**Student: Toktarov Nurislam**

**Group: SE-2302**

Task 2

import math  
a = -2  
b = 0  
c = (a+b)/2  
TOL = 0.000000001  
NMAX = 15  
  
  
def f(x):  
 return math.exp(x)-x\*\*2  
if f(c)!=0:  
 i = 0  
 while math.fabs(f(c))> TOL and i<=NMAX:  
 if f(c)\*f(a)>0:  
 a=c  
 b=b  
 else:  
 a=a  
 b=c  
 c = (a + b) / 2  
 print("f(c) = " + str(f(c))+ "; c = " + str(c))  
 i += 1  
else:  
 print(f(c)+'=0, c is root')

Task 3,4,5

import math  
  
TOL = 0.000000001  
NMAX = 15  
  
def bisection\_method(f, a, b, tol, nmax):  
   
 c = (a + b) / 2  
 iterations = []  
  
 if f(c) != 0:  
 i = 0  
 while math.fabs(f(c)) > tol and i <= nmax:  
 if f(c) \* f(a) > 0:  
 a = c  
 else:  
 b = c  
 c = (a + b) / 2  
 iterations.append((i + 1, c, f(c)))  
 i += 1  
 return iterations  
 else:  
 return [(0, c, f(c))]  
  
def secant\_method(f, x0, x1, tol, nmax):  
   
 iterations = []  
 for i in range(nmax):  
 if abs(f(x1)) < tol:  
 break  
 x2 = x1 - f(x1) \* (x1 - x0) / (f(x1) - f(x0))  
 iterations.append((i + 1, x2, f(x2)))  
 x0, x1 = x1, x2  
 return iterations  
  
def newton\_raphson\_method(f, df, x0, tol, nmax):  
   
 iterations = []  
 x = x0  
 for i in range(nmax):  
 fx = f(x)  
 dfx = df(x)  
 if abs(fx) < tol:  
 break  
 if dfx == 0:  
 raise ZeroDivisionError("Derivative is zero. No solution found.")  
 x\_new = x - fx / dfx  
 iterations.append((i + 1, x\_new, f(x\_new)))  
 x = x\_new  
 return iterations  
  
def fixed\_point\_iteration(g, x0, tol, nmax):  
   
 iterations = []  
 x = x0  
 for i in range(nmax):  
 x\_new = g(x)  
 iterations.append((i + 1, x\_new, abs(x\_new - x)))  
 if abs(x\_new - x) < tol:  
 break  
 x = x\_new  
 return iterations  
  
functions = {  
 "1": lambda x: math.exp(x) - x\*\*2,  
 "2": lambda x: math.exp(x) - math.cos(x),  
 "3": lambda x: math.exp(x) - x,  
 "4": lambda x: math.exp(x) + x + 7,  
 "5": lambda x: math.exp(x) + 4 \* math.sin(x),  
 "6": lambda x: math.cos(x) - x \* math.exp(x)  
}  
  
derivatives = {  
 "1": lambda x: math.exp(x) - 2 \* x,  
 "2": lambda x: math.exp(x) + math.sin(x),  
 "3": lambda x: math.exp(x) - 1,  
 "4": lambda x: math.exp(x) + 1,  
 "5": lambda x: math.exp(x) + 4 \* math.cos(x),  
 "6": lambda x: -math.sin(x) - math.exp(x) - x \* math.exp(x)  
}  
  
initial\_guesses = {  
 "1": (-2, 0),  
 "2": (0, 1),  
 "3": (0, 1),  
 "4": (-10, -5),  
 "5": (-2, 2),  
 "6": (0.5, 1)  
}  
  
results = {}  
for key, f in functions.items():  
 print(f"\nFunction {key} Results:")  
 a, b = initial\_guesses[key]  
  
 print("\nBisection Method:")  
 bisection\_results = bisection\_method(f, a, b, TOL, NMAX)  
 for i, c, fc in bisection\_results:  
 print(f"Iteration {i}: c = {c:.6f}, f(c) = {fc:.6e}")  
  
 print("\nSecant Method:")  
 x0, x1 = a, b  
 secant\_results = secant\_method(f, x0, x1, TOL, NMAX)  
 for i, x, fx in secant\_results:  
 print(f"Iteration {i}: x = {x:.6f}, f(x) = {fx:.6e}")  
  
 if key in derivatives:  
 print("\nNewton-Raphson Method:")  
 df = derivatives[key]  
 x0 = (a + b) / 2  
 newton\_results = newton\_raphson\_method(f, df, x0, TOL, NMAX)  
 for i, x, fx in newton\_results:  
 print(f"Iteration {i}: x = {x:.6f}, f(x) = {fx:.6e}")