Python

CLASSTIME
Date /

Week-1

L1.1 Introduction:
- talk to computer
- Python — widely spoken, easy to learn, powerful
- do complex things very fast
- open a "terminal" and "code"

L1.2 Introduction to Replit
- replit.com → website → create repl
- print ('Hello')
- print 7 lines with stars
- comp. does not know what to do

L1.3 More on Replit, Print and Common Mistakes
- add file → add folder
- settings → layout → code intelligence
- print ('a', 'b', 'c'), print (10)
- print commands many print statements together
- Mistakes —
  ① spelling could be wrong
  ② only () brackets allowed [], <>, {} [X]
  ③ for string things we need to have '' or " "
- this is known syntax.

L1.4 Introduc^n to Variables
- a = 10, print (a) b = 20; print (b)
- print (a+b) = 30
- → a = a+1 ⇒ 11 ⇐ print (a)
- Incrementa^n = (a = a + 1)
- so these are variables ....
- n = int (input (1)) ⇒ print (n)

→ please ask the person to write sth. and convert it to integer

CLASSTIME
Date / /

L1.5 Variables and Input Statement

- print ("Type in your name: ")
- n = str (input ()) → print (n)

print ("which place u r in ? ")
p = str (input ())
↳ sth different
print ('Hello', n, p)

- ',' gives the space

L1.6 Variables and Literals

- print → prints the msg
- input → it takes the input and stores it inside var.
- n = str (input ("what is your name? "))
  ↳ merging input & print statement
- n is the variable → it is a container to store values.
- literals are actual value in the variable at starting, it can be changed later.
- literals are used RH side of eqn, variables can be used on both sides

```
r = int (input ("Enter r : "))
area = 3.14 * r * r
print (area)
```

r, area → var
3.14 → literal

L1.7 Data Types 1

- 10 is a integer, Suddarshan is a string
  6.3 is a float
- print (type (n)) → this func tells the class of that var.

- floating → str. other than int., beyond an int.
- internally, comp. gives some particular memory
- there are particular 'data types'
- l = [10, 20, 30] ⇒ print (l[0]) ⇒ 10
  ↳ this is a list print (type (l[2])) ⇒ int

L1.8 Data Types -2

- b1 = True ; print (type (b1)) ⇒ bool
- a = int (5.7) = 5 ; b = int ('10') = 10
- a = float (a) = 9.0 ; b = float ('5.3') = 5.3
- a = str (a) = 9 ; b = str (5.3) = 5.3
- Type conversion / type casting
- a = bool (10) ⇒ True
  b = bool (0) ⇒ False
  c = bool (-10) ⇒ True
  } all int. value gives True bool value, but only 0 gives the false value.
- but if, a = bool ('0') is also true but a = bool ('') or empty string is false.

L1.9 Operators & Expressions - 1

- n = 3 + 2 ⇒ print (n) ⇒ 5
- a = 1, b = 2 ⇒ n = a + b ⇒ print (n) ⇒ 3
- a = "a", b = "b" ⇒ a + b ⇒ ab concatanate
- a = [1, 2], b = [3, 4] ⇒ a + b ⇒ [1, 2, 3, 4] ⇒ Union
- + operator ⇒ data type matters
- n = 10 + (13 * 2) ⇒ print (n) ⇒ 36
- Operator precedance ( ) → / → * → + → -

L1.10 Operators & Expressions - 2

- Arithmatic, Relational & Logical Operators
  ↳ +, -, *, /, //, %, **

CLASSTIME Pg. No.
Date / /

- + ⇒ addition
- ⇒ subtraction
* ⇒ multiplication
/ ⇒ division
// ⇒ floor division. Eg 7//3 ⇒ 2.33
7//3 ⇒ 2
%. ⇒ modulus Eg. 7%3 ⇒ $\frac{7}{3} = 2\frac{(1)}{3}$ ⇒ Remainder
= 1
** ⇒ exponential Eg. 6**2 ⇒ $6^2 = 36$

