

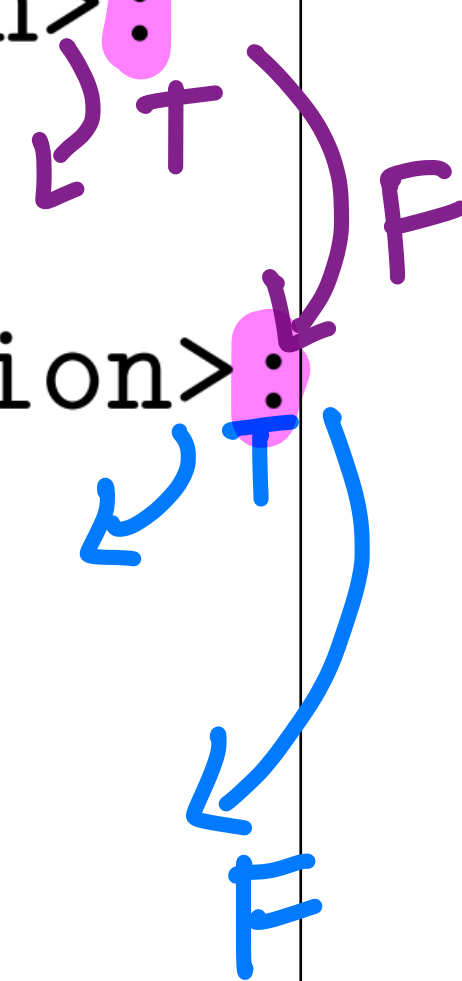
CS 61A

Discussion 1

yellkey.com/serve

Control (+ Boolean Operators)

```
if <conditional expression>:  
    <suite of statements>  
elif <conditional expression>:  
    <suite of statements>  
else:  
    <suite of statements>
```



```
>>> not None  
True  
>>> not True  
False  
>>> -1 and 0 and 1  
0  
>>> False or 9999 or 1/0  
9999
```

Truthy/Falsy
Boolean: True/False

not
False → True
0 → True

0 + 1 = 1
False + 1 = Error

True
False
True

True

Short - circuiting

$f(n)$ # $n=3 \rightarrow \text{Error}$

$n==3$ ~~or $f(n)$~~
True

Numbers

all numbers but 0 → True
0 → False

Strings

"hello" → True
"" → False

Q1: Case Conundrum

What is the result of evaluating the following code?

```
def special_case():  
    x = 10  
    if x > 0:  
        x += 2  
    elif x < 13:  
        x += 3  
    elif x % 2 == 1:  
        x += 4  
    return x
```

special_case()

```
def just_in_case():  
    x = 10  
    if x > 0:  
        x += 2  
    if x < 13:  
        x += 3  
    if x % 2 == 1:  
        x += 4  
    return x
```

just_in_case()

```
def case_in_point():  
    x = 10  
    if x > 0:  
        return x + 2  
    if x < 13:  
        return x + 3  
    if x % 2 == 1:  
        return x + 4  
    return x
```

case_in_point()

Q1: Case Conundrum

Which of these code snippets result in the same output, and why? When do you think using a series of if statements has the same effect as using both if and elif cases?

```
def special_case():  
    x = 10  
    if x > 0:  
        x += 2  
    elif x < 13:  
        x += 3  
    elif x % 2 == 1:  
        x += 4  
    return x
```

special_case()

```
def just_in_case():  
    x = 10  
    if x > 0:  
        x += 2  
    if x < 13:  
        x += 3  
    if x % 2 == 1:  
        x += 4  
    return x
```

just_in_case()

```
def case_in_point():  
    x = 10  
    if x > 0:  
        return x + 2  
    if x < 13:  
        return x + 3  
    if x % 2 == 1:  
        return x + 4  
    return x
```

case_in_point()

Q2: Jacket Weather

Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining.

Write a function that takes in the current temperature and a boolean value telling if it is raining. This function should return True if Alfonso will wear a jacket and False otherwise.

