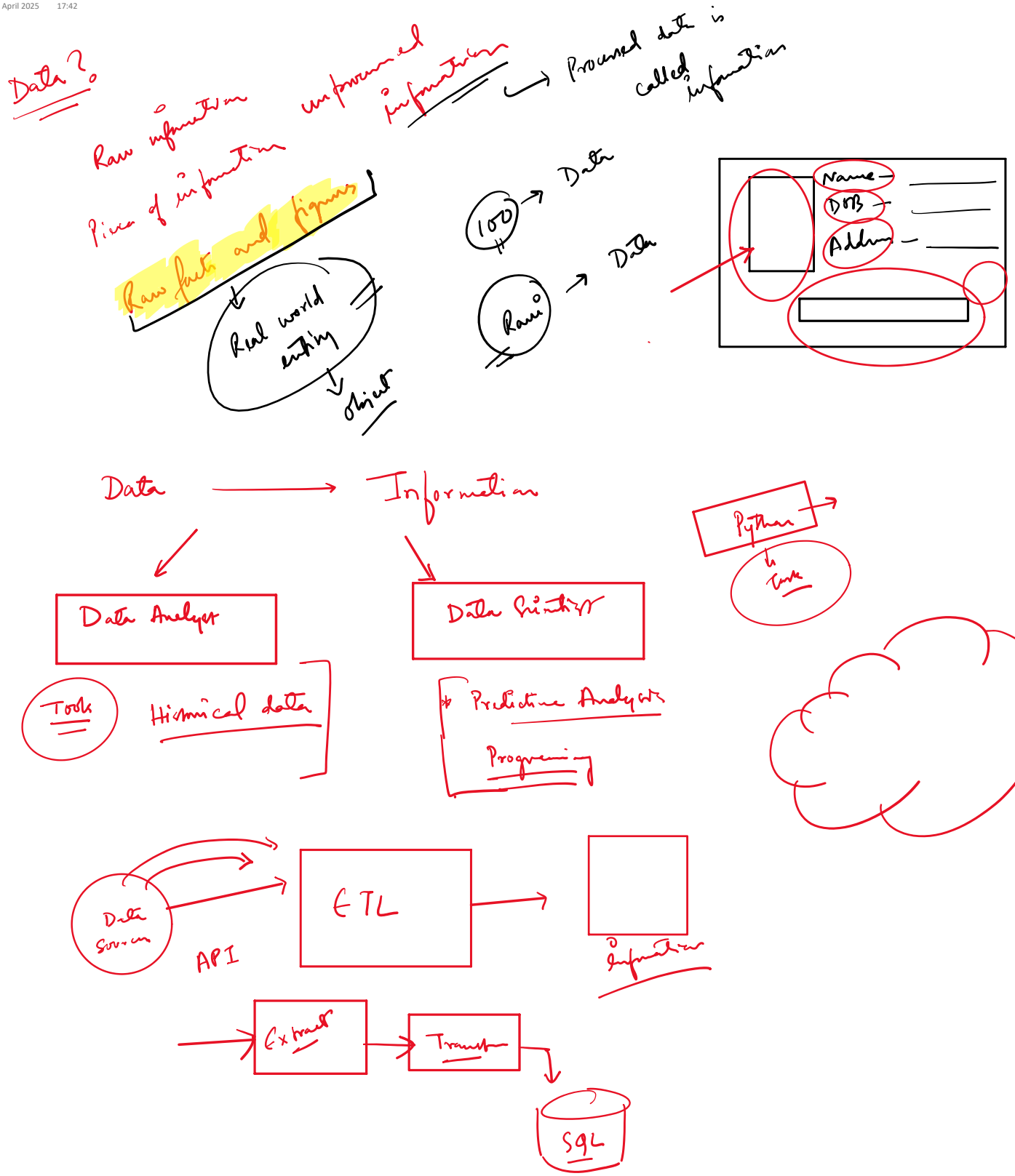
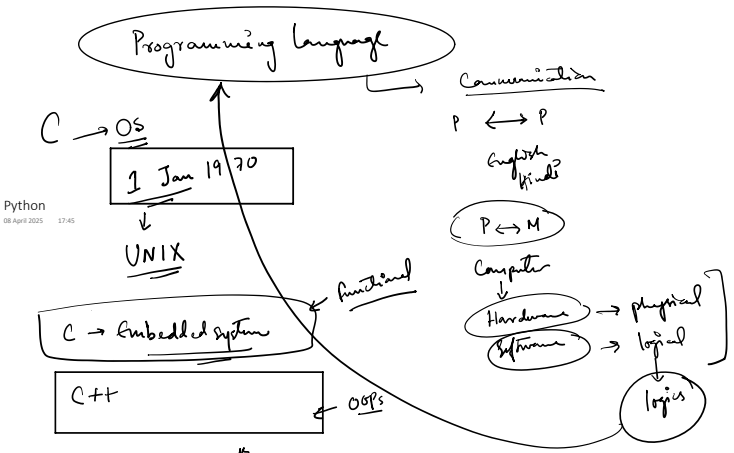
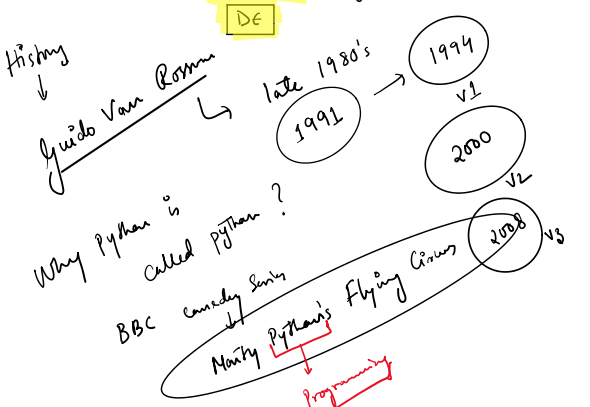
Data Engineering Introduction

