Refresher Session - 1

- Python + Libraries
- SQL
- Data Analysis & Visualization

} Revision + Practice

= Class starts @ 9:03PM =

Ops:

1. Question - - [follow-up] ✓
2. few mins → code / pseudo code → chat ✓
3. explain : pseudo-code [logic]

Homework : Python-code after the class =

(Q1) Reverse an integer

@ X → string & reverse

✓ [* Ask follow up questions

✓ { -ve numbers ✓

✓ { Zero check ✓

i/p & o/p : integers ✓

100 → 1

100.01 X

123 → 321

-124 → -421

26s

```
N= int(input())
result = 0
while N!=0:
  digit = N % 10
  N= N//10
  result *=10
  result += digit
print(result)
```

Case: -123

```
-123
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-3-c63aa6da822a> in <module>
      4   digit = N % 10
      5   N= N//10
----> 6   result *=10
      7   result += digit
      8 print(result)

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:26

RefresherSession1.ipynb - Cola

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=TPF5pLHyiSVl

+ Code + Text

RAM Disk

[5] 6s

KeyboardInterrupt:

SEARCH STACK OVERFLOW

0s

```
num=-123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
-321
```

5 / 5

Yes NO

Logic → 20:17

Mistake: dont jump into coding

→ handle all boundary case

while doing

Cases → ask follow-up questions

↳ dry run the code

[5]

6s

SEARCH STACK OVERFLOW

0s

```
num=100
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

def rev-int (x:int) →int:

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:36

RefresherSession1.ipynb - Cola

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=TPF5pLHyiSVl

+ Code + Text

RAM Disk

[5]

6s

SEARCH STACK OVERFLOW

ip

0s

```
num=100
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

op

sign +1 tve -1 -ve

100 → 1

100%10

8 / 8

RefresherSession1.ipynb - Colab

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=TPF5pLHyiSVI

+ Code + Text

[5]

SEARCH STACK OVERFLOW

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

1 % 10 → 1

12 % 10 → 2

ip = num = 123

op = reverse_num = 3

123 % 10

3×10 + 2 = 32

320 + 1

123 // 10 = 12

123 → 12

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:40

RefresherSession1.ipynb - Cola

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=TPF5pLHyiSVl

+ Code + Text

RAM Disk

[5]
6s

SEARCH STACK OVERFLOW

123 → 321

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

10 / 10

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:40

RefresherSession1.ipynb - Cola

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=TPF5pLHyiSVl

+ Code + Text

RAM Disk

[5]

6s

SEARCH STACK OVERFLOW

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

11 / 11

Take-aways

1. Follow up Questions
2. Test Cases →
3. Logic building → while; ✓ ✗ !! → Multi operators
   - ↳ hand on paper - logic
4. dry-run

$O(\lceil \log_{10} n \rceil)$

iteration

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:45

RefresherSession1.ipynb - Cola

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=TPF5pLHyiSVI

+ Code + Text

RAM Disk

[5]
6s

SEARCH STACK OVERFLOW

d-digits

n = 1234567893256 ....

n

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

3

```
1
```

O(d)

Time- Complx

n: input

O(n) ; O(n lgn);

X O(len(n)) ; O(lgn)

O(1) ; O(n!)

13 / 13

[5]

6s

SEARCH STACK OVERFLOW

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

d: #digits in n

d { — —

$[O(\lg_{10} n)]$

```
1
```

$3 < \log_{10} 1234 < 4$

[5]

6s

SEARCH STACK OVERFLOW

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

$\log_{10} 100 = 2$

$\log_{10} 1000 = 3$

$2 < \log_{10} 956 < 3$

$2 < \log_{10} 456 < 3$

[5]
6s

SEARCH STACK OVERFLOW

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

num ✓

O(Ceil(log10 |n|))

d

[5]

6s

SEARCH STACK OVERFLOW

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

$O(\lceil \log |n| \rceil)$

Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:54

RefresherSession1.ipynb - Cola

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=TPF5pLHyiSVl

+ Code + Text

RAM Disk

[5]

6s

SEARCH STACK OVERFLOW

ip

op

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

```
1
```

Space Complx.

O(1)

18 / 18

+ Code + Text

RAM
Disk

[5]
6s

SEARCH STACK OVERFLOW