Try solving this problem using an if statement.

Q2: Jacket Weather

Alfonso will only wear a jacket outside if it is below 60 degrees or it is raining. Write a function that takes in the current temperature and a boolean value telling if it is raining.

11:38

" : "

```
def wears_jacket_with_if(temp, raining):
```

```
    """
```

```
    >>> wears_jacket_with_if(90, False)
```

```
    False
```

```
    >>> wears_jacket_with_if(40, False)
```

```
    True
```

```
    >>> wears_jacket_with_if(100, True)
```

```
    True
```

```
    """
```

```
    "*** YOUR CODE HERE ***"
```

```
    if temp < 60 or raining:
```

```
        return True
```

```
    else:
```

```
        return False
```

or True } or 999
or False } or 10

While loops

```
while <conditional clause>:  
    <statements body>
```

this is true

do this

Q1: Case Conundrum

What is the result of evaluating the following code?

```
def square(x):  
    print("here!")  
    return x * x
```

```
def so_slow(num):  
    x = num x=5  
    while x > 0:  
        x = x + 1  
    return x / 0
```

```
square(so_slow(5))
```

5 > 0 ✓
x+1 → x=6
6 > 0 ✓
x+1 → x=7
7 > 0 ✓
...

Q5: Is Prime?

Write a function that returns True if a positive integer n is a prime number and False otherwise.

A prime number n is a number that is not divisible by any numbers other than 1 and n itself. For example, 13 is prime, since it is only divisible by 1 and 13, but 14 is not, since it is divisible by 1, 2, 7, and 14.

Hint: Use the % operator: $x \% y$ returns the remainder of x when divided by y .

modulo

7 \div 4

1) 7 divided by 4 = 1 R3

2) take remainder $\rightarrow 3$

$4 \% 3 \rightarrow 4/3 = 1 \text{ R}1 \rightarrow 1$

$5 \% 3 \rightarrow 5/3 = 1 \text{ R}2 \rightarrow 2$

$6 \% 3 \rightarrow 6/3 = 2 \text{ R}0 \rightarrow 0$

Q5: Is Prime?

Write a function that returns True if a positive integer n is a prime number and False otherwise.

Hint: Use the % operator: x % y returns the remainder of x when divided by y.

12:04

$$6 \Rightarrow 3 \cdot 2$$

$6/2 = 3$

$$4 \times \cdot 2 = 8$$
$$5 \times \cdot 2 = 10$$

```
def is_prime(n):
```

```
    """
```

```
    >>> is_prime(10)
```

```
False
```

```
    >>> is_prime(7)
```

```
True
```

```
    """
```

```
    """ *** YOUR CODE HERE *** """
```

```
    div = 2
```

```
    while div < n
```

```
        if n % div == 0: # div is factor
            return False
```

```
        div += 1
```

```
    return True
```

$$10 \% 2 = 0 \rightarrow F$$

2
3
4
5
6

$$\begin{array}{l} \text{div} = 2 \\ \downarrow \\ \text{div} = n-1 \end{array}$$

Q6: FizzBuzz?

Implement the fizzbuzz sequence, which prints out a single statement for each number from 1 to n.

For a number i,

- If i is divisible by 3 only, then we print "fizz".
- If i is divisible by 5 only, then we print "buzz".
- If i is divisible by both 3 and 5, then we print "fizzbuzz".
- Otherwise, we print the number i by itself.

Q6: FizzBuzz?

Implement the fizzbuzz sequence, which prints out a single statement for each number from 1 to n.

```
def fizzbuzz(n):  
    """ *** YOUR CODE HERE *** """
```


Environment Diagrams

Frames: helps us keep track of what variables have been defined in the current execution environment, and what values they hold.

Global frame: The frame we start off with when executing a program from scratch

$x = 3$
 $y = 3 + 5$

Global frame	
variables	values
x	3
y	8

Env Diagram: Assignment Statement

Assignment Statements: Define variables in programs $x=3$

1. Evaluate the expression on the right side of the = sign $y=3+5$ $\#8$
2. Write the variable name and the expression's value in the current frame.