- >, <, >=, <=, ==, != ⇒ gives boolean values
print (5>10) ⇒ False
(10>5) ⇒ True
(10>=5) ⇒ True
(5>=5) ⇒ True
(5==5) ⇒ True
(10!=5) ⇒ True

- and, or, not ⇒ logical operators
print (T & T) ⇒ T, (F & T) = (T & F) = (F & F) = F
print (T or T) = (F or T) = (T or F) = T (F & F) = F
not (True) ⇒ False
not {False} ⇒ True
} can be used with or w/o brackets.

1.11 Intro to Strings

- s = 'coffee' t = 'bread'
- print (s[0]) ⇒ c
- print (s[1:3]) ⇒ 'of' (s[1:5]) ⇒ 'offe'
(s[3:5]) ⇒ 'fe' ⇒ string slicing
- s = '01234' a = int(s[4]) b = int(s[2])
a + b ⇒ 4 + 2 = 6

CLASSTIME Pg. No.
Date / /

L1.12 More on strings

- Replication: s = 'good' ⇒ print(s * 5) ⇒ 5 times good
  print(s[0] * 5) ⇒ ggggg
- String comparison: x = 'India'
  print(x == 'India') ⇒ True
  print(x == 'india') ⇒ False
- print('apple' > 'one') ⇒ F
  print('four' < 'ten') ⇒ T
  0th letter of 1st string compares with 0th letter of 2nd string. Hence, a > a and t > f
- s = 'python' ⇒ print(s[2]) ⇒ t
  ..... [-2 -1] ⇒ negative string indexing
- s = 'python' ⇒ print(len(s)) ⇒ 36
  0 ⟶ (len - 1) ↳ gives the length of string

L1.13 Conclusion: FAQs

- learn the syntax properly → with them
- syntax errors and others → bit irritating
- easy to learn, powerful, most sought after, open-source softwares (its in Python) ⇒ Python
- practice a lot. 1. repeatedly fresh code)
  2. syntax & logic ↳ do it multiple times

CLASSTIME | Pg. No.
Date / /

Week-2

L2.1 Introduction

- practice → warm up
- doing complex things easily by comp.
- variables → bucket / store entities
  ↳ certain things at certain places only
- advanced print statements
- if loop conditioning for while

L2.2 Variable – Programmer's Perspective

- a = 100000 (ram_bank)
  b = 500000 (ram_loan)
  c = 2000000 (lak_bank)
  d = 1000000 (lak_loan)

} Var. should be self-explanatory

```
net_income = ram_bank + lak_bank
net_liability = ram_loan + lak_loan
final_value = net_income - net_liability
```

} add comments ## type

- generally, comments are below the code line.
- commenting is a good programming prac.

L2.3 Variables Revisited : Dynamic Typing

- a = 10 ⇒ print(type(a)) ⇒ int
  a = 'India' ⇒ " ⇒ str
- first line, ## dynamic typing ⇒ things Δ from 1 class to another
- n = 10 ⇒ print(type(n)) ⇒ int
  n = n/2 ⇒ " ⇒ float (due to division)

L2.4 More on Var; Operators and Expressions

- do not use official keywords (if, and) as your variable name.
- Only alpha-numeric char. & underscore can

CLASSTIME Pg. No.
Date / /

be used for variable.
- we must start the variable with alphabet or underscore but not numerical. 1a ✗
- Python is ~~not~~ case-sensitive.
- multiple assignment x, y = 1, 2
  print (x, y) ⇒ 1, 2
  or x = y = z = 10 ; print (x, y, z) ⇒ 10, 10, 10
- x, y = y, x ⇒ swaps the values and stores
- x = 10, print (x), del (x) ⇒ this deletes the variable
- shorthand operator
  c = 10
  c -= 1
  c -= 1
  print (c) ⇒ 8

  c = 0 → (c = c + 1)
  (c += 1) ⇒ c = 1
  c += 1 ⇒ c = 2
  print (c) ⇒ 2

  c = 20
  c *= 1
  print (c) ⇒ 20
- in ⇒ checks particular value or not
  print ('a' in 'abcdef') ⇒ True
- Chaining Operator ⇒ multiple relational operators

  x = 5
  print (1 < x < 10) ⇒ T (10 < x < 20) ⇒ F
  (5 == x > 4) ⇒ T (10 > x <= 9) ⇒ T

