Homework 2

Kevin Leary

Problem 1

Psuedo-code:

Ask user for limit for Pythagorean triple

Create variables to be used throughout program

While C^2 is less than the limit

For each m in the range of 1 to n+1

Do calculations for Pythagorean triple

If a b or c is 0 then

End loop

If a b or c is divisible by a primitive pythagoren triple

End loop

Print a b and c

Increment n

Set variables x and c

While c is less than the limit entered

Calculate all the primitives

Print a b and c

Increment x

```
limit=int(input("Enter upper limit: "))
c=0
n=2
while(c<limit):     #as long as C is lower then the limit
    for m in range(1,n+1):
        a=n*n - m*m
        b=2*n*m
        c=n*n+m*m
        if(c>limit):    #goes til limit
            break
        if(a==0 or b==0 or c==0):       #wont display a 0
            break
```

```
        if((a%3 == 0 and b%4==0 and c%5 == 0) or (a%4 == 0 and b%3 == 0 and
c%5 == 0)):   #wont duplicate a primitive triple
            break
        print(a,b,c)
    n=n+1

x = 1
c = 0
while(c<limit):     #for primitive triples

    a = x*3
    b = x*4
    c = x*5

    print(a,b,c)

    x = x+1
```

Problem 2

In function find_dup_str

For each character in the string s

Set variable og_string to s with splicing

If the length of the substring is less than the given length

Return “”

For each character in the string s

Set variable dup_string to s with correct splicing

If og_string and dup_string are equal

return one of them

If the length is not the same between og and dup

Then stop comparing them

If og and dup are equal

Return one of them

Out of function find_dup_str

Ask user for a string and a length

Print find_dup_str

In function find_max_dup

Set iter to one less than the length

While iter is greater than 0

If function find_dup_str does not equal “”

then end loop

decrement iter

return find_dup_str

call functions

```
def find_dup_str(s, n):
    #find the first letter in the string then compare to the others
    og_string = ''
    dup_string = ''

    for i in list(range(len(s))):         #iterate through string
        og_string = s[i:i+n:1]
        #j = n+i                              #alwasy starts right after
og_string
        if(len(s[i:i+n:1]) < n):             #case for if the string is less
then n
            return ""

        for j in list(range(n+i, len(s))):
            dup_string = s[j:j+n:1]
            #print(s[j:j+n:1])

            if(og_string == dup_string):
                return og_string

        if(len(og_string) != len(dup_string)):      # when it stops being
same len stop comparing
                return ""

        if(og_string == dup_string):
                return og_string


        if(og_string != dup_string and i == (len(s) - 1)):
            return ""


def find_max_dup(s):

    i = int(len(s) - 1)          #i starts at second element as it becomes n

    while(i > 0):        #increment through
```

```
        if(find_dup_str(s, i) != ""):
            break

        i = i-1

    return find_dup_str(s, i)

s = input("Enter a string to find dups: ")
#n = int(input("Enter the length of duplicate: "))

#print(find_dup_str(s,n))
print(find_max_dup(s))
```

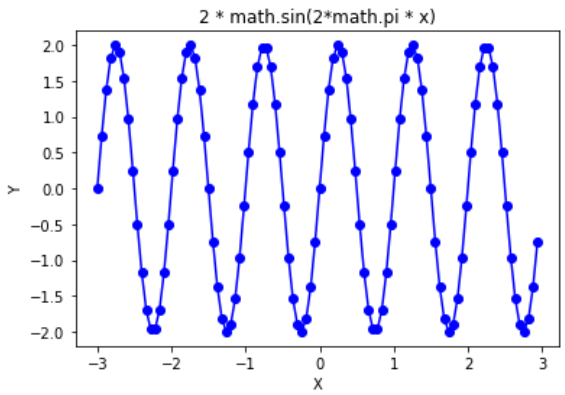
Problem 3

```
Enter function with variable x: 2 * math.sin(2*math.pi * x)

Enter the number of samples: 100

Enter xmin: -3

Enter xmax: 3
          X             Y
--------------------------
         -3.0          1.4695761589768238e-15
         -2.94          0.73624910536936
         -2.88          1.3690942118573772
         -2.82          1.8096541049320403
         -2.76          1.9960534568565436
         -2.6999999999999997          1.9021130325903068
         -2.6399999999999997          1.5410264855515758
         -2.5799999999999996          0.9635073482034244
         -2.5199999999999996          0.25066646712860235
         -2.4599999999999995          -0.49737977432971486
         -2.3999999999999995          -1.175570504584953
         -2.3399999999999994          -1.6886558510040341
         -2.2799999999999994          -1.9645745014573792
         -2.2199999999999993          -1.9645745014573757
         -2.1599999999999993          -1.6886558510040255
         -2.099999999999999          -1.1755705045849367
         -2.039999999999999          -0.497379774329699
         -1.979999999999999          0.2506664671286221
         -1.919999999999999          0.963507348203442
         -1.859999999999999          1.5410264855515863
--------------------------
More values hidden
```



```
import pylab



fun_str = input("Enter function with variable x: ")
ns = int(input("Enter the number of samples: "))
xmin = float(input("Enter xmin: "))
xmax = float(input("Enter xmax: "))

xrange = (xmax - xmin) / ns
x = xmin
i = 0

xs = list()
ys = list()

while(x <= xmax):
    xs.append(x)
    x += xrange

for x in xs:
    y = eval(fun_str)
    ys.append(y)

print("          X             Y")
print("--------------------------")
while(i < 20):

    print("          {0}           {1}".format(xs[i], ys[i]))
    i= i+1;

print("---------------------------")
print("More values hidden")

pylab.plot(xs,ys, "bo-")

pylab.xlabel("X")
pylab.ylabel("Y")
pylab.title(fun_str)
```

