Project 5: Float Analysis

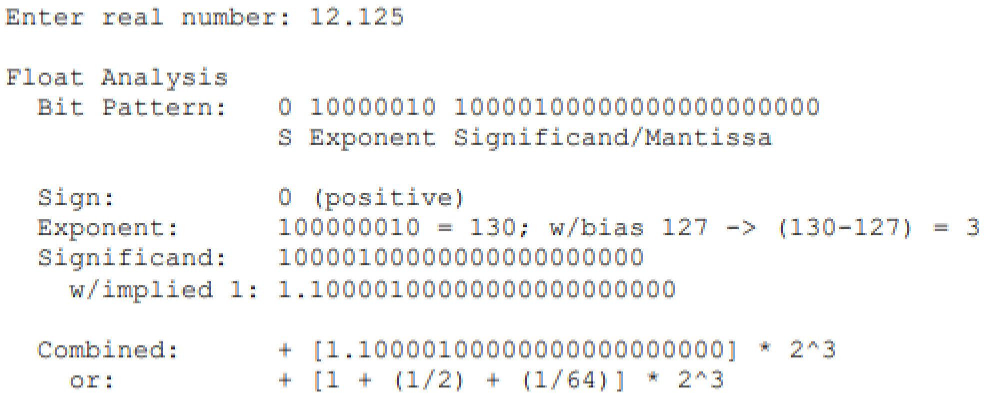
Alenn Wright
October 6th, 2021
CS 200

Introduction

For this project we are tasked with making a program that takes in floating point numbers and displays a bit-level analysis. The information output should be the Bit pattern, the sign, exponent, significand, significand with the implied 1, a combined display, and the fractional display. The code should also be able to identify special cases such as negative zero. This code is not to convert the number but to analyze it.

Solution

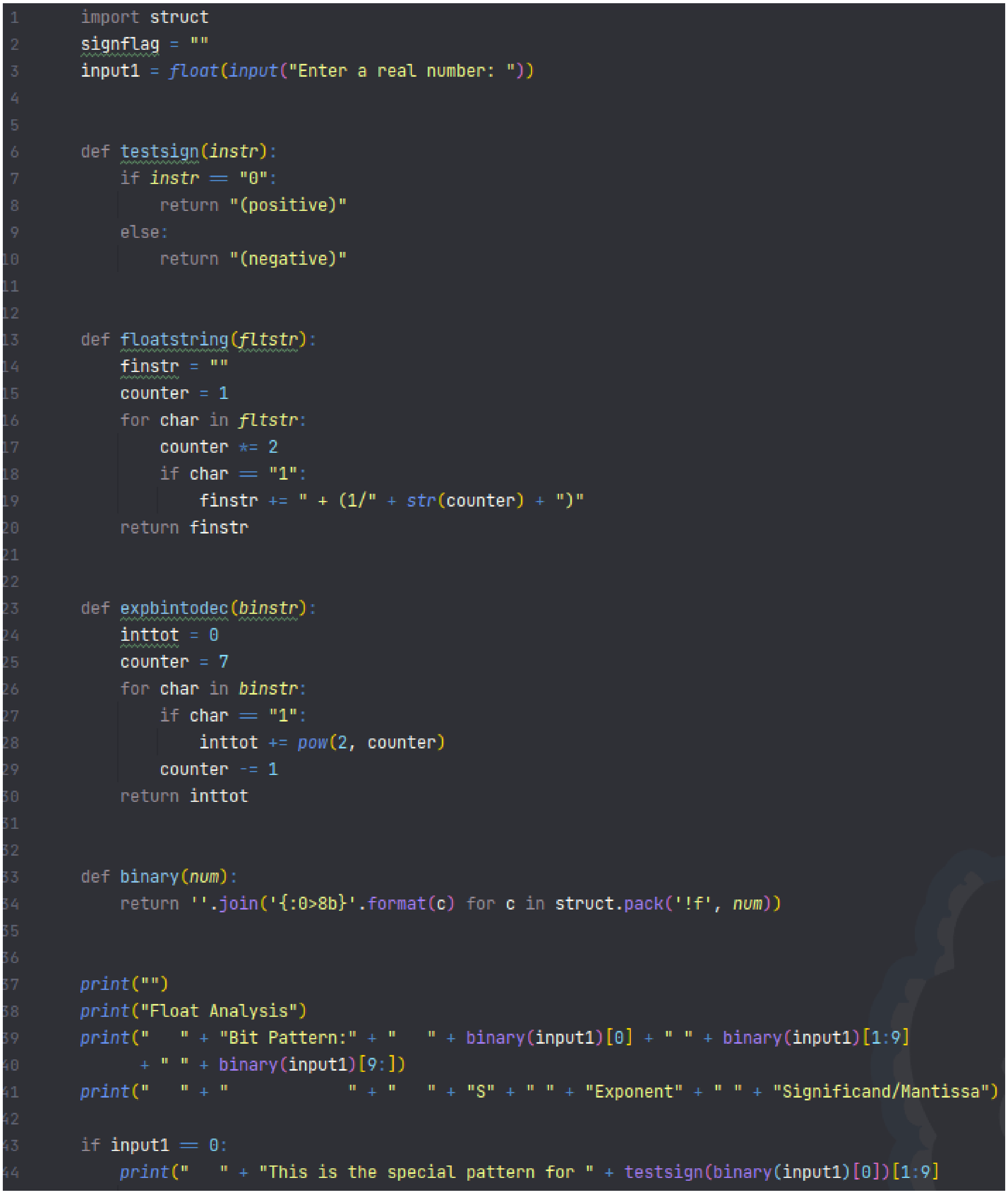
For the solution, I took the sample output and recreated it. To reduce the hard coded switch cases I made methods to test the sign of the float, a method to convert the exponent bits to a decimal, and a method to make the fractional summation string.



