello("Emily"))
```



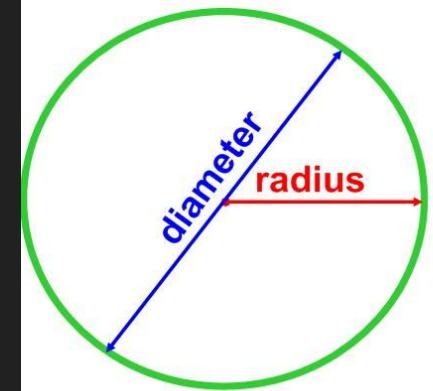
## Programming Challenge

Create a module called “geometry\_helper”

Write two functions in this module:

- Area of circle
- Perimeter of circle

Each of these functions will accept one argument (a radius) and will print out the result to the user.



Area of a circle  
 $= \pi \times \text{radius}^2$

Circumference of a  
circle  $= \pi \times \text{diameter}$

remember that the  
 $\text{diameter} = 2 \times \text{radius}$

myfunctions.py - /Volumes/GoogleDrive/My Drive/teaching/intro-to-computer-program...

```
import math

def getArea(r):
    return math.pi * (r**2)

def getPerimeter(r):
    return math.pi * 2 * r
```

Ln: 8 Col: 0

myprogram.py - /Volumes/GoogleDrive/My Drive/teaching/intro-to-computer-programming/fall-22/...

```
import myfunctions as m

r = 5
print("Area:", m.getArea(r))
print("Perimeter", m.getPerimeter(r))
```

Ln: 1 Col: 23

# Midterm Survey



[pollev.com/emilyzhao](https://pollev.com/emilyzhao)

## Topics Covered:

# Basic Programming Mechanics

- Functions
  - What is a function?
  - How to call a function
  - Arguments
  - Return Values
- Commenting your code
- Variables
  - What is a variable?
  - Creating variables
  - Using variables in expressions
  - Naming rules
- Reading input from the keyboard with the `input()` function

# Math Expressions

- Math operators (+, -, /, //, \*)
- Writing math expressions
- Evaluating math expressions
- Storing & printing the results of math expressions
- Difference between the two division operators (/ and //)
- Order of operations in math expressions
- The exponent operator (\*\*)
- The modulo operator (%)

# Data Types

- What is a data type?
- Strings
- Numeric data types
  - Integers (int)
  - Floating point numbers (float)
- Mixed type expressions
- Data type conversion
  - Using the float() and int() function to convert strings into numbers
  - User input & data types (converting strings to floats / ints for calculation purposes)
- The Boolean data type
- Boolean variables

# Output with the print() function

- General use of the print function and its default behavior
  - Unlimited arguments
  - Spaces inserted between arguments
  - Line break after each call to the function
- Customizing line endings (end=)
- Customizing argument separators (sep=)
- Escape characters (\n, \t, etc.)



# Basic String Manipulation

- Combining two strings (concatenation) – "+" operator
- Multiplying a string (repetition) – "\*" operator
- Formatting numbers using the format() function
  - Formatting Strings – width, left align, right align, center align
  - Formatting Integers – width, left align, right align, center align
  - Formatting Floats – width, left align, right align, center align, # of decimal places, "," separator
- Case manipulation using str.lower() and str.upper()
- Calculating string length using the len() function

# Selection Statements

- The structure of an IF statement (IF keyword, condition, colon, indentation)
- Writing a condition for an IF statement
- Boolean operators (<, >, ==, !=, >=, <=)
- Comparing numeric values using Boolean expressions
- Comparing string values using Boolean expressions
- Using the IF-ELSE statement
- Nesting decision structures (IF statements inside other IF statements)
- The IF-ELIF-ELSE statement
- Logical operators (and, or, not)

## Condition Controlled Loops

- The structure of a "while" loop
- Mechanics & how they work
- Setting up conditions for a while loop
- Infinite loops and how to work with them
- Sentinels (defining a value that the user enters that causes the loop to end)
- Input validation loops (asking the user to continually enter a value until that value matches some condition)
- Setting up and using accumulator variables
- Self referential assignment statements (i.e. `counter = counter + 1`)
- Augmented assignment operators (i.e. `counter += 1`)

## The Range Function

- mechanics and how the function works
- creating simple ranges (i.e. `range(5)`)
- creating ranges with defined start and end points (i.e. `range(3,10)`)
- creating ranges with a step value (i.e. `range(5,50,5)`)
- creating ranges that count backwards (i.e. `range(50,5,-5)`)
- user controlled ranges (i.e. `range(1, somevariable)`)

# Functions

- mechanics and how functions work
- function definitions
- arguments
- return values
- calling a function
- local variables (variables that are defined inside a function and can only be accessed inside that function)
- passing arguments to your own functions
- passing multiple arguments to your own functions
- global variables (variables created outside a function that can be accessed by any part of your program)
- making changes to global variables inside a function using the 'global' keyword
- writing a value returning function (i.e. using the 'return' keyword to send a result from your function to the part of your program that called that function)
- returning multiple values from a function
- Input, Processing & Output notation

# Miscellaneous Concepts

- Generating random numbers
- Errors & error types
- Debugging strategies
- Pseudocoding

# Midterm Review Game

Select all valid variable names

Next

▲ 0 ✓

◆ 0 ✓

● 0

■ 0 ✓

Show media

▲ bagel2



◆ Class



● 2bar



■ \_qux







5 % 7

Next

▲ 0

◆ 0 ✓

● 0

■ 0

Show media

▲ 0



◆ 5



● 7



■ 2



## What's the output?

Next

input = 2

