`Week: 03 Date: 02/01/2024

| 15-110 Recitation Week 3 |
| --- |

# **Reminders**

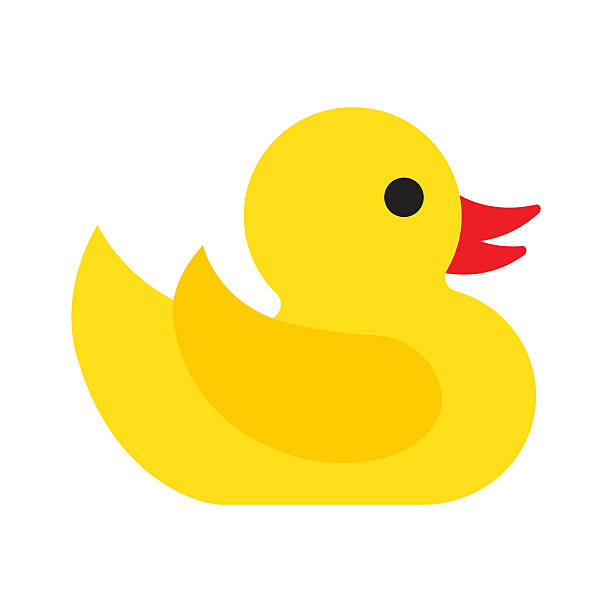
* Check 2 due Monday 2/05 at Noon!
* Check your email if a TA can’t find you during OH
* HW 1 grades are out!
* Check 1 and HW1 Revisions (+Ex revisions) due 2/06 at Noon
* For recitation today:
  + There is a starter code file on the website
  + Recitation feedback form: <https://forms.gle/dWgvmGvTSMbRi7rv5>

# **Overview**

* Debugging
* Conditionals Practice
* Circuits and Gates

| Problems |
| --- |

# **DEBUGGING**

1. Rubber Duck Debugging 
2. Printing and Experimenting
3. Thorough Tracing

**Catch the Error:**

**These are also in the starter file, with some additional code to help you.**

For each function, specify **the type of error** (syntax/runtime/logical), **error name,**  and **what the error means.**

1. def computeIntAverage(x, y, z):

sum = x + y + z

count = 3

return sum // count

print(computeIntAverage(5, “6”, 7))

| **Type:** |  |
| --- | --- |
| **Name:** |  |
| **What it means:** |  |

1. def findLineIntersect(m1, b1, m2, b2):

x = (b2 - b1) / (m1 - m1)

y = m1 \* x + b1

print(‘y coord’, y)

return x

| **Type:** |  |
| --- | --- |
| **Name:** |  |
| **What it means:** |  |

1. # leaving is a boolean that tells us if we should say hello or goodbye

def sayHelloOrGoodbye(leaving, name):

if leaving == True:

return “Hello ”+ name

else:

return “Goodbye, ”+ name+ “ and have a great day!

| **Type:** |  |
| --- | --- |
| **Name:** |  |
| **What it means:** |  |

1. import math

def circleArea(x0, y0, x1, y1):

minRad = x\_1 - x0 / 2

maxRad = y1 - y0 / 2

area = math.pi \* minRad \* maxRad

return area

| **Type:** |  |
| --- | --- |
| **Name:** |  |
| **What it means:** |  |

1. import math

def roundDistance(x1, y1, z1, x2, y2, z2):

dX = x2 - x1\*\*2

dY = (y2 - y1) \*\* 2

dZ = pow(2, (z1 - z2) )

dist = math.ceil(dX + dY + dZ) \*\* 0.5

return dist

| **Type:** |  |
| --- | --- |
| **Name:** |  |
| **What it means:** |  |

# 

# **CONDITIONALS PRACTICE**

In the heart of the fantastical town of Ticktockton lies the Enigmatic Clock Tower, which chimes magical melodies based on the time of day and the weather.

As the timekeeper of the Clock Tower, your responsibility is to decide the tower's chimes. The head timekeeper left you this python function to help:

def clockTower(time, weather):

if time == "Morning":

if weather == "Sunny":

print("Chime of Cheerful Chipmunks")

else:

print("Melody of Morning Dew")

elif time == "Afternoon":

if weather == "Windy":

print("Anthem of Airy Adventures")

else:

print("Ballad of Breezy Bluffs")

print(“With an encore Serenade of Starry Skies")

elif weather == "Rainy":

print("Raindrop Rhapsody")

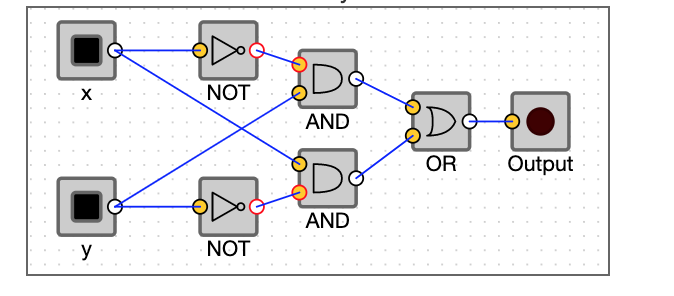
else:

print("Twilight Tune")

| Function Call | Displayed in Shell |
| --- | --- |
| clockTower(“Evening”, “Rainy”) |  |
| clockTower(“Afternoon”, “Windy) |  |
| clockTower(“Morning”, “Rainy”) |  |
| clockTower(“Night”, “Cloudy”) |  |

# **CIRCUITS AND GATES**

Write the equivalent Boolean expression demonstrated by the circuit:



Fill out the truth table that corresponds to the circuit and boolean expression above:

| **x** | **y** | **output** |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |