```python
"""
Homework 4 Problem 13.13
Develop a progrom using a programing or macro language to implement the
golden-section serach algorithm. Design teh program so that it is expressly
designed to locate a maximum or minimum based on user preference. The
subroutine should have the following features:
    -iterate untill the relative error falls below a stopping criterion or
    exceeds a maximum number of iterations.
    -return both the otimal x and f(x)
    -minimize the number of function evaluations

@author: Jacob Needham
"""

#getting user decision on max or min
def wantMaxOrMin():
    count = 0
    answer = False
    while answer != True and count<5:
        print("Type \"MAX\" if you would like to find the max or \"MIN\" \
              to find the minimum of the function")
        user = input()
        count +=1
        if user == "MAX":
            return True
            break
        if user == "MIN":
            return False
            break
        elif user != "MAX" or user != "MIN":
            print("\nInvalid input, try again")


#function equation, in a real problem the function would come from the user
def function(x):
    y = (x**3 - 30*x**2 - 661*x -1791)/1791
    return y


#Golden search subroutine
def goldenSearchMax(es,iMax,L,U):
    iCount = 0
    ea = es
    R = ((5**.5)-1)/2
    xL = L
    xU = U
    d = R *(xU - xL)
    x1 = xL + d
    x2 = xU - d
    f1 = function(x1)
    f2 = function(x2)
    if f1>f2 :
        xopt = x1
        fx = f1
    else:
        xopt = x2
        fx = f2
```

```python
    for iCount in range(iMax):
        d = R*d
        xinit = xU - xL
        if f1 > f2:
            xL = x2
            x2 = x1
            x1 = xL + d
            f2 = f1
            f1 = function(x1)
        else:
            xU = x1
            x1 = x2
            x2 = xU - d
            f1 = f2
            f2 = function(x2)
        iCount += 1
        if f1 > f2:
            xopt = x1
            fx = f1
        else:
            xopt = x2
            fx = f2
        if xopt != 0:
            ea = (1-R)* abs(xinit/xopt)*100
        if ea < es:
            break
    print("\nNumber of iterations to convergence:", iCount)
    print("The optimal value is:", round(fx,4))
    print("The x value for the optimal value is:",round(xopt,4))


def goldenSearchMin(es,iMax,L,U):
    iCount = 0
    ea = es
    R = ((5**.5)-1)/2
    xL = L
    xU = U
    d = R *(xU - xL)
    x1 = xL + d
    x2 = xU - d
    f1 = function(x1)
    f2 = function(x2)
    if f1<f2 :
        xopt = x1
        fx = f1
    else:
        xopt = x2
        fx = f2
    for iCount in range(iMax):
        d = R*d
        xinit = xU - xL
        if f1 < f2:
            xL = x2
            x2 = x1
            x1 = xL + d
            f2 = f1
            f1 = function(x1)
```

```
        else:
            xU = x1
            x1 = x2
            x2 = xU - d
            f1 = f2
            f2 = function(x2)
        iCount += 1
        if f1 < f2:
            xopt = x1
            fx = f1
        else:
            xopt = x2
            fx = f2
        if xopt != 0:
            ea = (1-R)* abs(xinit/xopt)*100
        if ea < es:
            break
    print("\nNumber of iterations to convergence:", iCount)
    print("The minimal value is:", round(fx,4))
    print("The x value for the minimal value is:",round(xopt,4))


def main():
    answer = wantMaxOrMin()

    if answer == True:
        goldenSearchMax(.001,200,-20,0)
    else:
        goldenSearchMin(.001,200,-20,0)
main()
```

```
OUTPUT
Type "MAX" if you would like to find the max or "MIN"
to find the minimum of the function

MAX

Number of iterations to convergence: 25
The optimal value is: 0.595
The x value for the optimal value is: -7.8978
```

```python
"""
Homework 4 Problem 13.15
Develop a program using a programming or macro language to implement Newton's
method. The subroutine should have the following features:
    -Iterate until the relative error falls below a stopping criterion or
    exceeds a maximum number of iterations.
    -Returns both the optimal x and f(x).
Test your program with the same problem as Example 13.3.

@author: Jacob Needham
"""
#function equation
def function(x):
    y = -1.5*x**6 - 2*x**4 + 12*x
    return y

#first derivitive
def functionDer(x):
    y = -9*x**5 -8*x**3 + 12
    return y

#second derivitive
def functionDer2(x):
    y = -45*x**4 -24*x**2
    return y


#Newton Raphson technique
def newtonMethod(x0,es,iMax):
    ea=es
    xR = x0
    iCount = xOld = ea = None
    for iCount in range(0,iMax):
        iCount +=1
        xOld = xR
        xR = xOld - (functionDer(xOld)/functionDer2(xOld))
        if xR != 0:
            ea = abs((xR-xOld)/xR)*100
        if ea<es or iCount > iMax:
            break
    print("The max was found after", iCount,"iterations")
    print("The x value of the max of the function is: ", round(xR,4))
    print("The function evaluated at the max is:", round(function(xR),4))
    print("The error is: ", round(ea*100,4) , "%")

newtonMethod(2, .001, 30)
```

```
OUTPUT
The max was found after 7 iterations
The x value of the max of the function is:  0.9169
The function evaluated at the max is: 8.6979
The error is:  0.0197 %
```

```python
# -*- coding: utf-8 -*-
"""
Homework 4 Problem 13.20
The normal distribution is a bell-shaped curve defined by y=e^(-x^2).
Use the golden-section search to determine the location of the inflection
point of this curve for positive x.

@author: Jacob Needham
"""
import math

#function equation
def function(x):
    y = math.exp(-x**2)
    return y

def functionDer(x):
    y = -2*x*math.exp(-x**2)
    return y

def functionDer2(x):
    y = (4*x**2-2)*math.exp(-x**2)
    return y


def goldenSearchMin(es,iMax,L,U):
    iCount = 0
    ea = es
    R = ((5**.5)-1)/2
    xL = L
    xU = U
    d = R *(xU - xL)
    x1 = xL + d
    x2 = xU - d
    f1 = functionDer(x1)
    f2 = functionDer(x2)
    if f1<f2 :
        xopt = x1
        fx = f1
    else:
        xopt = x2
        fx = f2
    for iCount in range(iMax):
        d = R*d
        xinit = xU - xL
        if f1 < f2:
            xL = x2
            x2 = x1
            x1 = xL + d
            f2 = f1
            f1 = functionDer(x1)
        else:
            xU = x1
            x1 = x2
            x2 = xU - d
            f1 = f2
            f2 = functionDer(x2)
```

```
        iCount += 1
        if f1 < f2:
            xopt = x1
            fx = f1
        else:
            xopt = x2
            fx = f2
        if xopt != 0:
            ea = (1-R)* abs(xinit/xopt)*100
        if ea < es:
            break
    print("The x value for the inflection point is:",round(function(fx),4))
    print("Number of iterations to convergence:", iCount)


goldenSearchMin(.001,100,-20,20)
```