```
while True:
    x = int(input("Enter a number (1 - 10): "))
    if x < 0 or x > 10:
        print("Invalid", end=" ")
        continue
    else:
        print("Thank you!", end=" ")
        break
print("All done.", end=" ")
```

▲ Invalid All done.

● Thank you!



■ Error

What's the output?

Next

```
name = "Emily"  
age = 26  
  
print("Emily is " + age + " years old")
```

▲ "Emily is 26 years old" ✕

◆ "Emily is age years old" ✕

● Error ✓

■ "Emily is26years old" ✕

```
num = 2
```

```
if num == 3 or 4:  
    print("case 1")  
else:  
    print("case 2")
```

▲ case 1

● error

■ nothing will print

```
for a in "HELLO THERE":  
    print(a, sep="*")  
    if a == "E":  
        break
```

▲ H\*E\*L\*L\*O\*T\*H\*E

● HE

■ none of the above

```
for a in "SM":  
    for b in "EA":  
        print(a, b, sep="", end="")
```

▲ SESAME



SESAMEMA

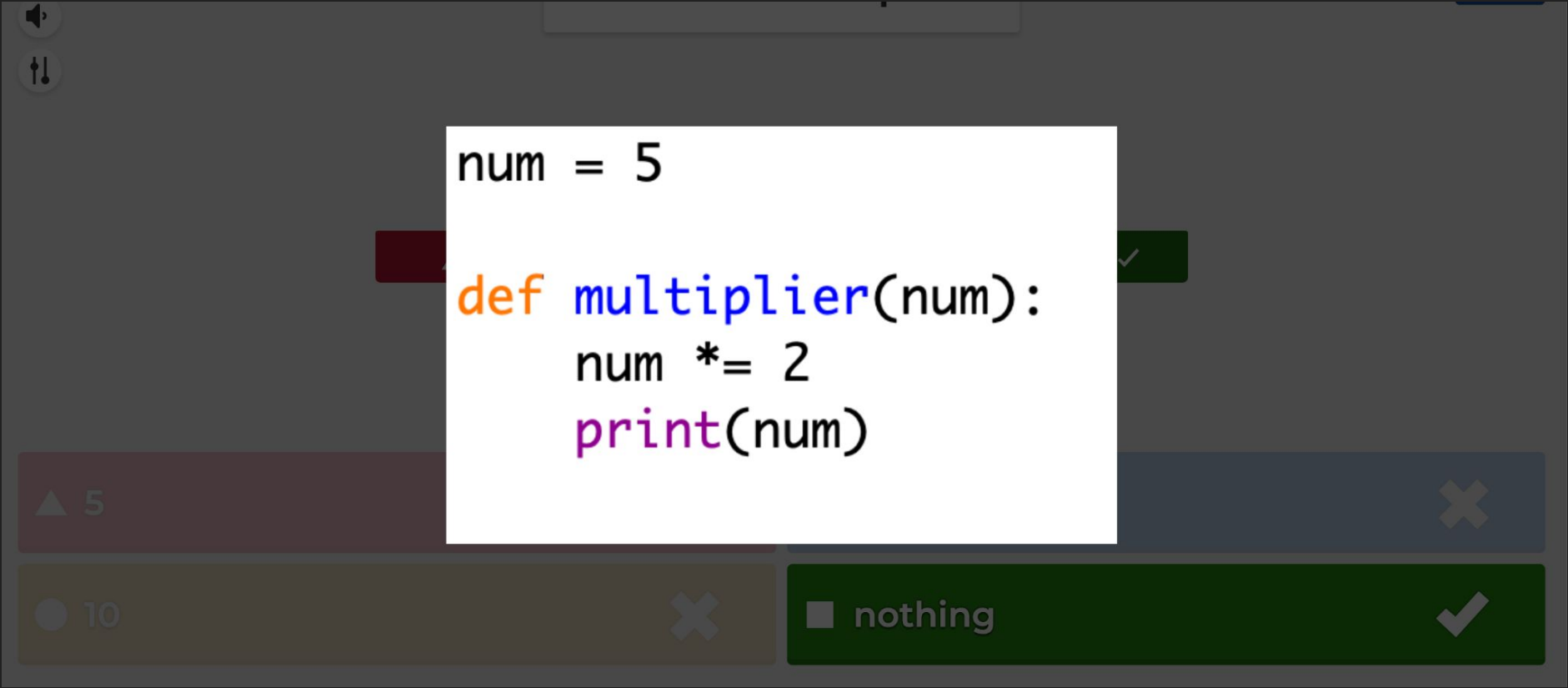


● SEAMEA



■ SEMA



The image shows a blurred background of a Python IDE interface. In the top-left corner, there are icons for a speaker and a pair of scissors. Below these, there are three colored buttons: a red one with a white 'x', a blue one with a white 'x', and a green one with a white checkmark. At the bottom, there are three more buttons: a blue one with a white triangle and the number '5', a green one with a white circle and the number '10', and a red one with a white square and the text 'nothing'. A white rectangular box is overlaid in the center, containing Python code.

```
num = 5
```

```
def multiplier(num):  
    num *= 2  
    print(num)
```

```
z = 23
q = -2
a = 4

if a < q:
    print('lion', end='_')
    if q >= z:
        print('eagle', end='_')
    elif z >= q:
        print('tuna', end='_')
    else:
        print('spider', end='_')
elif a < z:
    print('pear', end='_')
    if q < a:
        print('tomato', end='_')
    elif a < q:
        print('purple', end='_')
    else:
        print('dog', end='_')
else:
    print('potato', end='_')
    if z >= q:
        print('hyena', end='_')
    elif q >= z:
        print('cat', end='_')
    else:
        print('red', end='_')

print('aardvark')
```

pear\_tomato\_aardvark



```
def getName():  
    print("Emily")  
  
print(getName())
```

▲ Emily

● None

■ Emily \n None

# Homework

- Midterm next Thursday
- Assignment #6 (due next Thursday)