# Big Program - Illustrating The Binary Representation of a Floating Point Number

Things we will need:

1. knowledge of int, float, string.

2. knowledge of functions, both ones that we write, and built-in functions that are part of the int, float, and string classes.

3. knowledge of how floating point #'s are stored in Python.

4. Loops (for, while)

5. Branching (If-else)

(6) Knowledge of how to get input from the user.

(7) Knowledge of how to output information in a formatted way. (print)

## Program Design.

**Step 1:** Get a floating point # from the user.

```
n = input("Enter your floating point value : \n")
```

→ note: n will be a string !!!

**Step 2:** Get how many decimal ... we want for the places we want,

binary "decimal" represent-ation

i.e. $0.1_{10} = 0.00011001100011\ldots\ldots$

How many ???

p = int(input("Enter the number of decimal places of The result : \n"))

→ note: p will be an int !!!

Step 3 : Write a function which takes n, p as inputs, and returns the binary "decimal" representation as a string.

What is the algorithm that this function will use?? To figure this out, let's imagine doing the steps involved.

by hand, and the

it by ...

Ex. 3.125, to 6 decimal places.

Step 1: Separate the two parts, before and after the decimal.

$$3 + 0.125$$

Step 2: Convert the part in front of the decimal to binary.

$3 \rightarrow 11$ (Multiple steps here!!)

Step 3: Convert the part after the decimal to binary.

six places.

$0.125 \rightarrow 0.\boxed{001000}00.....$

$\frac{1}{2}$ $\frac{1}{4}$ $\frac{1}{8}$

(Multiple steps!!)

Step 4: Construct a string out of these results.

res = "11" + "." + "001000"

11

$$3.125_{10} = 11.001000_2$$ to 6 decimal places.

## Step 2, broken down

$$3_{10} \rightarrow 11_2$$

**Algorithm:** Use integer division + mod function.

Ex. $13_{10} = 1101_2$

| | | | | |
|---|---|---|---|---|
| | 1 | 1 | 0 | 1 |

$$13 / 2 = 6$$
$$13 \% 2 = 1$$

---

$$6 / 2 = 3$$
$$6 \% 2 = 0$$

---

$3 / 2 = 1$
$3 \% 2 = 1$

$1 / 2 = \boxed{0}$ Done
$1 \% 2 = 1$

```
num = 13
res = ""
while (num > 0):
    digit = num % 2
    res = str(digit) + res
    num = num / 2

print(res)
```

Now that we know how to do this, let's save it as a function:

```
def convert_to_binary(num):
```

return res

---

Step 3, broken down →

0.125 → $\underbrace{001000}_{\text{6 places.}}$

Again, let's do it by hand:

$0.95_{10}$

if (num > 0.5) $\xrightarrow{Y}$ $\boxed{1}$ → .95 - .5 = .45

$\xrightarrow{N}$ 0 → do nothing.

$0.45_{10}$

if (num > 0.25 $\xrightarrow{Y}$ $\boxed{1}$ → .45 - .25 = .20

$\xrightarrow{N}$ 0 → do nothing.

$0.20_{10}$

if (num > 0.125 $\xrightarrow{Y}$ $\boxed{1}$ → .20 - .125 = .075

$\xrightarrow{N}$ 0 → do nothing.

$\boxed{1}$ → .075 - .0625 = .0125

$0.075_{10}$

if (num > 0.0625 $\xrightarrow{Y}$ $\boxed{1}$

if (num > 0.0625 $\xrightarrow{N}$ 0 → do nothing.

⋮

---

```
num = 0.95
p = 6

count = 0
res = ""
while num > 0 and count < p :
        comparison = 2 ** (-1 - count)
        if (num > comparison) :
                res = res + "1"
                num = num - comparison
        else:
                res = res + "0"

        count = count + 1

print (res)
```

---

Again, now that we have this bit working, write it as a function!!

```
def convert_fraction_to_binary(num, p):
    }
    }
    return res
```

Now, put steps 1, 2, 3, 4 together....

See get_binary_rep.py in Week 1 examples.

OK, now we have the fractional binary representation of a real floating point decimal number.

Next thing: Get the 32 bit floating point binary

32 bit floating p

representation !!

$$1.\overbrace{0011011}^{\text{mantissa}} \times 2^{\overbrace{101}^{\text{exponent}}}$$

Step 5 : Determine the mantissa and exponent.