1. What is duck typing?

Python’s method of deciding what can and can’t be done with values stored in variables. If you try to call a method on object, it’s legal as long as the object has the method.

How does this pertain to object-oriented programming (programming with classes)?

It allows us to write multiple classes with the same interface and use these objects interchangeably.

What is the following code’s output?

Module 1:

class OrderedPair:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def calculate(self):

return self.x + self.y

def get\_coordinates(self):

return (self.x, self.y)

class FractionalCoordinates:

def \_\_init\_\_(self, x, y):

self.x = x

self. y = y

def calculate(self):

return (x\*2000, y\*2000)

def get\_coordinates(self):

return(self.x, self.y)

Module 2:

if \_\_name\_\_ == ‘\_\_main\_\_’:

a = OrderedPair(500, 500)

b = FractionalCoordinates(.5, .5)

c = FractionalCoordinates(.75, .75)

d = OrderedPair(1000, 1000)

list = [a, b, c, d]

for item in list:

print(item.calculate())

1000

(1000.0, 1000.0)

(1500.0, 1500.0)

2000

1. What is the “with” statement? What’s an example of its usage?

Allows you to use a file and automatically close it outside of the with statement or if an error is raised

**def** one():

**with** open(‘some file.txt’, ‘w’) **as** b:

print('Inside with statement')

**def** two():

**with** open(‘some file.txt’, ‘r’) **as** b:

print('Inside with statement')

**raise** ValueError('This is not cool')

print('Still inside with statement')

1. What are events in tkinter?

In event based programming, you tell tkinter what events your code would find “interesting”. When an interesting event happens, your code will process it.

What does the following code do?

import random

import tkinter

class App:

def \_\_init\_\_ (self):

self.\_root = tkinter.Tk()

self.\_canvas = tkinter.Canvas(master = self.\_root,

width = 500, height = 500, background = ‘#000000’)

self.\_canvas.grid(row = 0, column = 0, padx = 10, pady = 10)

self.\_previous =[]

self.\_current = ‘#000000’

self.\_canvas.bind(‘<Button-1>’, self.\_on\_left\_click)

self.\_canvas.bind(‘<Button-2>’, self.\_on\_right\_click)

def start(self):

self.\_root.mainloop()

def \_on\_left\_click(self, event):

self.\_previous.append(self.\_current)

random = ‘#’

for x in range(6):

random += random.randint(0, 9)

self.\_current = random

self.\_canvas = tkinter.Canvas(master = self.\_root,

width = 500, height = 500, background = random)

def \_on\_right\_click(self, event) if len(self.\_previous) == 0: self.\_current = ‘#000000’ else: self.\_current = self.\_previous[len(self.\_previous) -1] self.\_previous.pop() self.\_canvas = tkinter.Canvas(master = self.\_root, width = 500, height = 500, background = self.\_current)

1. What’s the difference between grid() and pack()?

Grid allows you to place widgets onto different cells on an invisible grid

Pack automatically handles the placement of the widgets

1. What’s the difference between fractional coordinates and absolute coordinates?

Fractional coordinates are based on the window size

Absolute coordinates are actual pixel distances

From before the midterm:

1. Review try, except, else, finally statements.
2. Programming with classes. Make sure you’re able to write your own classes!