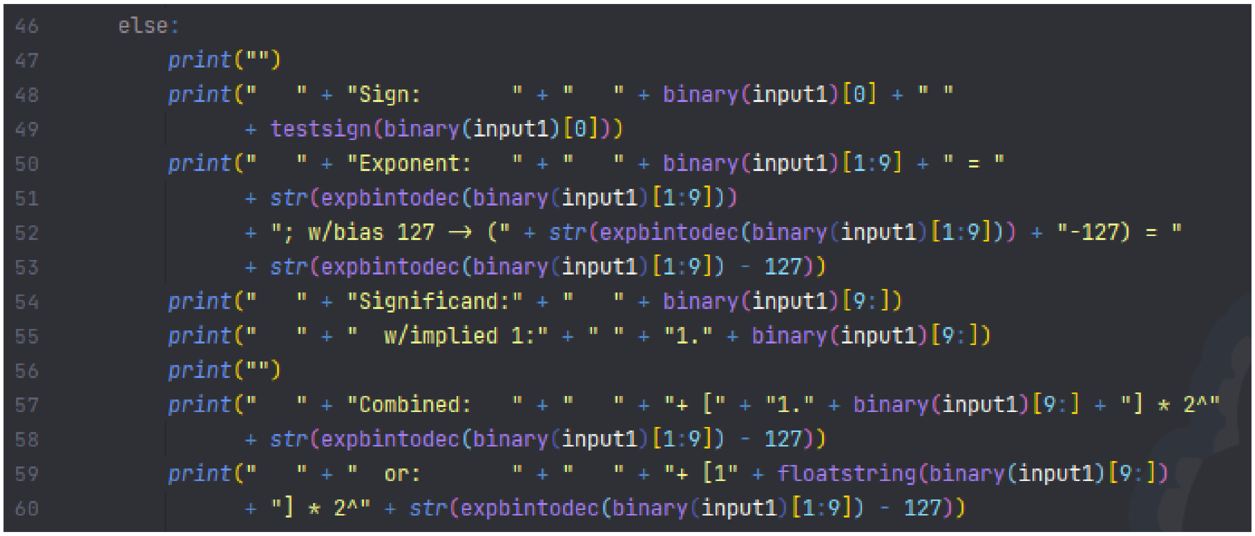
```
Enter real number: 12.125

Float Analysis
  Bit Pattern:   0 10000010 10000100000000000000000
                 S Exponent Significand/Mantissa

  Sign:          0 (positive)
  Exponent:      100000010 = 130; w/bias 127 -> (130-127) = 3
  Significand:   10000100000000000000000
    w/implied 1: 1.10000100000000000000000

  Combined:      + [1.10000100000000000000000] * 2^3
    or:          + [1 + (1/2) + (1/64)] * 2^3
```

Figure is the sample output given to us.