OUTPUT

```
The x value for the inflection point is: 0.4791
Number of iterations to convergence: 32
```

```python
"""
Homework 4 Problem 14.9
Develop a program using a programming or macro language to implement the
random search method. Design the subprogram so that it is expressly designed
to locate a maximum. Test the program with f(x, y) from Prob. 14.7. Use a
range of -2 to 2 for both x and y.

@author: Jacob Needham
"""

import random

#function equation
def function(x,y):
    z = 4*x + 2*y + x**2 - 2*x**4 +2*x*y - 3*y**2
    return z

#Random Search takes in a number of points and finds the max value of a function
def randomSearch(maxNumPoints):
    listOfPoints = []
    for i in range(maxNumPoints):
        randPointX = random.uniform(-2,2)
        randPointY = random.uniform(-2,2)
        listOfPoints.append ([function(randPointX,randPointY),randPointX,randPointY])
    index = listOfPoints.index(max(listOfPoints))
    domainMax = listOfPoints[index][0]
    xValue = listOfPoints[index][1]
    yValue = listOfPoints[index][2]


    print("The max of the set")
    print("4*x + 2*y + x^2 - 2*x^4 + 2*x*y - 3*y^2")
    print("bounded by -2<=x<=2 and -2<=y<=2 and with", maxNumPoints, "searches is:")
    print(round(domainMax,3),
          " at x=", round(xValue,3),
          ", y=", round(yValue,3),sep="")

randomSearch(1000)
```

```
OUTPUT
The max of the set
4*x + 2*y + x^2 - 2*x^4 + 2*x*y - 3*y^2
bounded by -2<=x<=2 and -2<=y<=2 and with 1000
searches is:
4.271 at x=0.899, y=0.534
```

```python
"""
Homework 4 Problem 14.10
The grid search is another brute force approach to optimization. The
two-dimensional version is depicted in Fig. P14.10. The x and y dimensions
are divided into increments to create a grid. The function is then evaluated
at each node of the grid. The denser the grid, the more likely it would be to
locate the optimum.
Develop a program using a programming or macro language to implement the grid
search method. Design the program so that it is expressly designed to locate a
maximum. Test it with the same problem as Example 14.1.

@author: Needh
"""
import numpy as np

def function(x,y):
    z = y - x -2*x**2 - 2*x*y - y**2
    return z

#Golden search subroutine
def randomSearch(gridIncrement,xLower,xUpper,yLower,yUpper):
    listOfPoints = []
    xStep = np.arange(xLower,xUpper,gridIncrement)
    yStep = np.arange(yLower,yUpper,gridIncrement)

    for x in xStep:
        for y in yStep:
            listOfPoints.append([function(x,y),x,y])
    index = listOfPoints.index(max(listOfPoints))
    domainMax = listOfPoints[index][0]
    xValue = listOfPoints[index][1]
    yValue = listOfPoints[index][2]

    print("The max of the set")
    print("y - x -2*x^2 - 2*x*y - y^2")
    print("bounded by ",xLower,"<=x<=",xUpper," and ",yLower, "<=y<=",yUpper,\
          " with a grid increment of ", gridIncrement, " is:",sep="")
    print(round(domainMax,3)," located at x=",xValue,", y=",yValue,sep="")

randomSearch(.005,-2,2,1,3)
```

```
OUTPUT
The max of the set
y - x -2*x^2 - 2*x*y - y^2
bounded by -2<=x<=2 and 1<=y<=3 with a grid increment
of 0.005 is:
1.25 located at x=-1.0, y=1.5
```

Numerical Methods

Homework #4 Problem #6

Find the gradient & Hessian for the following functions.