07 April 2025 17:42



Introduction to Python :-





Features →

- Easy to implement
- Dynamic
  - ↳ Typed
  - ↳ Memory allocation

C language

```
#include<stdio.h>
#include<conio.h>
void main()
{
    clrscr();
    printf("Hello World!");
    getch();
}
```

sys, stdout

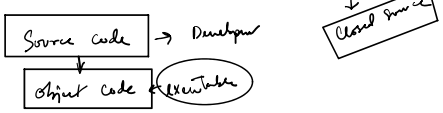
print("Hello World!")

main

int a=45;

a=45

Open Source :-

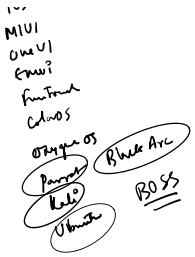


Windows
Closed Source

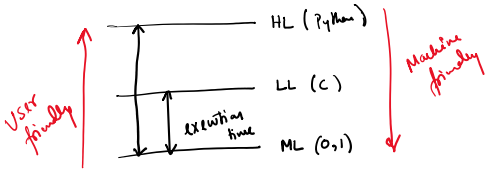
Linux
Android

Windows
Macos
Unix
Ios
MIUI
OneUI
Emui
Funtouch

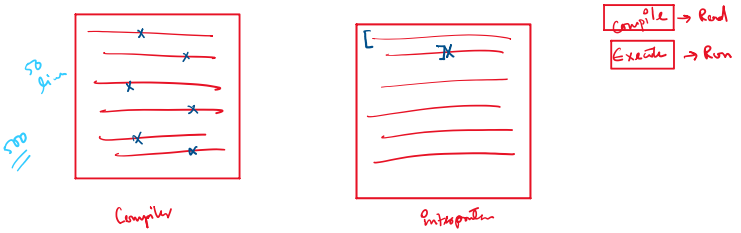
Kernel
Shell

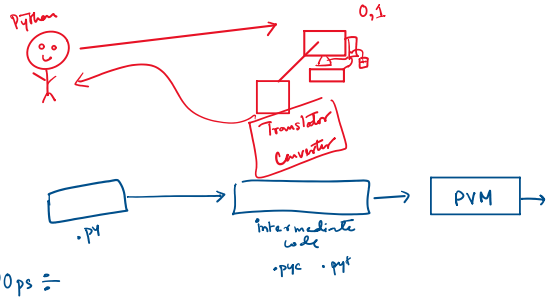


High level

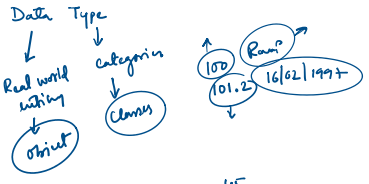


Interpreted →





OOps :-



a = 45

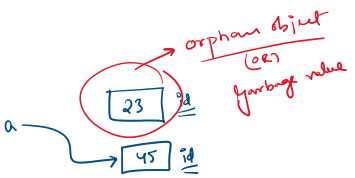
Human

Functions :- read(), write(), dance()
eat(), think(), code()
Sleep() - — -

Attributes :- name, age, designation

Data Types

a = 23
a = 45



a = 1
b = 1

Strings :-

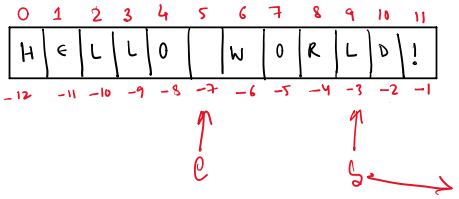
S = "Hello World!"