```
num=123
sign = -1 if num < 0 else 1
num = abs(num)
reverse_num = 0
while num > 0:
  reverse_num = reverse_num * 10 + num % 10
  num //= 10

reverse_num *= sign

print(reverse_num)
```

O(1)

```
1
```

(Q2) Find the index of "first" unique character in a string.

(a) no unique char
e.g: abcd dcba → -1 ✓

(b) assume my string has all lowercases

(c) e.g: a → 0

e.g: 1213 → 1
0 1 2 3

e.g: 1213abc32acb
↳ 0

firstUniqueChar(s):

s is in lowercase: ASSUMPTION

1 → counts = {} ✓

(n) building dict O(n)

for l in s:

O(1) dict

if l not in counts: → O(1)

counts[l] = 1 ✓

else:

counts[l] += 1 ✓

s = a b c a c d

abcd

DS: dict

Char	Cnt/Freq
a	1+1=2
b	1 ✓
c	1+1=2
d	1

↳ dictionary (?)

Set ?

firstUniqueChar

```
for i in range(len(s)):
    if counts[i] == 1 → O(1)
        return i
```

O(n)

s → len(s) = n

TIME:

O(n)

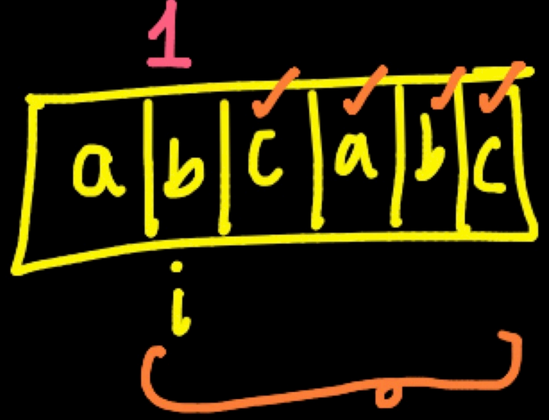
No Unique Char → return -1

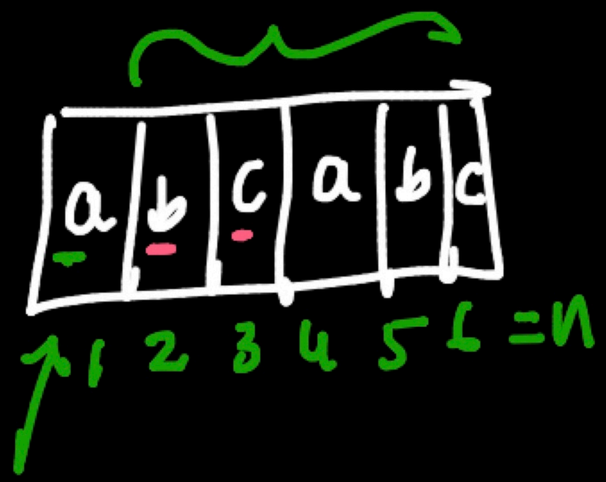
Take-aways:

1. Dict → Search fast O(1) time
   Count frequencies

2. Jerusalem twice is OK

if s[i] not in s[i : len(s)]





(n-1) + (n-2) + (n-3)
- -

↳ $O(n^2)$

Recap:

1. boundary cases -100, 100, 123..
   dry-run
2. use correct DS → dictionary to search

(Q3)

power(x, n)

x^n

x^{*+n}

- ✓ n can be -ve or +ve
- ✓ n = 0 (can be)
- ✓ x can be real ; $x \neq 0$
  n is integer $n \neq \infty$

Ans 1:

iteration

```
power (x, n):
    res = 1
    for i in range(n)
        res *= x

    return res
```

$n < 0$

$n = 0$

Ans 2:-

```
pow(x,n):
    if n==0
        return 1
    else:
        ret = 1
        for i in range(n)
            ret *= x

    return ret
```

2^{-3}

$n<0$

```
1
```

```
def power(x:int,n:int) -> int:
ans =1
sign = -1 if n<0 else 1
n=abs(n)
for i in range(n):
  ans *= x
print(ans)
if sign == -1:
  return (1/ans)
else:
  return (ans)
```