Q7: Assignment Diagram

$x = 11 \% 4 = 3$
 $y = x$
 $x **= 2$ $3^2 = 9$

Global frame	
x	3
y	3

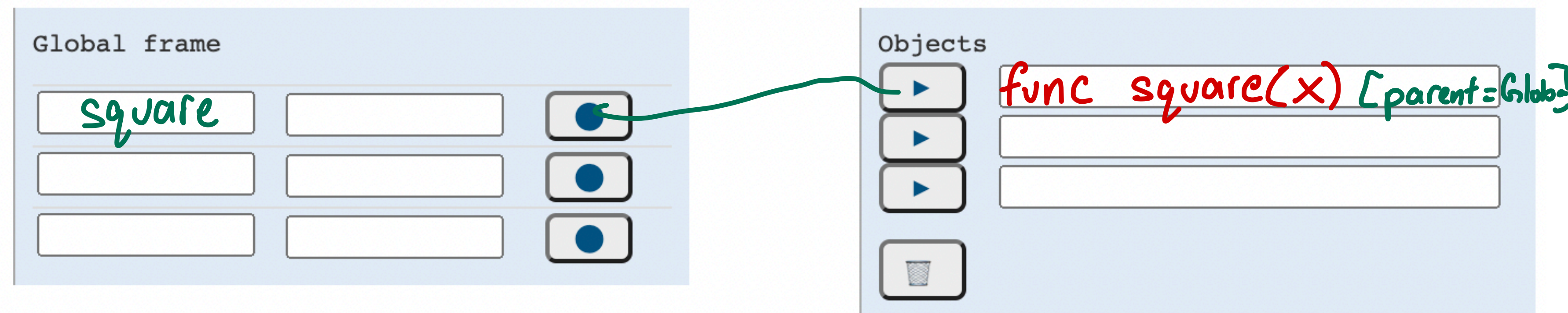
Env Diagram: def Statement

def statement: creates ("defines") a function object and binds it to a name.

```
def square(x)  
    return x * x
```

record the function name (under objects) and bind the function object to the name.

It's also important to write the parent frame of the function, which is where the function is defined