— Data Structure (Linear)
— Similar data types

S = "Hello World!"

S = Hello World;

— Similar data types

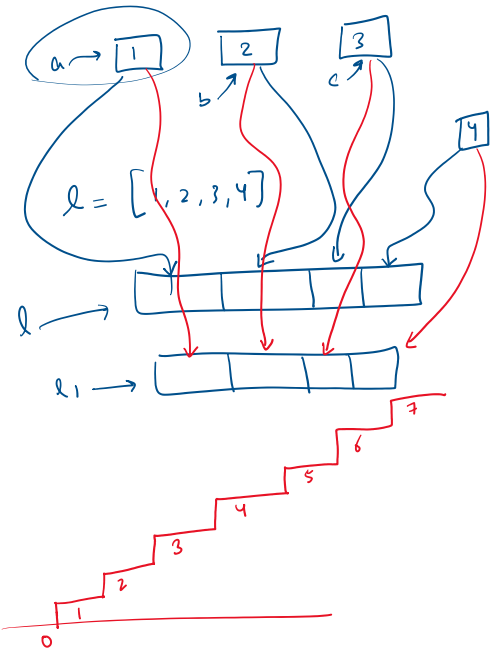


S = "Hello World!"

s[start : stop : step]
direction

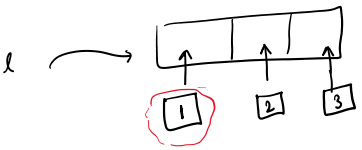
```
print(s[2::5])#Lo, ew!
print(s[:-5:2])#ll, !l
print(repr(s[:4:-3]))#!r_, !r l
print(repr(s[-3:4:2])) #lo l,
```

a = 1
b = 2
c = 3

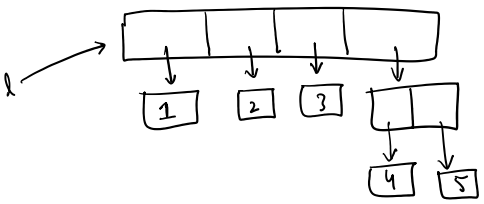


List

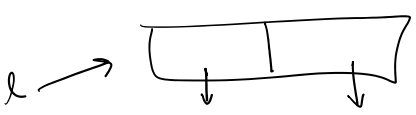
l = [1, 2, 3]

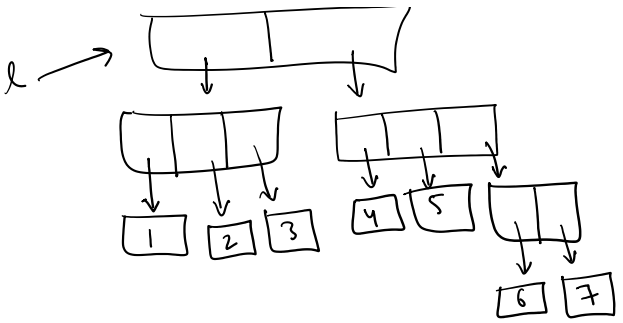


l = [1, 2, 3, [4, 5]]



l = [[1, 2, 3], [4, 5, [6, 7]]]