L2.5 Escape charac. & types of quotes
- print ('it\'s nice') or "it's"
  ↳ escape charac. to consider this as a apostrophe.
- print ('We are from \"IIT\" Madras.')
- print ('a. b') ⇒ to increase space
  print ('a.\t\tb')
  print ('a.\nb')
  ↳ print a & b in the separate lines

print ("a'b'c") } what happens?
print ('a"b"c') }

- ' ' " " ⇒ strings
  ↳ only for single statements
- z = """ a
  b
  c """ ⇒ 3 quotes ⇒ prints as its multiple statements
- """ comment 1
  comment 2
  comment 3 """ } gives multi-line comments

L 2.6 String Methods

- x = "pyTHOn" → print(x.lower())
  lower() → all into lower
  upper() → all into upper
  capitalize() → 1st letter into upper
  title() → every word 1st letter into upper
  swapcase() → swaps lower to upper & vice versa
- is lower() ⇒ check all are lower or not
  is upper() ⇒ " all are upper " " } T/F result
  is title() ⇒ checks title rules
  is digit() ⇒ checks for num
  is alpha() ⇒ " for alphabets
  is alnum() ⇒ " for both num & alphabet
- x = '----P----' → print(x.strip('-'))
  strip() ⇒ trims all extra things
  lstrip() ⇒ trims on left things
  rstrip() ⇒ " " right thing

case sensitive
- startswith() ⇒ if str starts with particular value
- endswith() ⇒ " " ends " " "
- count() ⇒ how many times that letter appeared in statement
- index() ⇒ returns the pos" of that specific thing of 1st occurence
- replace() ⇒ value replaces other value.
  x = x.replace('s', 's') } find and replace feature
  print(x)

CLASSTIME Pg. No.
Date / /

L2.7 An interesting cipher → more on strings

- alpha = 'abcdefjk' i = 0
  print (alpha[i]) a
  print (alpha[i+1]) b
  i = 26 print (alpha[i%26]) ⇒ a
  ⋮
  [(i+1)%26]

- s = 'sudarshan' # treebstabo (one letter shift)
  t = '', i = 0, k = 1 ⟺ use of var. k = 2 then shift is by 2 letter
  print (alpha.index(s[0])) ⇒ 18
  t = t + (alpha[(alpha.index(s[i]) + k) % 26])
  i+1
  ⋮
  i+2

⟹ t = t + (alpha[((alpha.index(s[i])) + k) % 26)])

- Caesar Cipher in Cryptography

L2.8 Intro. to if statement

PG 13+ (Parental guidance 13+ yrs)

- b-y = int(input()) c-y = 2021
  age = c-y - b-y
  if (age < 13):
  tab ---- print("No")
  else:
  ------ print("Yes")

L2.9 Tutorial on if, else, & elif statements

Eg. Find whether the given no. is even or odd

```
i = int(input())
if (i%2 == 0):
    print('even')
else:
    print('odd')
```

CLASSTIME | Pg. No.
Date / /

Eg. Find whether the given no. with 0 or 5 or any other no.

```
i = int(input())
if (i%5 == 0):
    if (i%10 == 0):
        print(0)
    else:
        print(5)
else:
    print('Number is not')
```

Eg. Find the grade of student based on "marks"

