6/30 Thur

$\boxed{6.1}$ Evaluation

- Got answers for both questions in the problem.
  - For int to str conversion, I rushed and tried to code first w/o any plan.
  - Once I got the design/plan on paper, I was able to code it up.

→ (i) str to int : $\underset{\text{design}}{20} + \underset{\text{code}}{9}$ min

([illegible])

(ii) int to str : 44 min to design and took upto to 1hr 3min to code and pass all tests.

$\boxed{\text{Total time 1hr 3min}}$

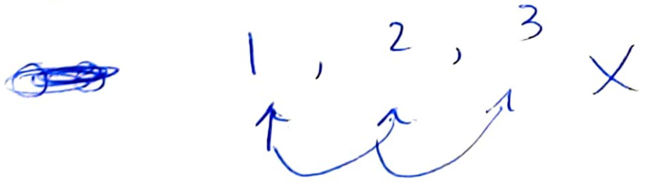
- When solving another sub. problem don't try to use the same ~~logic~~ ideas from previous problem.

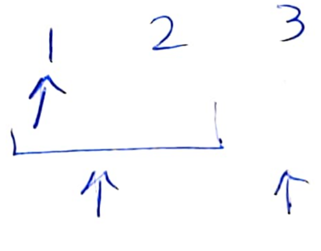
- I had problem in coming up w/ the ~~[illegible]~~ python methods to use.

  ↳ ("").join(s),
  list(s),

- Forgot edge case when x=10 and to state complexit~~y~~ies.

- Could've reduced runtime by inspecting more than one digit.





$\boxed{6.1}$ 6/30/22 Thur

I/O: str / int

The func. takes in string that ~~rep~~ represents a number and returns that number in $\mathbb{Z}$.

- Must handle negative number.

Constaints: $-\infty < \underset{\substack{\uparrow \\ \text{int}(s)}}{} < \infty$

The thing we must worry about is the negative sign $\rightarrow s =$ "-123".

The Naive way is to split the string input, iterate, then use conditionals to check.

Then we can append to an empty list which will result in $O(\underset{\substack{\uparrow \\ \text{cardinality}}}{|s|})$. $\Leftrightarrow O(n)$

Splitting the string ~~can~~ also adds to the time comp. and the whole process can take $\underbrace{O(n^2)}_{\text{time}}$.

We can also use a hashmap to ~~see if the character is in the~~ return the integers.

1/4

I think there is a way to avoid the linear space complexity. and get O(1)

For the edge case ~~car~~ when input represents a neg. ~~[illegible]~~ we can multiply by -1

Maybe not, I can't think of one.

So the sol'n will be
- Set negative equal False
- Split the string check if it's negative
- Create new empty list ~~result~~
- loop through the list of string and check by comparing w/ hashmap.
- ~~Append to ~~an~~ result list~~

☆ Join requires string items — ☹

INSTEAD, we can use the index i in the for loop to check which digit the string is in and multiply by it's place. —

⟶ Example

2/4

$\boxed{\text{Ex}}$ digit_place = 10 May be reversed? ✓

```
for i in range(len(( ).split(s))):

    ~~if  if i == 0~~

        total += d[s[i]] · digit_place
        digit_place = digit_place · 10
```

and check if neg is true

return total.

O(n), O(1)

20min + 9

29min total

3/4

6.1 (cont.) Second problem

int → str.

First check if negative.

- Like the prev. function, we can use digit place to check (Don't need to make a list of int?)
- We can loop over the integer finding the remainder of division then append to list.

$123 \% 10 = 3$

We can we use a while loop and divide the input by 10 until it reaches less than or equal to 1.

This will take $O(n)$ time and space

47 min

4/4

Write a func. throw-dice (N, faces, total)

- This func. returns a value that represents the number of ways to get total summed by throwing a dice w/ ~~#~~ input value of faces, N number of times.

- This looks like a combinatorics problem.

$\boxed{\text{Ex}}$ td(3, 6, 7) $\Longrightarrow$ $\textcircled{15}$

{3, 3, 1}, {2, 4, 1}, ...

The dice face can repeat.

Can I generate a list of numbers from 1 to faces ~~and that~~ N times?

So [1, 2, 3, ..., 6, 1, 2, 3, 4, 5, 6 ... 1, ..., 6]

Or rather, [1, 2, 3, 4, 5, 6] and ~~repeat N times~~?

and choose N items from the list?

$\boxed{\text{Ex}}$ $[1, 2, 3, 4, 5, 6]$

$\uparrow\uparrow\uparrow$

$t_1\, t_2\, t_3$ $\qquad t_i =$ ~~[illegible]~~ ith throw

The we would choose three items that adds to the input total.

Should I move the pointers one at a time?

What are the edge cases?

↳ $N > \text{total}$

We can take a pointer and move others