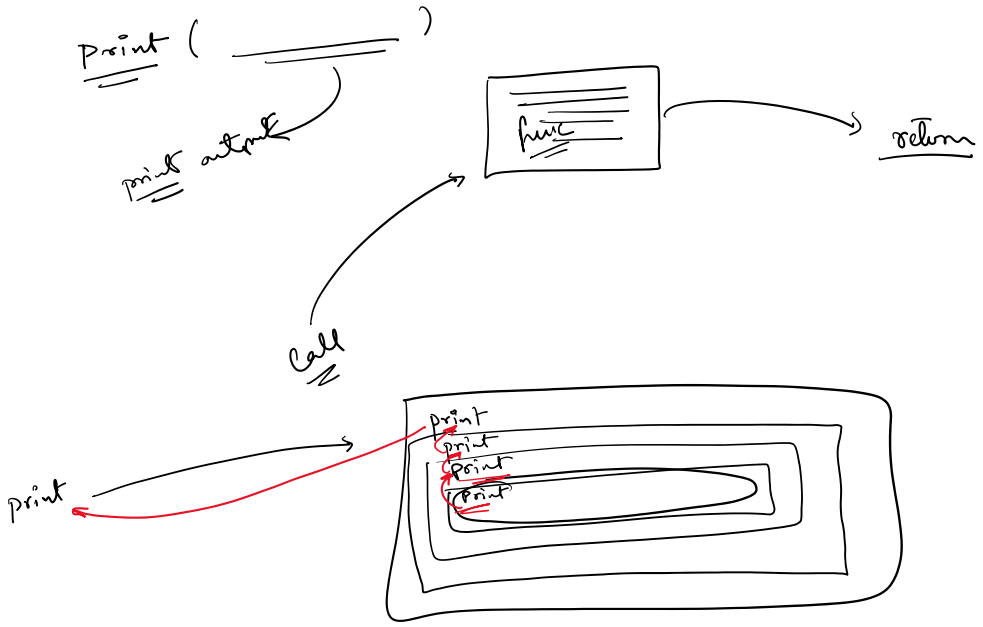




Properties of list :-

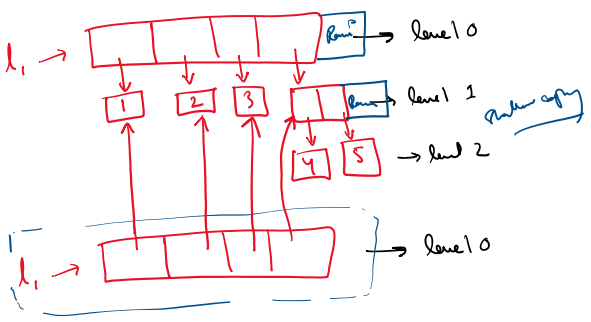
1. Sequential → indexing (slicing)
2. Ordered
3. Mutable (can be changed (inplace))