```
[ ]
```

```
1
```

```
def power(x:int,n:int) -> int:
ans =1
if n==0:
  return 1
sign = -1 if n<0 else 1

n=abs(n)
for i in range(n):
  ans *= x
print(ans)
if sign == -1:
  return (1/ans)
else:
  return (ans)
```

```
    reverse_num *= sign
[15]
    print(reverse_num)
```

```
1
```

0s

```
def power(x:int,n:int) -> int:
  ans =1
  sign = -1 if n<0 else 1

  n=abs(n)
  for i in range(n):
    ans *= x
  print(ans)
  if sign == -1:
    return (1/ans)
  else:
    return (ans)
```

0s

```
[18] power(2,0)
```

```
1
1
```

```
    reverse_num *= sign
[15]
    print(reverse_num)
```

```
1
```

```
def power(x:int,n:int) -> int:
  ans =1
  sign = -1 if n<0 else 1
  n=abs(n)
  for i in range(n):
    ans *= x
  print(ans)
  if sign == -1:
    return (1/ans)
  else:
    return (ans)
```

n=0

n<0 ⟶ n= -3

$1/x^3 = x^{-3}$

```
[18] power(2,0)
```

0s

```
1
1
```

1

```
def power(x:float,n:int) -> float:
  # x !=0
  ans =1
  sign = -1 if n<0 else 1
  n=abs(n)
  for i in range(n):
    ans *= x
  print(ans)
  if sign == -1:
    return (1/ans)
  else:
    return (ans)
```

Time Complx:

O(|n|) ; O(log n)

can we do better than this

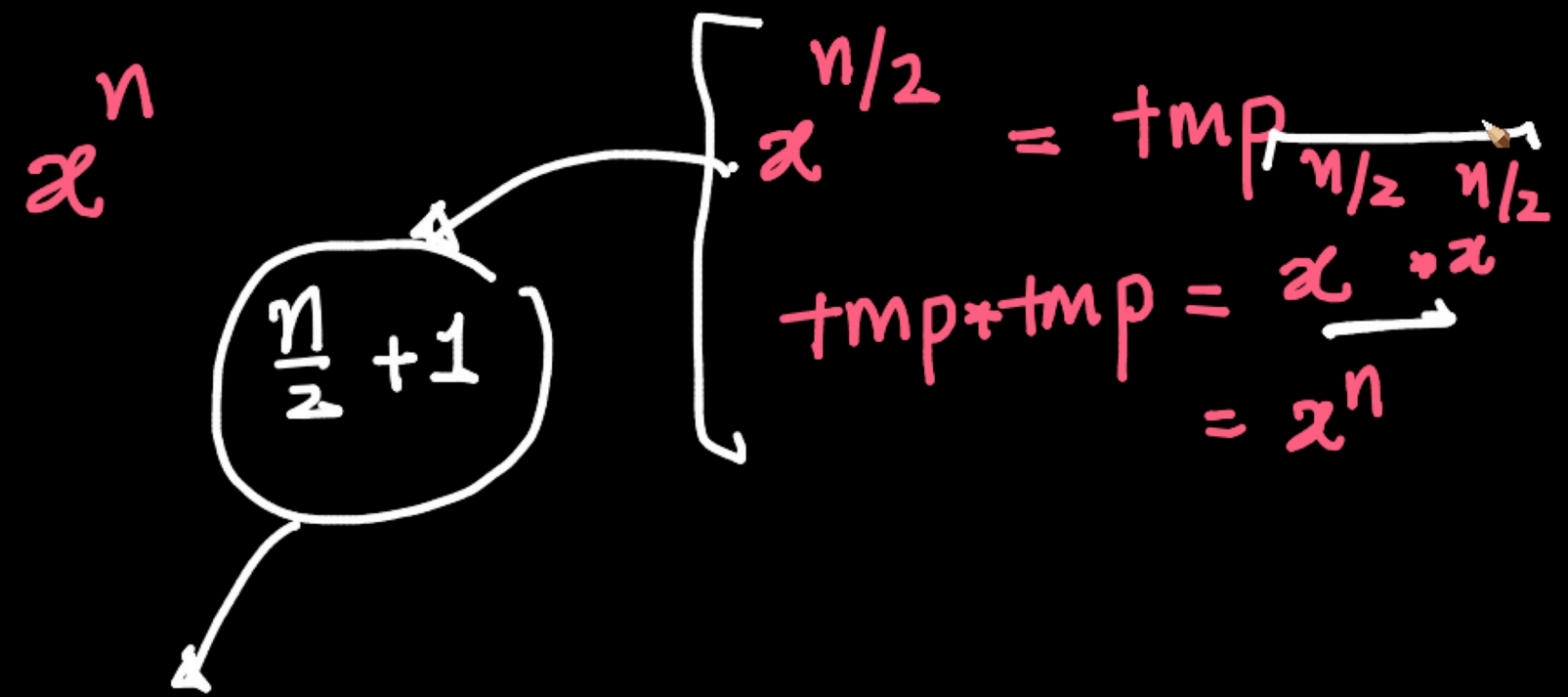
[20] power(2,0)

0s

1
1

Math:

n is even



O(n) time

Pattern: Recursion

$n = 0$ — case

$n < 0$ — case

$n > 0$

1