A	≥ 90
B	≥ 80 & < 90
C	≥ 70 & < 80
D	≥ 60 & < 70
E	< 60

```
m = int(input())
if (m >= 90):
    print('A')
if (m >= 80 and m < 90):
    print('B')
else:
    print('E')
```

```
if (m >= 90)
    print('A')
elif (m >= 80)
    print('B')
```

else: invalid input

elif statement happens only if the initial code was executed

Eg. convert the flowchart into a python code

```
print('Travel A to B')
t = int(input())
l = int(input())
if (t >= l):
    h = int(input())
    if (h >= h):
        print('Train')
    else:
        print('coach')
else:
    p = int(input())
    h = int(input())
    if (p >= h):
        print('business')
    else:
        print('executive')
print('Arrive at city B')
```

CLASSTIME Pg. No.
Date / /

L2.10 Intro. to import library

- import math ⟶ book, bring it to the home from lib.
  print (math.log (10))
  print (math.sqrt (10))
  16 → 4
  print (math.factorial (5))
  print (math.pow (10, 3))
- import random
  print (random.random()) → get 0 → 1
- # let's simulate a coin toss
  import random
  a = random.random()
  if (a < 0.5):
  print ("Heads")
  else:
  print ("Tail")
- # simulate a six-faced die.
  import (random)
  print (random.randrange (1, 7)
  dice 1 = "
  dice 2 = " } total
  print (Total)

L2.11 Diff. ways to import lib.

- import calendar
  print (calendar.month (year, month())
  " (" .calendar())
- from calendar import *
  print (calendar (2021))
  " (month (2021, 10))

- from calendar import month; calendar.
  print(month(2021, 10))
  " (calendar(2021))
- import calendar as c
  print(c.month(2021, 00))

} most used way

} reduces waiting time

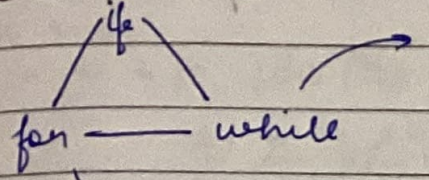
L2.12 Conclusion

- declare variables
- the if loop ⟹ ciphers ⇒ shifts by some unit
- storing related entities
- import possibility ⇒ library functions

Week - 3

CLASSTIME | Pg. No.
Date / /

L3.1 Introduction



→ u keep running, until i tell u too stop

→ keep jogging, keep counting upto 4000, then stop

L3.2 Intro. to while loop

```
print ("India got indep ? ")
year = int(input())
if (year == 1947):
    print ('Right')
else:
    print ('wrong')
```

} how to get multiple chances, until you've entered right

```
while (year != 1947)
    print ("Wrong")
    year = int (input())
print ("Nice")
```

L3.3 While to compute fac.

```
print ('Enter : ")
n = int (input())
```

a = 1 , i = 1

```
while (i <= n)
    a = a * i
    i = i + 1
print (a)
```

L3.4 Tutorial on while loop

Eg. Find the fac. of a given no.

```
num = int (input())
fac = 1
while (num > 0):
    fac = fac * num
    num = num - 1
print (fact)
```

```
if (num < 0):
    print ("NO")
else:
```

CLASSTIME Pg. No.
Date / /

Q. Find the no. of digits in the given no.

```
n = int(input())
n = abs(int(input()))   # abs. → +ve → -ve
d = 1
while (n > 9):
    n = n // 10
    d = d + 1
print(d)
```

1234 → 4
12345789 → 9
8 → 1
0 → 1
-4 → 1

Q. Reverse the digits in the given no.

```
n = int(input())
abs n = abs(n)
if (n >= 0)
    n = 0 * (num % 10)
    n = n // 10
    while (n > 0):
        n1 = n % 10
        n = n // 10
        n = (n * 10) + n1
    print(n)
else:
    n = abs n % 10
    abs = abs n // 10
    while (abs n > 0):
        n1 = abs % 10
        n = n // 10
        n = (n * 10) + n1
    print(n - 2 * n)
```

1234 → 4321
8 → 8
-1234 → -4321

```
n = int(input())
while (abs n > 0):
if (n >= 0)
    print(n)
else:
    print(n - 2 * n)
```