Problem 4

```
def input_tuple(prompt, types, sep):

    try:
        new_tuple = tuple()
        info = input(prompt)              #setting up user given info
        info_extra = info.split(sep)
        if len(info_extra) != len(types):
            return new_tuple         #returns empty tuple
        else:
            for i in range(len(types)):
```

```
                new = types[i](info_extra[i])
                new_tuple.append(new)
            new_tuple = tuple(new_tuple)
        return new_tuple

    except ValueError:
        print("Wrong parameters entered")
        return ()

#part B

def input_tuple_lc(prompt, types, sep):

    try:
        new_list=[]
        info = input(prompt)
        info_extra = info.split(sep)

        if len(info_extra) != len(types):
            return new_list             #returns empty list

        else:
            new_list=[types[i](info_extra[i]) for i in range(len(types))]
            new_tuple=tuple(new_list)
        return new_tuple

    except ValueError:
        print("Wrong parameters entered")
        return ()

#part C

def read_tuple(file_obj, types, sep):       #extra

    try:
        new_list=[]
        new_list=[types[i](file_obj[i]) for i in range(len(types))]
#reading from the file
        new_tuple=tuple(new_list)
        return new_tuple

    except ValueError:
        print("Wrong parameters entered")
        return ()

call1 = input_tuple("Enter first name, last name, age (float), ID (int),
fulltime (bool): ", (str, str, float, int, bool), ',')
call2 = input_tuple("Enter first name, last name, age (float), ID (int),
fulltime (bool): ", (str, str, float, int, bool), ',')

f = open("cars.csv", "r")

    for line in f:
```

```
        cars=read_tuple(line, (str, str, float, int, bool), ',')
        print(cars)
```

Problem 5

```
#Part A
numbers = [4, -3, 0, 2, -1, 5]
numbers_squared = ['y*y=' + str(n*n) for n in numbers]
#print(numbers_squared)

#part B

numbers_solution = ['solution #' + str(n+1) + '=' + str(numbers[n]**2) for n
in range(len(numbers))]
print(numbers_solution)

#part C

lst = ["zero", "one", "two", "three"]

new_lst = [str(i) + ' ' + lst[i] for i in range(len(lst))]

print(new_lst)

#part D

a = ['a','b','c']
b = [1,2]
cartesian = []

for x, y in [(x,y) for x in a for y in b]:
    cartesian += (x,y)

print(cartesian)

#part E

lst1 = [56, 25, 8, 11, 16, 20, 18, 50, 7, 42]
lst2 = [5, 3, 6]
```

Problem 6

```
def get_csv_data(f, string_pos_lst, sep=","):

    lebron_lst = []

    f.readline()
    for line in f:          #iterate through
        line_list = line.split(sep)
        if(string_pos_lst in line_list):        #special str values entered
            lebron_tuple = (str(string_pos_lst))
```

```
            lebron_list.append(lebron_tuple)
    return lebron_lst

#part B

def get_columns(lebron_lst, cols_lst):

    col_data = []
    for column in cols_lst:
        try:
            index = lebron_lst[0].index(column)
            tmp_col_data = []
            for lbj in lebron_lst[1:]:      #iterate through list with a
splice
                tmp_col_data.append(lbj[index])
            col_data.append(tmp_col_data)

        except ValueError:
            print("Wrong columns taken")
            pass                #keep it going

    return col_data


bb_file = open("lb-james.csv", "r")     #error here for some reason
james_lst = get_csv_data(bb_file, [0, 2, 3, 4], ",")
print(james_lst)

selected_cols_lst = get_columns(james_lst, ["Season","Age","PTS"])

selected_col_list = get_columns(james_list, ["Season", "3P%", "2P%", "FT%"])

x_axis = [int(x.split("-")[0]) for x in selected_col_list[0]]
y1 = selected_col_list[1]
y2 = selected_col_list[2]
y3 = selected_col_list[3]

pylab.plot(x_axis, y1, '-b', label="3-point precentage")
pylab.plot(x_axis, y2, '-r', label="2-point precentage")
pylab.plot(x_axis, y3, '-g', label="free throw precentage")

pylab.title("Lebron vs Time")
pylab.xticks(x_axis, x_axis)

pylab.xlabel("Year")
pylab.ylabel("Percentage")
pylab.ylim(0, 1.0)
```