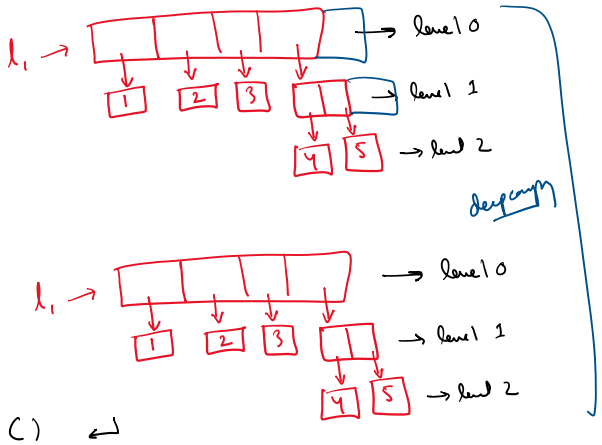
Methods of list :-



Ravi Ramnot

$l_1 = [1, 2, 3, [4, 5]]$





append ( ) ↵

extend ( ) ↵

Copy → shallow and deep copy

clear ( ) →

1100 → 12

& 0111

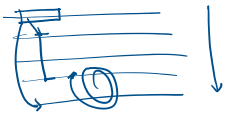
0100 → 4

1100

(OR) 1001

1101 → 13

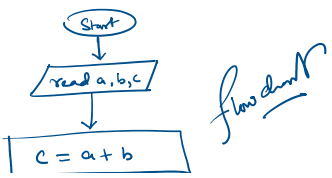
Control flow

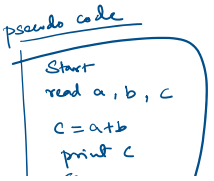


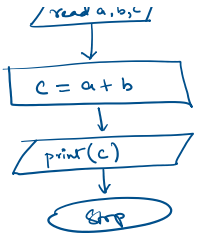
① Conditional Statements

② Iteration Statements

③ Break, continue

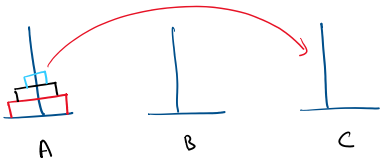


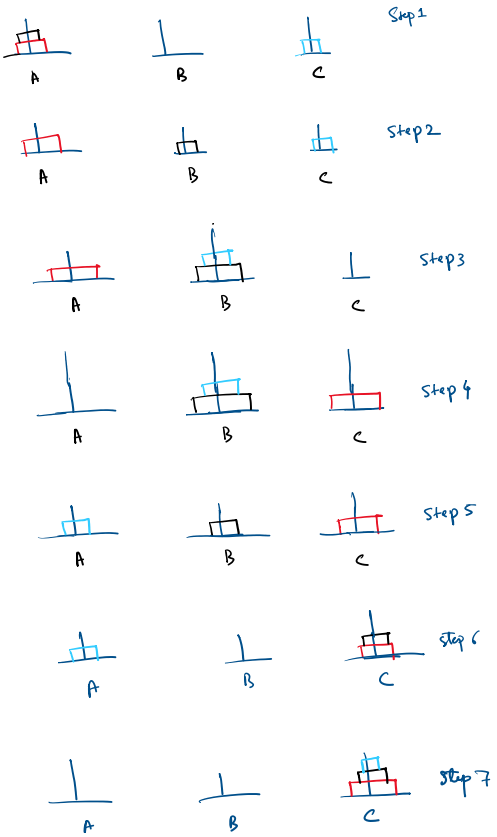




flow chart

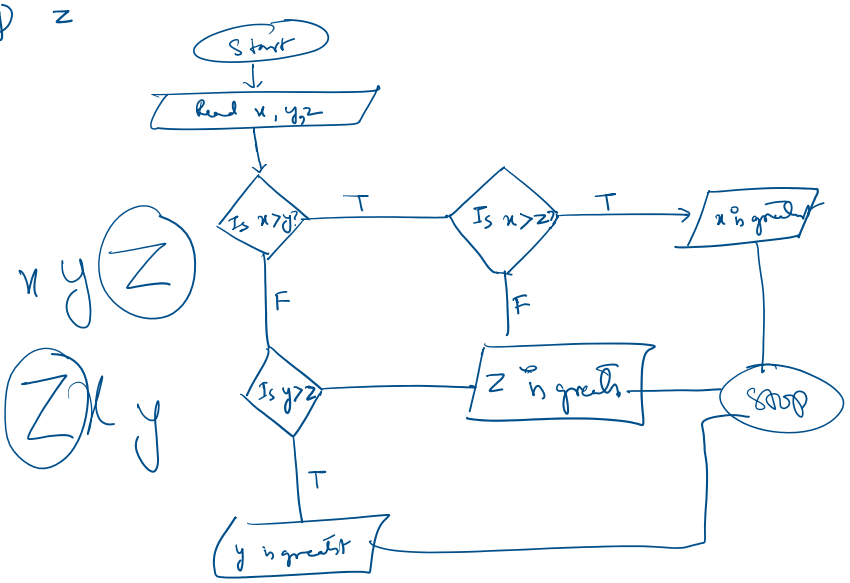
read a , b , c
c = a+b
print c
Stop



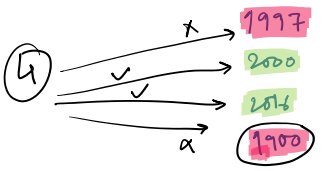


N_1 N_2 N_3

(x) (y) z



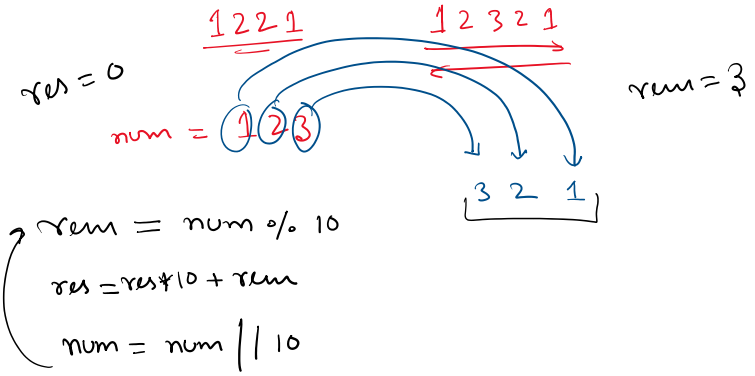
Leap



✓ 2000
✗ 1900
→ year % 400 == 0

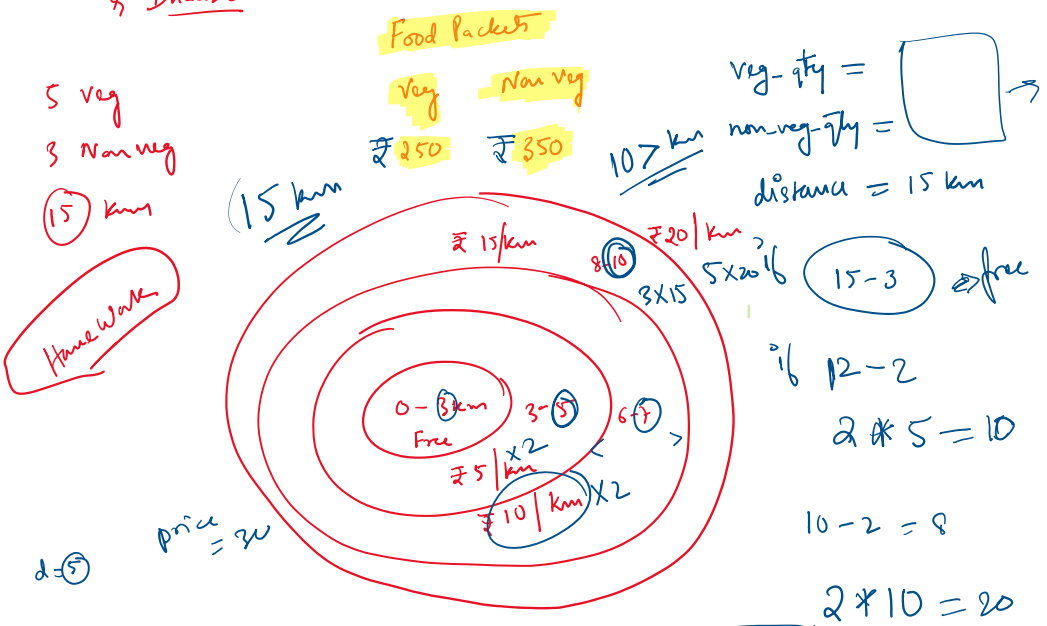
2016
1997
→ Non century

year % 100 != 0 and year % 4 == 0



rem = num % 10
res = res * 10 + rem
num = num // 10

Dhaaba



veg-qty = ☐
non-veg-qty = ☐
distance = 15 km