```
def power(x,n):
  if n<=1:
    return x
  else:
   return(power(x*x,n-1))

  x=2
  n=-2
  sign = 1
  if n<0:
    print((1/power(2,abs(n))))
  elif n==0:
    print(x)
  else:
    print(power(2,abs(n)))
```

Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:56

RefresherSession1.ipynb - Cola | 5^10 - Google Search

colab.research.google.com/drive/106vTGSscGzRIBfO3G_ZVTe1HHagdss0u#scrollTo=Jz98oTo02usw

+ Code + Text RAM Disk

```
1340780792994259709957402499820584612747936582059239337772356144372176403007354697680187429816690342
```

```
[24] def power(x,n):
       if n == 0:
         return 1
       else:
         return x * power(x, n-1)
```

time complx: O(n)

```
power(5,10)
```

```
9765625
```

$x^n \rightarrow x * x^{n-1}$

$x * x^{n-2} \rightarrow x * x^{n-3}$. ---- $x * x^0$

$O(\log_{10} n)$

37 / 37

$$x^n = \begin{cases} 1 & \text{if } n=0 \\ x \times x^{n-1} \times x & -(1) \longrightarrow O(n) \\ (x^2)^{n/2} & \end{cases}$$

termination case

$$x^n = (x^2)^{n/2}$$

n is even

$$x^n = (x^2)^{n/2}$$

n is odd

$$x^n = x^1 * (x^2)^{\frac{n-1}{2}}$$

$$= x^1 * x^{n-1}$$

$$= x^n$$

n is 0 ; $x^n = 1$

power (x, n) -10

if n==0 return 1 ✓

else

if n<0 return 1.0/pow(x, -n) ✓

else:

→ if n%2==0 return pow(x*x, n//2)

→ else: return pow(x*x, n//2)*x

O(log n)

$n = 2^k$

$k = \log_2 n$

$\frac{n}{2^k} = 1$

n → 1 → $x^2 \times x$ (n/2) → 2 → $x^2 \times x^{n/4}$ → 3 → $n/8 = 2^3$ → ... → $\frac{n}{2^k} = 2^1$ → 4

Lessons:

① Follow-up Questions + iteration
Various Cases
(boundary)

② Dict to Seaselo-fast + two times iteration

③ Recursion

SQL

SQL, or *Structured Query Language*, is a language to talk to databases. It allows you to select specific data and to build complex reports. Today, SQL is a universal language of data. It is used in practically all technologies that process data.

SAMPLE DATA

COUNTRY

id	name	population	area
1	France	66600000	640680
2	Germany	80700000	357000
...	...	...	...

CITY

id	name	country_id	population	rating
1	Paris	1	2243000	5
2	Berlin	2	3460000	3
...	...	...	...	...

QUERYING SINGLE TABLE

Fetch all columns from the `country` table:

```
SELECT *
FROM country;
```

Fetch `id` and `name` columns from the `city` table:

```
SELECT id, name
FROM city;
```

Fetch city names sorted by the `rating` column in the default ASCending order:

```
SELECT name
FROM city
ORDER BY rating [ASC];
```

Fetch city names sorted by the `rating` column in the DESCending order:

