Michael Quach

Problem 1 Pythagorean Numbers

a)

```
Read input “limit”
for a=1 to limit:
        for b=1 to limit:
                c=sqrt((a*a)+(b*b)
                if c<limit:
                        print the triple
                b+=1
        a+=1
        b=1
```

```
#Michael Quach
import math
#Read input “limit”
limit=input("Hi. I calculate Pythagorean Triples. \
How far do you want me to look for them?\nEnter an integer: ")
limit=int(limit)
print("a\tb\tc")
a,b,c=1,1,1
while a<limit:
    while b<limit:
        c=((a*a)+(b*b))
        #d will serve as a temporary sqrt(c), saved to compare later
        d=math.sqrt(c)
        d=int(d)
        #Now that d is an int, it should be equal to sqrt(c)
        #if sqrt(c) is an int (even if it's technically a float)
        if d<limit and math.sqrt(c)==d:
            print(a,"\t",b,"\t",d)
        b+=1
    a+=1
    b=1
```

Problem 2 Duplicated Substrings

```
find_dup_str(s, n):
        for each substring of length n:
                cycle through later substrings of length n and check if they are the same as the
                        previous substring
```

```
                if same, return the substring
                if not, continue
        if nothing's found, return empty string

find_max_dup(s)
        for i=0 to length of s
                if find_dup_str(s,i) does not return “”
                        answer = find_dup_str(s,i)
        return answer

#Michael Quach
#int n is length of duped substring to find in string s
def find_dup_str(s, n):
    left=0
    right=n
    i=0

    if(n>len(s)):
        return ""
#For all characters in string s:
    while(i<len(s)):
        i+=1
#If a duplicate character is found:
        if(s[left:right]==s[left+i:right+i]):
            #print ("Substring", s[left:right], "found at positions",\
            #       left, "and", left+i, ".")
            return s[left:right]
#if i reaches the end of the string but we haven't compared every substring yet
        if(i==(len(s)-1) and right<(len(s)-1)):
            #reset to the beginning of the string
            i=1
            #but increment the chosen substring
            left+=1
            right+=1
    return ""

def find_max_dup(s):
    for i in range(0,len(s)):
        temp=find_dup_str(s,i)
        if(temp!=""):
            answer=temp
    return answer
```

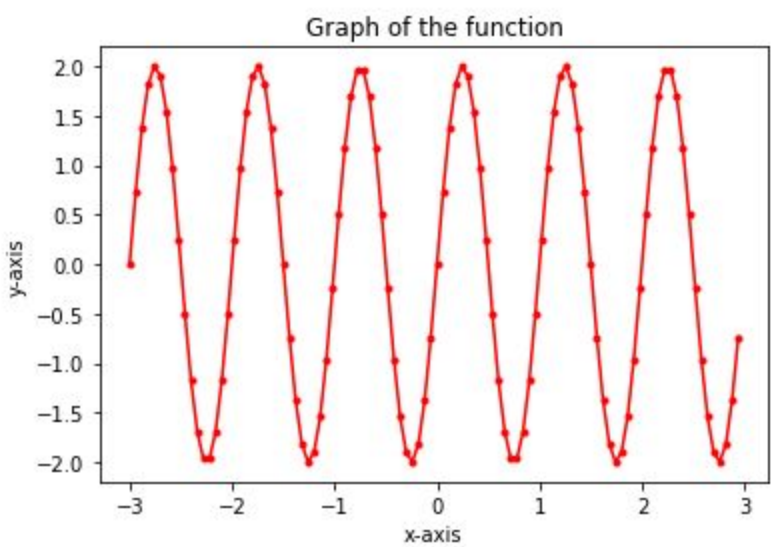
Problem 3 Function Visualization

```
#Michael Quach
import pylab
import math
#Problem 3 Function Visualization
#Read in string fun_str [x as parameter], domain, and int ns
fun_str = input("Enter fun_str, using x as the parameter: ")
x0 = input("Enter beginning of domain, x0: ")
x1 = input("Enter end of domain, x1: ")
ns = input("Enter number of points: ")
x0, x1, ns = float(x0), float(x1), int(ns)
xs=[]
ys=[]
#Fill ys=[] with results of evaluating xs on fun_str expression
dx=(x1-x0)/ns
x=x0
while x<=x1:
    xs.append(x)
    y = eval(fun_str)
    ys.append(y)
    x+=dx
#Print table of xs and ys,
print("\tx\t\ty")
for counter in range(0,ns):
    print("{:10.2f}\t{:10.2f}".format(xs[counter],ys[counter]))
#counter=0
#while counter<=ns:
#    print("x=",xs[counter],"\ty=",ys[counter])
#    counter+=1
#Print graph of xs and ys
pylab.plot(xs, ys, "r.-")
pylab.title("Graph of the function")
pylab.xlabel("x-axis")
pylab.ylabel("y-axis")
pylab.show()
```

IPython console

Console 1/A

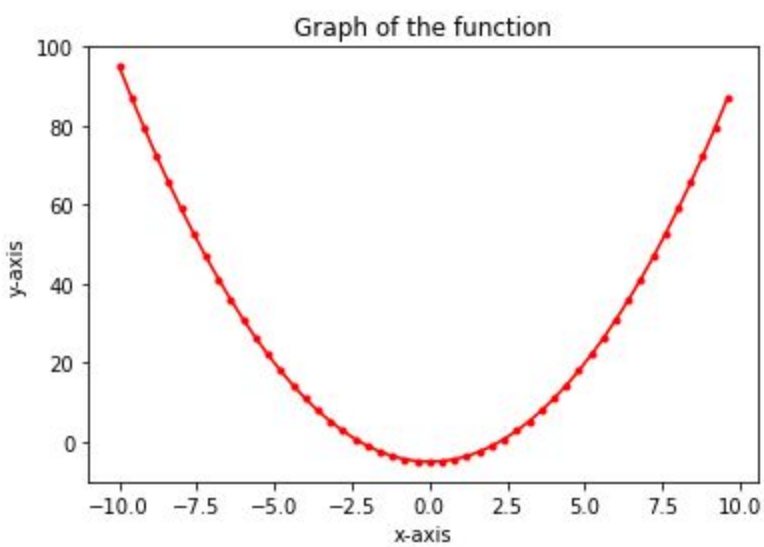
```
1.98          -0.25
2.04           0.50
2.10           1.18
2.16           1.69
2.22           1.96
2.28           1.96
2.34           1.69
2.40           1.18
2.46           0.50
2.52          -0.25
2.58          -0.96
2.64          -1.54
2.70          -1.90
2.76          -2.00
2.82          -1.81
2.88          -1.37
2.94          -0.74
```



IPython console

Console 1/A

```
      3.60              7.96
      4.00             11.00
      4.40             14.36
      4.80             18.04
      5.20             22.04
      5.60             26.36
      6.00             31.00
      6.40             35.96
      6.80             41.24
      7.20             46.84
      7.60             52.76
      8.00             59.00
      8.40             65.56
      8.80             72.44
      9.20             79.64
      9.60             87.16
```



In [4]: