**2013 SI@UCF Object-Oriented Programming in Python Test #1 Solutions**

**Date: 7/15/2013**

1) (10 pts) Fill in the following class definition. The \_\_init\_\_ function should set self.color to the color parameter and self.miles to the miles parameter. The drive function should make self.miles incremented by additional\_miles.

class **Car**:

 def **\_\_init\_\_**(*self*, color = *"black"*, miles = 0):

 self.color = color (3 points)

 self.miles = miles (3 points)

 def **drive**(*self*, additional\_miles):

 self.miles += additional\_miles (4 points)

2) (5 pts) What is the output of the following code segment?

number1 = 7

number2 = 5

if (number1 - number2 > 0 and number1 == 7):

 if (number2 > number1 or number2 < 8):

 print(*"Oranges"*)

 else:

 print(*"Orangutan"*)

else:

 print(*"Salad"*)

Output: Oranges (5 points for Oranges and 0 points for anything else)

3) (15 pts) Complete the following function definition. This function should return the value of the base parameter raised to the exponent parameter. As an example, the value of power(2, 4) is 16 because 24 = 16. Note: You may NOT use the \*\* operator or call the pow function. You must use a loop and use multiplication. (You only need to worry about positive, integer exponents).

def **power**(base, exponent):

 answer = 1

 for i in range(exponent):

 answer \*= base

 return answer

(3 points for initializing answer)

(5 points for looping construct)

(5 points for updating answer)

(2 points for return)

4) (5 pts) What is the output of the following code segment?

class **SnorMax**:

 def **\_\_init\_\_**(*self*, hours\_rested=2):

 *self*.hours\_rested = hours\_rested

 def **sleep**(*self*, hours):

 *self*.hours\_rested += hours

small\_bear = SnorMax()

small\_bear.sleep(5)

print(str(small\_bear.hours\_rested))

Output: 7 (5 points for 7, 2 points for 5)

5) (5 pts) Will the following code segment ever complete execution? Why or why not?

countdown = 10

while (countdown > 0):

 print(str(countdown) + *" second(s) until launch."*)

print(*"We have lift off."*)

Answer: countdown is never decremented -> infinite loop

(5 points for something relating to infinite loop, 0 points for anything else)

6) (15 pts) Write a segment of code that simulates rolling a pair of fair six-sided dice 1000000 times and reports the sum of all those rolls as well as the number of times snake-eyes (a sum of 2 on the pair of dice) was rolled out of the million times.

import math

sum = 0

snake = 0

for i in range(1000000):

 total = math.randint(1,6) + math.randint(1,6)

 if (2 == total):

 snake += 1

 sum += total

print(“sum is “+ str(sum))

print(“snake-eyes is “ + str(snake))

(2 points for initializing counters)

(2 points for loop)

(4 points for getting total in some way)

(4 points for checking for snake-eyes)

(2 points for updating sum)

(1 point for printing)

7) (15 pts) What is the output of the following segment of code?

str = "ABCDEFGHIJ"

print(str[2:7])

print(str[:8])

print(str[3:])

print(str[-6:-2])

print(str[:-5])

CDEFG

ABCDEFGH

DEFGHIJ

EFGH

ABCDE

(3 points per line)

8) (25 pts) Complete the money class shown below, that stores the dollars and cents for an amount of money as integers.

class money:

 def \_\_init\_\_(self, dollars=0, cents=0):

 self.dollars = dollars + cents//100

 self.cents = cents%100

 # amount is a money object, this amount should be

 # subtracted out of self, and self should be adjusted

 # accordingly.

 def spend(self, amount):

 self.cents -= amount.cents

 if (self.cents < 0):

 self.dollars -= 1

 self.cents = 100 + self.cents

 self.dollars -= amount.dollars

 (2 points for subtracting cents)

 (6 points for checking and fixing negative cents)

 (2 points for subtracting dollars)

 # amount is a money object, this amount should be

 # added to self, and self should be adjusted

 # accordingly.

 def add(self, amount):

 self.cents += amount.cents

 if (self.cents > 99):

 self.cents %= 100

 self.dollars += 1

 self.dollars += amount.dollars

 (2 points for adding cents)

 (6 points for checking and fixing extra cents)

 (2 points for adding dollars)

 # Returns the number cents self is worth. (ie $2.34 should

 # return 234.

 def numCents(self):

 return str(100\*self.dollars + self.cents)

 (5 points for correct)

9) (5 pts) For what breakfast food is Dunkin Donuts known? Donuts