```
SELECT name
FROM city
ORDER BY rating DESC;
```

ALIASES

COLUMNS

```
SELECT name AS city_name
FROM city;
```

TABLES

```
SELECT co.name, ci.name
FROM city AS ci
JOIN country AS co
  ON ci.country_id = co.id;
```

FILTERING THE OUTPUT

COMPARISON OPERATORS

Fetch names of cities that have a rating above 3:

```
SELECT name
FROM city
WHERE rating > 3;
```

Fetch names of cities that are neither Berlin nor Madrid:

```
SELECT name
FROM city
WHERE name != 'Berlin'
  AND name != 'Madrid';
```

TEXT OPERATORS

Fetch names of cities that start with a 'P' or end with an 's':

```
SELECT name
FROM city
WHERE name LIKE 'P%'
   OR name LIKE '%s';
```

Fetch names of cities that start with any letter followed by 'ublin' (like Dublin in Ireland or Lublin in Poland):

```
SELECT name
FROM city
WHERE name LIKE '_ublin';
```

OTHER OPERATORS

Fetch names of cities that have a population between 500K and 5M:

```
SELECT name
FROM city
WHERE population BETWEEN 500000 AND 5000000;
```

Fetch names of cities that don't miss a rating value:

```
SELECT name
FROM city
WHERE rating IS NOT NULL;
```

Fetch names of cities that are in countries with IDs 1, 4, 7, or 8:

```
SELECT name
FROM city
WHERE country_id IN (1, 4, 7, 8);
```

QUERYING MULTIPLE TABLES

INNER JOIN

`JOIN` (or explicitly `INNER JOIN`) returns rows that have matching values in both tables.

```
SELECT city.name, country.name
FROM city
[INNER] JOIN country
  ON city.country_id = country.id;
```

CITY

id	name	country_id
1	Paris	1
2	Berlin	2
3	Warsaw	4

COUNTRY

id	name
1	France
2	Germany
3	Iceland

FULL JOIN

`FULL JOIN` (or explicitly `FULL OUTER JOIN`) returns all rows from both tables – if there's no matching row in the second table, `NULL`s are returned.

```
SELECT city.name, country.name
FROM city
FULL [OUTER] JOIN country
  ON city.country_id = country.id;
```

CITY

id	name	country_id
1	Paris	1
2	Berlin	2
3	Warsaw	4
NULL	NULL	NULL

COUNTRY

id	name
1	France
2	Germany
NULL	NULL
3	Iceland

LEFT JOIN

`LEFT JOIN` returns all rows from the left table with corresponding rows from the right table. If there's no matching row, `NULL`s are returned as values from the second table.

```
SELECT city.name, country.name
FROM city
LEFT JOIN country
  ON city.country_id = country.id;
```

CITY

id	name	country_id
1	Paris	1
2	Berlin	2
3	Warsaw	4

COUNTRY

id	name
1	France
2	Germany
NULL	NULL

CROSS JOIN

`CROSS JOIN` returns all possible combinations of rows from both tables. There are two syntaxes available.

```
SELECT city.name, country.name
FROM city
CROSS JOIN country;

SELECT city.name, country.name
FROM city, country;
```

CITY

id	name	country_id
1	Paris	1
1	Paris	1
2	Berlin	2
2	Berlin	2

COUNTRY

id	name
1	France
2	Germany
1	France
2	Germany

RIGHT JOIN

`RIGHT JOIN` returns all rows from the right table with corresponding rows from the left table. If there's no matching row, `NULL`s are returned as values from the left table.

```
SELECT city.name, country.name
FROM city
RIGHT JOIN country
  ON city.country_id = country.id;
```

CITY

id	name	country_id
1	Paris	1
2	Berlin	2

COUNTRY

id	name
1	France
2	Germany

NATURAL JOIN

`NATURAL JOIN` will join tables by all columns with the same name.

```
SELECT city.name, country.name
FROM city
NATURAL JOIN country;
```

CITY

country_id	id	name
6	6	San Marino
7	7	Vatican City
5	9	Greece
10	11	Monaco

COUNTRY

name	id
San Marino	6
Vatican City	7
Greece	9
Monaco	10

`NATURAL JOIN` used these columns to match rows:
city.id, city.name, country.id, country.name