5 x 20 % (15-3) ⇒ free

% 12-2
2 * 5 = 10

10 - 2 = 8
2 * 10 = 20

8 - 2 = 6
2 * 15 = 30

6 * 20 = 120 + 30 + 20 + 10
= 180

If d>8 and d<=10:

d_charge = (d-3) * 5 + (d-5) * 10 + (d-8) * 15

price = (d-8) * 15
d = 8

price = price + (d-5) * 10
= 20 + 30

price = price(d-5) * 10]

30 + 30

d = 5

= 180

₹ 500
₹ 200
₹ 100
₹ 50
₹ 20
₹ 10
₹ 5
₹ 2
₹ 1

₹ 25677 /- → 500
177 /- → 100
77 /- → 50
27 /- → 20
7 /- → 5
2 /- → 2

8 → 10

(3 * 0) + (2 * 5) + (2 * 10) + ((d-7) * 15)

5 → 7

(3*0) + (2*5) + ((d-5) * 10)

Iteration Statements

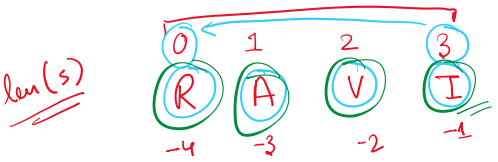
(1) While

Syntax :-

```
while condition :
    Statement - 1
    - - - 2
    - - - 3
    - - - - - -
    - - - - - n
```

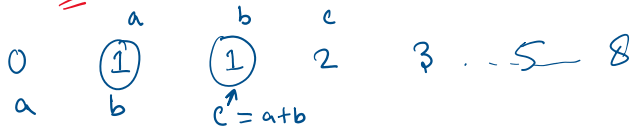
5 x 1 = 5
5 x 2 = 10
| | | | |
5 x 10 = 50

0 1 2 3




```
print(s[3])
print(s[2])
print(s[1])
print(s[0])
```

10 n = ?

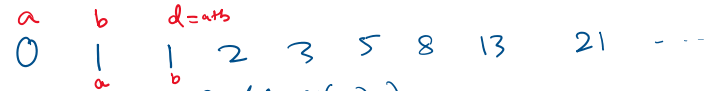


a + b

a = a + b

0 + 1

a = 1




```
n = int(input())
c = 1
a,b = 0,1
while c <= n :
    d = a+b
    print(d)
    a = b
    b = d
```