Optimization

Date / /

Q. Find the whether the entered no. is palindrome or not

```
n = int(input)
abs n = abs(n)
ni = abs n %. 10
abs n = abs n //10
while(abs n > 0)
    ni = abs n %. 10)
    abs n = abs n //10
    n = n * 10 + ni
if (n < 90)
    n = n - 2 * n
if (n == n)
    print('Palindrome')
else:
    print('No')
```

L3.5 Intro. to for loop

```
for i in range((10)):
    print(i, "Hello world")
if (i %. 2 == 0)
else:
    print(i, "Hi World")
```

n = int(input())
- the var. i will take values 10 (0 → 9)

L3.6 for loop to add the first n nos.

```
n = int(input())
ans = 0
for i in range(n+1):
    ans = ans + i
# i = i + 1 (not needed here) → think why?
```

Gauss, a mathematician to calculate the sum of first n numbers.

CLASSTIME | Pg. No.
Date / /

L3.7 for loop for multiplication tables

```
n = int(input())
till = int(input())
for i in range(1, till):
    print(n, 'x', i, '=', n*i)
```

⟹ it says range from 1 → (11 − 1)

L3.8 More on range & for loop w/o range

- ```
  for x in range(10):
      print(x)
  ```
- ```
  for x in range(1, 11):
      print(x)
      if (x%2 != 0)
  ```
  ⟹ gives odd nos.
- ```
  for x in range(1, 11, 2):
      print(x)
  ```
  ⟹ step of 2 (optional)
  start (1) (optional); end (11 − 1) (mandatory)
- ```
  for x in range(9, -1, -1):
      print(x)
  ```
  decrement → (0 ⇝ −1)
- ```
  country = 'India'
  for letter in country:
      print(letter)
  ```
  ⟹ print(country[0]), print(country[1]), ..., print(country[5])

foreach kind of for loop ⇒ many advt.

L3.9 Formatted Printing

- ```
  for x in range(10):
      print(x, end=' ')
  ```
  ↳ default is enter to go into a new line
- ```
  d=10, m=5, yy=2021
  print("Date is", d, m, yy, sep='/')
  ```
  ↳ default is space
  ```

d ⇒ int

f ⇒ float

CLASSTIME Pg. No.
Date / /

print("Today is", end = ' ')
print(d, m, y, sep = '-')

- n = int(input())
for i in range(1, 11)
print(f'{num} x {i} = {num*i}')

↳ formatted printing

format func ⇐ print('{0} x {1} = {2}'.format(n, i, n*i))
⇐ print('%d x %d = %d' % (n, i, n*i))
print using string modulo operator ⇒ C lang.

- pi = 22/7
print(f'Value of pi = {pi}') → :.2f → 0:.2f
" ('Value of pi = {0}'.format(pi))
" ('Value of pi = %f' % (pi)) → %.2f

⇓ Format specifiers (3.14 only)

(pi.:.2f)

- print('{0:5d}'.format(1))
" " (11)
(111)
(1111)
(11111)

L3.10 Tutorial on for loop & diff. btw. while & for

Eg. Factorial of a given no.

n = int(input()) f = 1
if n < 0
ND
else:
for i in range(n, 1, -1):
f = f * i
print(f)

but better is while loop for infinite things

CLASSTIME Pg No.
Date / /

Eg Find the no. of digits in a given no.
Q. write foreach way of for loop

```
n = abc(float(input()))
s = str(n)
digits = 0
for c in s:
    digits = digits + 1
print(digits)
```

Eg Reverse the digits in the given no.