A) $f(x,y) = \ln(x^2 + 3xy + 2y^2)$

$$\nabla f(x,y) = \begin{bmatrix} \dfrac{2x + 3y}{x^2 + 3xy + 2y^2} \\[2ex] \dfrac{3x + 4y}{x^2 + 3xy + 2y^2} \end{bmatrix}$$

$$\text{Hessian } f(x,y) = \begin{vmatrix} \dfrac{-2x^2 + 6xy + 5y^2}{(x^2 + 3xy + 2y^2)^2} & \dfrac{-2x + 3y}{(x^2 + 3yx + 2y^2)^2} \\[2ex] \dfrac{-2x + 3y}{(x^2 + 3yx + 2y^2)^2} & \dfrac{-5x^2 + 12xy + 8y^2}{(x^2 + 3xy + 2y^2)^2} \end{vmatrix}$$

$$= \frac{(-2x^2 + 6xy + 5y^2)(-5x^2 + 12xy + 8y^2)}{(x^2 + 3xy + 2y^2)^4} - \frac{(2x + 3y)^2}{(x^2 + 3yx + 2y^2)^4}$$

B) $f(x,y,z) = x^2 + y^2 + 3z^2$

$$\nabla f(x,y,z) = \begin{bmatrix} 2x \\ 2y \\ 6z \end{bmatrix} \qquad \text{Hessian} = \begin{vmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{vmatrix} = \underline{\underline{24}}$$

Homework #4 Problem # 7

A) Start w/ an initial guess of $(x_0, y_0) = (1,1)$ and apply two iterations of the steepest ascent method to maximize $f(x,y)$

$$f(x,y) = 2xy + 2y - 1.5x^2 - 2y^2$$

$$\nabla f(x,y) = \begin{bmatrix} 2y - 3x \\ 2x + 2 - 4y \end{bmatrix}$$

$$g(h) = f(1-h, 1)$$

$$= 2(1-h) + 2 - 1.5(1-h)^2 - 2(1)^2$$

$$= .5h - 1.5h^2$$

$$\nabla f(x,y) = \begin{bmatrix} -1 \\ 0 \end{bmatrix}$$

$$g'(h) = -3h + 1 \qquad h^* = 1/3$$

$$(x_1, y_1) = (x_0, y_0) + h \nabla f(x_0, y_0)$$

$$= (1,1) + \tfrac{1}{3}(-1, 0)$$

Iteration #1 $\rightarrow$ $= (2/3, 1)$

$$g(h) = f((\tfrac{2}{3}, 1) + h(0, 2/3))$$

$$= f(\tfrac{2}{3}, 1 - \tfrac{2}{3}h)$$

$$= 2(\tfrac{2}{3})(1 - \tfrac{2}{3}h) + 2(1 - \tfrac{2}{3}h) - 1.5(\tfrac{2}{3})^2 - 2(1 - \tfrac{2}{3}h)^2$$

$$= \frac{-4(h + .5)(2 \cdot h - 3)}{9}$$

$$g'(h) = .\bar{8} - 1.\bar{7}h \qquad h^* = \frac{1}{2}$$

$$(x_2, y_2) = (x_1, y_1) + h \nabla f(x_1, y_1)$$

$$= (2/3, 1) + \tfrac{1}{2}(0, -2/3)$$

Iteration # 2 $\rightarrow$ $= (2/3, \tfrac{2}{3})$

B)

$f(x,y) = 2xy + 2y - 1.5x^2 - 2y^2$

$$\nabla f(x,y) = \begin{bmatrix} 2y - 3x \\ 2x - 4y + 2 \end{bmatrix}$$

$\frac{\partial^2 f}{\partial x^2} = -3$

$\frac{\partial^2 f}{\partial x \partial y} = 2$

$\frac{\partial^2 f}{\partial y^2} = -4$

$$\text{Hessian } f(x,y) = \begin{vmatrix} -3 & -2 \\ 2 & -4 \end{vmatrix} = 8$$

Because $|H|$ is > 0 & $\frac{\partial^2 f}{\partial x^2} < 0$, the value is a max

Solving for max point:

$2y - 3x = 0$ $\quad x = 1/2$

$2x - 4y + 2 = 0$ $\quad y = 3/4$