Source Code



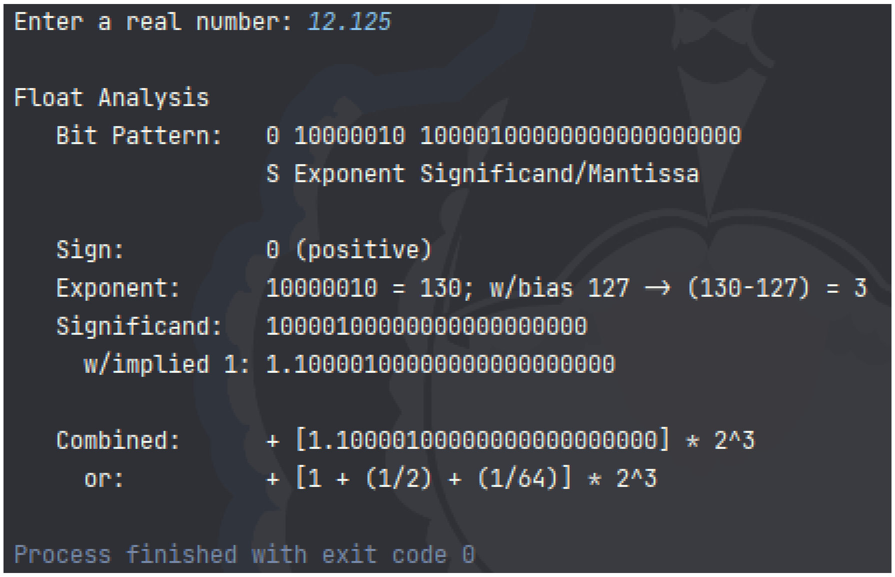
```python
import struct
signflag = ""
input1 = float(input("Enter a real number: "))


def testsign(instr):
    if instr == "0":
        return "(positive)"
    else:
        return "(negative)"


def floatstring(fltstr):
    finstr = ""
    counter = 1
    for char in fltstr:
        counter *= 2
        if char == "1":
            finstr += " + (1/" + str(counter) + ")"
    return finstr


def expbintodec(binstr):
    inttot = 0
    counter = 7
    for char in binstr:
        if char == "1":
            inttot += pow(2, counter)
        counter -= 1
    return inttot


def binary(num):
    return ''.join('{:0>8b}'.format(c) for c in struct.pack('!f', num))


print("")
print("Float Analysis")
print("   " + "Bit Pattern:" + "   " + binary(input1)[0] + " " + binary(input1)[1:9]
      + " " + binary(input1)[9:])
print("   " + "            " + "   " + "S" + " " + "Exponent" + " " + "Significand/Mantissa")

if input1 == 0:
    print("   " + "This is the special pattern for " + testsign(binary(input1)[0])[1:9]
```



```
    else:
        print("")
        print("   " + "Sign:        " + "    " + binary(input1)[0] + " "
              + testsign(binary(input1)[0]))
        print("   " + "Exponent:    " + "    " + binary(input1)[1:9] + " = "
              + str(expbintodec(binary(input1)[1:9]))
              + "; w/bias 127 → (" + str(expbintodec(binary(input1)[1:9])) + "-127) = "
              + str(expbintodec(binary(input1)[1:9]) - 127))
        print("   " + "Significand:" + "    " + binary(input1)[9:])
        print("   " + "  w/implied 1:" + " " + "1." + binary(input1)[9:])
        print("")
        print("   " + "Combined:    " + "    " + "+ [" + "1." + binary(input1)[9:] + "] * 2^"
              + str(expbintodec(binary(input1)[1:9]) - 127))
        print("   " + "  or:        " + "    " + "+ [1" + floatstring(binary(input1)[9:])
              + "] * 2^" + str(expbintodec(binary(input1)[1:9]) - 127))
```

