Lab-1

September 12, 2023

[ ]:
```
C201050
AYAT ULLAH
7BM
```

[ ]:
```
# Write a program to count number of significant digits in a given number.
```

[16]:
```
def int_str(data):
    num_int = int(data)
    num_str = str(num_int)
    return num_str

def significant(number):
    if "." in number:
        before_dot,after_dot = number.split(".")
        before_int = int(before_dot)

        if before_int==0:
            return len(int_str(after_dot))
        else:

            return len(int_str(before_dot))+len(after_dot)

        return before_int
    else:
        num_str = int_str(number)
        reverse_str = "".join(reversed(num_str))
        num_str = int_str(reverse_str)
        return len(num_str)

number = input("Enter a number: ")
print(f'significant bit is {significant(number)}')
```

```
Enter a number: 10
significant bit is 1
```

[ ]:
```
# Write a program to round off a number with n significant figures using
 ↪banker's rule.
```

[51]:

```
def printSignificant(num,n,x):
    output=""
    if x==10:
        old = num
        newnum = str(int("".join(num.split("."))[:n])+1)
        pos=0
        for i in newnum:
            if old[pos]==".":
                output+="."
                output+=i
            else:
                output+=i
            pos=pos+1
        print(output)

    else:
        for i in range(n):
            output+=num[i]
        print(f'{output}{x}')
num =input("Enter the number: ");
n = int(input("Enter the signigicant bits: "))
if n>=len(num)-1:
    print(num)
else:
    x = int(num[n])
    x1 = int(num[n+1])
    #print(f'x {x} x1 {x1}')
    if x1<5:
        printSignificant(num,n,x)
    elif x1>5:
        x=x+1
        printSignificant(num,n,x)
    elif x1==5 and x%2==1:
        x=x+1
        printSignificant(num,n,x)
    else:
        printSignificant(num,n,x)
```

```
Enter the number: 3.1596
Enter the signigicant bits: 4
3.160
```

[ ]:

```
```

[27]:

```
# Write a program to evaluate a polynomial f(x) = x^3 - 2x^2 + 5x + 10 by using
 ↪Horner's
# rule x = 5.
```

[52]:
```python
nums = [1,-2,5,10]
x=5
p = []

for i in range(len(nums)):
    if(i==0):
        p.append(nums[i]);
    else:
        res = p[i-1]*x + nums[i];
        p.append(res)
print(p[-1])
```

110

[31]:
```python
# Write a program to find the root of the equation x 3 - 9x + 1 = 0, correct to
 ↪3 decimal
# places, by using the bisection method.
```

[2]:
```python
def fun(x):
     return pow(x,3)-(9*x)+1



a=1
b=3
c = (a+b)/2
if (fun(a)*fun(b)) < 0:
    while abs(a-b)>0.005:
        c = (a+b)/2;
        fc = fun(c);
        fa = fun(a);
        if fa * fc < 0:
            b=c;
        elif fc==0:
            break;
        else:
            a=c;

else:
    a = int(input('a= '))
    b = int(input('b= '))
    while a-b>=0.005:
        c = (a+b)/2
        fc = fun(c)
        fa = fun(a)
        if fa * fc < 0:
            b=c
        else:
```

```
            a=c
print(f'{(a+b)/2:.3f}')
```

2.943

[ ]:

```

```

[1]:

```
# Write a program to find the root of the equation x 5 + 3x 2 - 10 = 0, correct␣
↪to 3 decimal places, by
# using fixed point method.
```

[3]:

```
import math
# x^5+3x^2-10 = 0
# x = sqrt(10/(x^3+3))
def g_of_x(x):
    return math.sqrt((10/(pow(x,3)+3)))
def fix_point():
    initial = 1
    xnot = g_of_x(initial)
    for i in range(50):
        initial = xnot
        xnot = g_of_x(xnot)
        if abs(initial-xnot)<0.001:
            print(f'{xnot:.3f}');
            break;
fix_point()
```

1.352

[45]:

```
# Write a program to find the root of the equation x 3 - 6x + 4 = 0, correct to␣
↪3 decimal places, by using
# Newton- Raphson method.
```

[4]:

```
def f_of_x(x):
    return pow(x,3)-6*x+4
def f_prime_x(x):
    return 3*pow(x,2)-6
def newton_raphson():
    initial = 0
    for i in range(100):
        x_n = initial - (f_of_x(initial)/f_prime_x(initial))

        if abs(initial-x_n)<0.001:
            print(f'{x_n:.3f}')
            return
        initial = x_n
newton_raphson()
```

0.732

[ ]:
```
# Write a program to find the root of the equation x 3 - x + 2 = 0, correct to␣
↪3 decimal places, by using
# false position method.
```

[11]:
```
def f_of_x(x):
    return pow(x,3)-x+2

def search_bra():
    return math.sqrt(pow(0/1,2)-2*((-1)/(1)))
def false_position(x1,x2):
    return x1 - ((f_of_x(x1)*(x2-x1))/(f_of_x(x2)-f_of_x(x1)))

x_1  = -2
x_2 = 2

for i in range(10):
    x_0 = false_position(x_1,x_2);
    fx0 = f_of_x(x_0)
    if f_of_x(x_0)*f_of_x(x_1)<0:
        x_2 = x_0
    else:
        x_1 = x_0
    if i>0 and abs(prev-x_0)<0.001:
        print(f'{x_0:.3f}')
        break
    prev = x_0
```

```
-1.521
```

[ ]:
```
# Write a program to find the root of the equation x 3 - 5x 2 - 29 = 0, correct␣
↪to 3 decimal places, by using
# secant method.
```

[15]:
```
def fun(x):
    return pow(x,3)-5*pow(x,2)-29
def secant_method(x1,x2):
    return  x1 -( (fun(x1)*(x1-x2))/(fun(x1)-fun(x2)) )
x1  = 2
x2  = 4
f1  = fun(x1)
f2  = fun(x2)
for i in range(100):
```

```
    x3 = secant_method(x1,x2)
    if abs(x3-x2)<0.001:
        print(f'{x3:.3f}')
        break
    else:
        x1=x2
        f1=f2
        x2=x3
        f2=fun(x3)
```

```
5.848
```

[ ]:
```
# Write a program to find the quotient polynomial q(x) such that p(x) = (x-2)␣
 ↪q(x) where the
# polynomial p(x) = x 3 - 5x 2 + 10x - 8 = 0 has a root at x = 2.
```

[54]:
```
import math
fmax = abs(math.sqrt(abs(0/1)**2 - 2*(-6/1)))
fmax = int(fmax)
print(-fmax, fmax)
```

```
-3 3
```