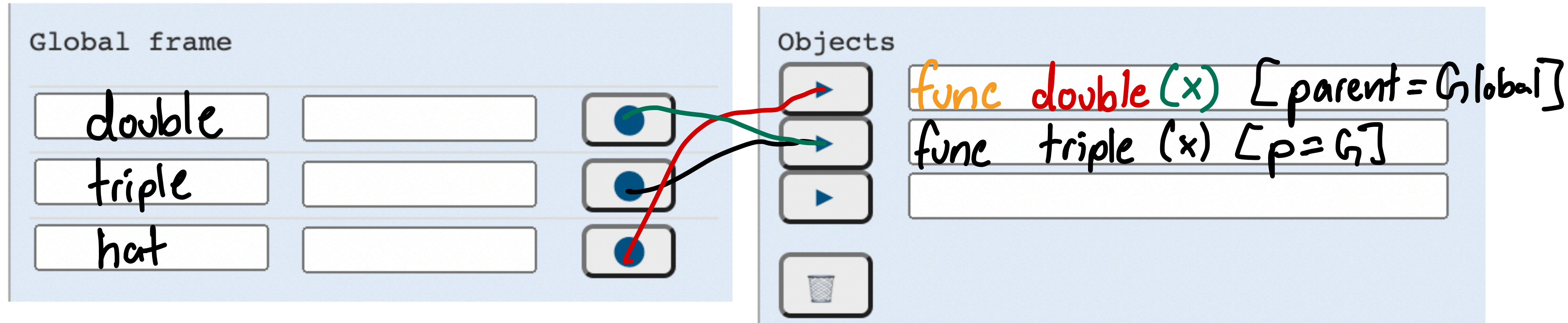


Q8: def Diagram

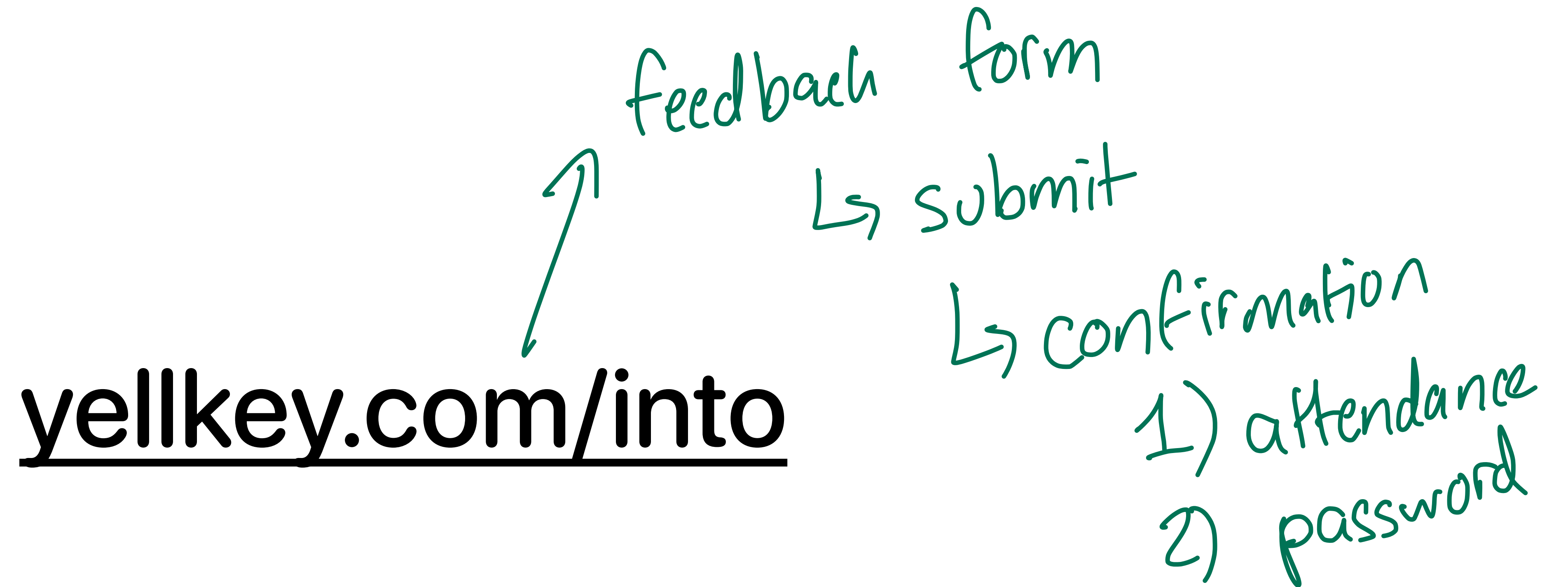
```
def double(x):  
    return x * 2
```

```
def triple(x):  
    return x * 3
```

```
hat = double  
double = triple
```



Feedback + Attendance



Q1 (True/False)

1.1: The goals of floating point are to have a large range of values, a low amount of precision, and real arithmetic results

1.2: The distance between floating point numbers increase as the absolute value of the numbers increase.

Q2

2.1: How many zeroes can be represented using a float?

2.2: What is the largest finite positive value that can be stored using a single precision float?

Q2

2.5: Convert the following single-precision floating point numbers from binary to decimal or from decimal to binary.

You may leave your answer as an expression.

1) `0x00000000`

2) `-infinity` ($-\infty$)

3) `8.25`

4) `0xFF94BEEF`

Q3

3.1: What is the next smallest number larger than 2 that can be represented completely?

3.2: What is the next smallest number larger than 4 that can be represented completely?

Q3

3.3: Define stepsize to be the distance between some value x and the smallest value larger than x that can be completely represented. What is the step size for 2? 4?