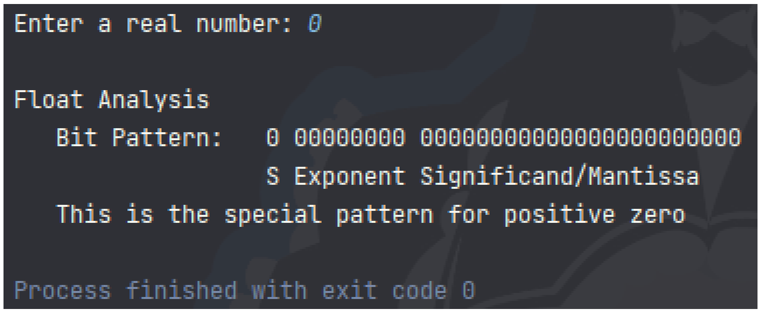
Test Value outputs

- 12.125



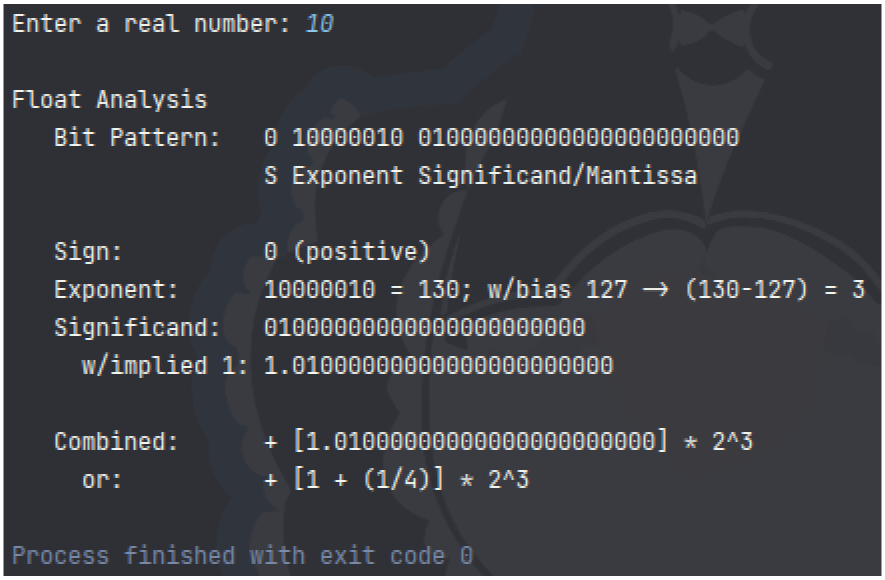
```
Enter a real number: 12.125

Float Analysis
   Bit Pattern:   0 10000010 10000100000000000000000
                  S Exponent Significand/Mantissa


   Sign:          0 (positive)
   Exponent:      10000010 = 130; w/bias 127 → (130-127) = 3
   Significand:   10000100000000000000000
     w/implied 1: 1.10000100000000000000000

   Combined:      + [1.10000100000000000000000] * 2^3
     or:          + [1 + (1/2) + (1/64)] * 2^3

Process finished with exit code 0
```

- 0



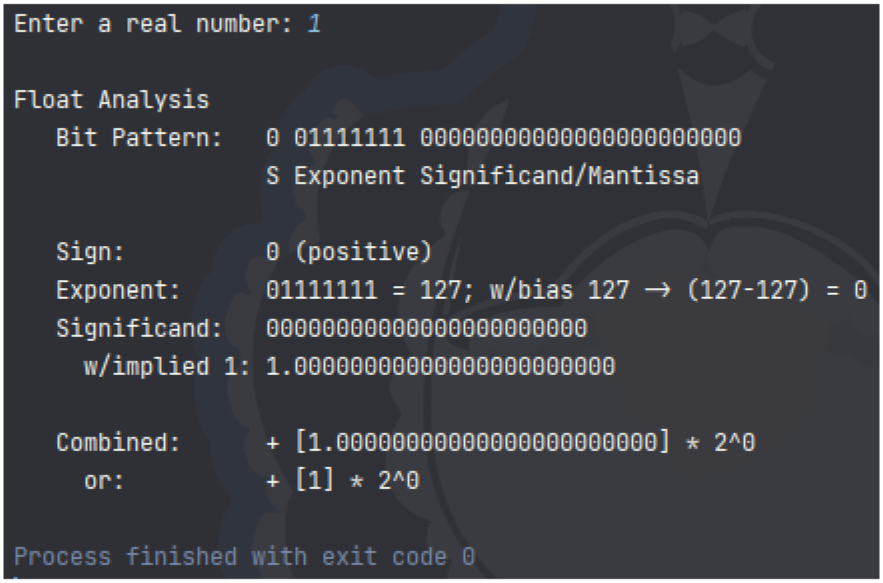
```
Enter a real number: 0

Float Analysis
   Bit Pattern:   0 00000000 00000000000000000000000
                  S Exponent Significand/Mantissa
   This is the special pattern for positive zero

Process finished with exit code 0
```

- 10.0



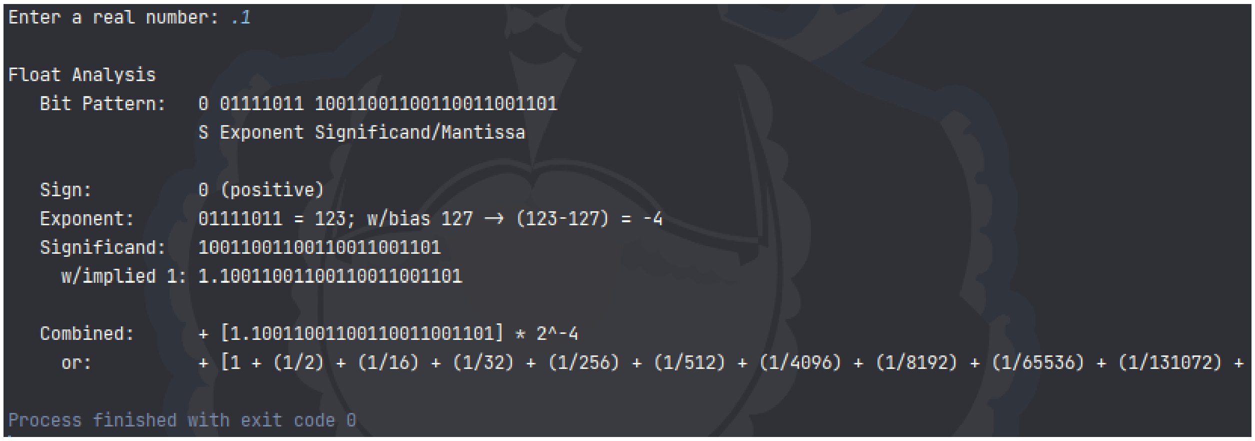
```
Enter a real number: 10

Float Analysis
   Bit Pattern:    0 10000010 01000000000000000000000
                   S Exponent Significand/Mantissa


   Sign:           0 (positive)
   Exponent:       10000010 = 130; w/bias 127 → (130-127) = 3
   Significand:    01000000000000000000000
     w/implied 1: 1.01000000000000000000000

   Combined:       + [1.01000000000000000000000] * 2^3
     or:           + [1 + (1/4)] * 2^3

Process finished with exit code 0
```

- 1.0



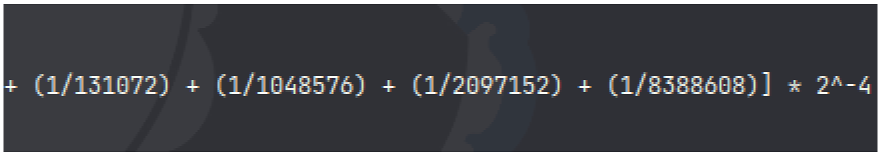
```
Enter a real number: 1

Float Analysis
   Bit Pattern:    0 01111111 00000000000000000000000
                   S Exponent Significand/Mantissa


   Sign:           0 (positive)
   Exponent:       01111111 = 127; w/bias 127 → (127-127) = 0
   Significand:    00000000000000000000000
     w/implied 1: 1.00000000000000000000000

   Combined:       + [1.00000000000000000000000] * 2^0
     or:           + [1] * 2^0

Process finished with exit code 0
```

- 0.1



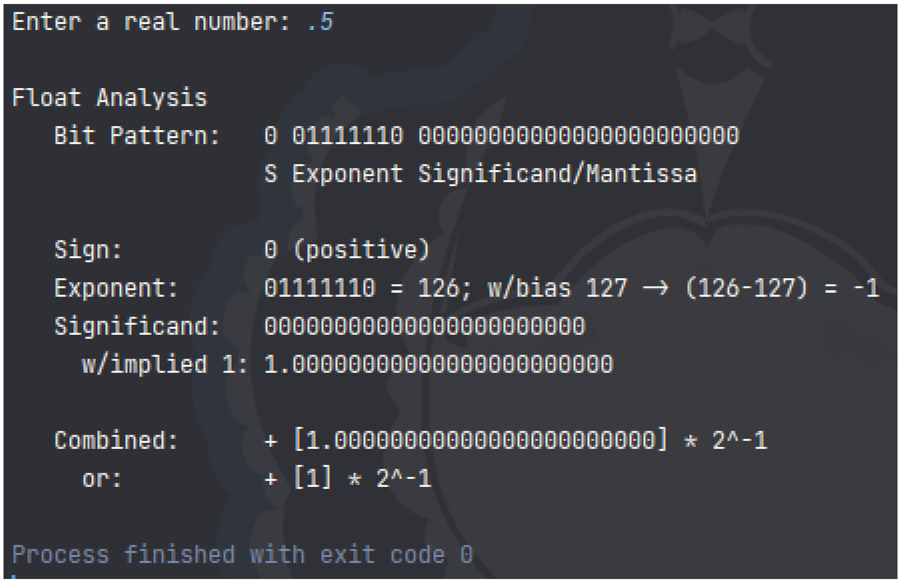
```
Enter a real number: .1

Float Analysis
   Bit Pattern:    0 01111011 10011001100110011001101
                   S Exponent Significand/Mantissa


   Sign:           0 (positive)
   Exponent:       01111011 = 123; w/bias 127 → (123-127) = -4
   Significand:    10011001100110011001101
     w/implied 1: 1.10011001100110011001101

   Combined:       + [1.10011001100110011001101] * 2^-4
     or:           + [1 + (1/2) + (1/16) + (1/32) + (1/256) + (1/512) + (1/4096) + (1/8192) + (1/65536) + (1/131072) +

Process finished with exit code 0
```

-



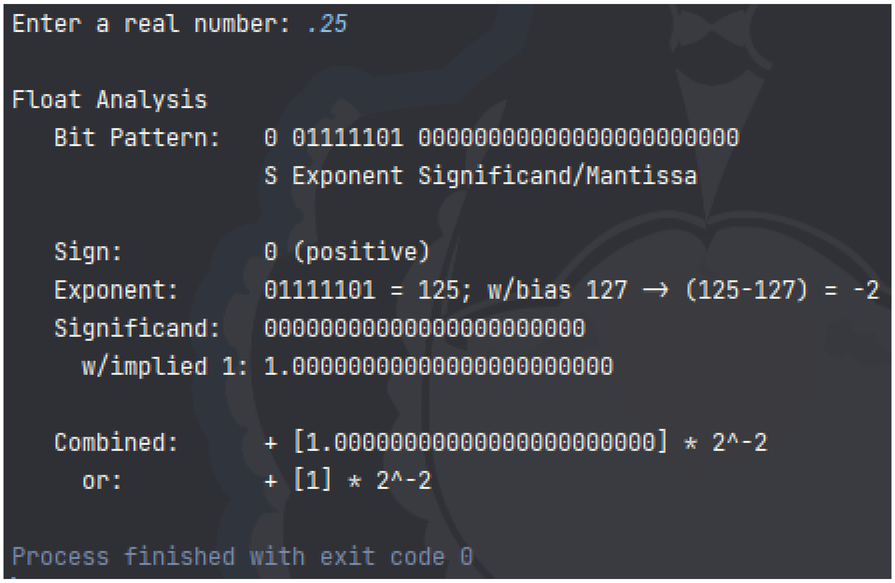
```
+ (1/131072) + (1/1048576) + (1/2097152) + (1/8388608)] * 2^-4
```

- 0.5



```
Enter a real number: .5

Float Analysis
   Bit Pattern:   0 01111110 00000000000000000000000
                  S Exponent Significand/Mantissa

   Sign:          0 (positive)
   Exponent:      01111110 = 126; w/bias 127 → (126-127) = -1
   Significand:   00000000000000000000000
     w/implied 1: 1.00000000000000000000000

   Combined:      + [1.00000000000000000000000] * 2^-1
     or:          + [1] * 2^-1

Process finished with exit code 0
```

- 0.25



```
Enter a real number: .25

Float Analysis
   Bit Pattern:   0 01111101 00000000000000000000000
                  S Exponent Significand/Mantissa

   Sign:          0 (positive)
   Exponent:      01111101 = 125; w/bias 127 → (125-127) = -2
   Significand:   00000000000000000000000
     w/implied 1: 1.00000000000000000000000

   Combined:      + [1.00000000000000000000000] * 2^-2
     or:          + [1] * 2^-2

Process finished with exit code 0
```

Conclusion

In conclusion, this project was fun and I learned alot from pythons built in binary interpretation of numbers. At first I was worried about separating the string to understand the sections but python defaults to a 32 bit representation so the same sections of bits are in the same spot in the string every time. So I parsed it to interpret the sections. This program also seems like it will be useful for me to play with later.