```
n = str(n)
for c in str(n):
    rev = c + rev
if (num >= 0):
    print(rev)
else:
    print('-' + rev)
```

Eg find whether the no. is palindrome or not

```
if (num < 0):
    rev = '-' + rev
if (num == int(rev)):
    print('Palindrome')
else:
    print('Not Palindrome')
```

- factorial of a given no. —— for
- find the no. of digits —— while
- reverse the digits —— while
- palindrome check —— while
- accept integers using input to find max until -1] — while
- print the multiplication table —— for
- for whether the no. is prime/not — for
- find the sum of all digits in given no. — while
- all the nos. divisible by 3 or 5 which — for are smaller than n.

- find factors of given no. — for

L3.11 Nested for loop

- s = 'VIBGYOR'
```
for i in range(7):
    print(s[i])
```
print(s[i])

- s = t = 'VIBGYOR'
```
count = 0
for i in range(7):
    for j in range(7):
        print(s[i], s[j])
        count = count + 1
print(count)
```
} nested loop

L3.12 Tutorial on nested loops

Q. find all prime nos. < than entered no.

```
n = int(input())
if (num > 2):
    print(2, end = ' ')
for i in range(3, n):
    flag = false
    for j in range(2, i):
        if (i % j == 0):
            flag = false
            break  ⟶ break the inner loop
        else:
            flag = true
    if (flag):
        print(i, end = ' ')
```

Q. Find the total profit / loss

```
Id = input()
while (Id != -1):
    trade = int(input())
```

```
profit = loss = 0
while (trade != 0):
    profit = profit + trade
    loss = loss + trade
print ("Profit for Id", Id, "is", profit - loss)
Id = input()
```

Eg. Find the day wise for the entered duration of time

```
days = int(input())
for i in range(1, days+1):
    total = 0
    rainfall = int(input())
    while (rainfall != -1):
        total = total + rainfall
        rainfall = int(input())
    print(rainfall)
```

Eg. find the length of longest word from the set of words entered by the use

```
word = input()
max = 0
while (word != -1):
    count = 0
    for letter in word:
        count = count + 1
    if (count > max):
        max = count
    word = input()
print(word, count)
```

L3.13 Break, continue and pass

```
email = input()
for c in email:
    if (c == '@'):
        break
    print(c, end='')
```

(exit loop in CT)

→ stops the for loops and come out to line 7

- email = input()
for c in email:
    if (c == '@'):
        print('')
        continue
    print(c, end='')

instead of stop full for loop, it simply jumps to next iteration by skipping to the next iteration

- for x in range(11):
    if (x % 3 == 0):
        print(x)
    else:
        pass

its a null statement, with no operation as such

13.14 Conclusion
- manage to talk with if, for, while
- logical thinking with these concepts
- try new ques.

Week-4

L4.1 Introduction
- practice programs
- 5-6 ques → revisited in fourth-coming weeks
- birthday paradox, search, sort, matrix operations
⇒ - code the same program multiple times

L4.2 Warmup with lists
- l = [1, 7, 4] → print(l)
- l.append(1024) → l = [1, 7, 4, 1024]
- l.remove(1) → l = [7, 4, 1024]
- if there are 2 same, then l.remove will remove the first occurence.
- x = [] ⇒ x.append(l) ⇒ [[7, 4, 1024]]
- m = (10, 20, 30) → " → [[7, 4, 1024], [10, 20, 30]]
- x.append([100, 101, 102])
- M = [] ⇒ M.append([1, 2, 3]) / ([4, 5, 6]) / ([7, 8, 9]) → [[1, 2, 3], [4, 5, 6], [7, 8, 9]
  print(M[0])
  print(M[0][0]) ⇒ 1
  print(M[2][2]) ⇒ 9
  This creates 2-D data.

L4.3 Bday Paradox
- import random ⇒ print(random.random())
  l = [] ⇒ l.append(random.random())
  for i in range(100)
  l.append(random.randint(1, 10))
- for i in range(30):
  l.append(random.randint(1, 365))
- l.sort() ⇒ print(l)

```
i=0, flag=0
while (i <= len(l) - 1):
    if (l[i] == l[i+1]):
        print ('Repeats'), l[i], l[i+1]
        flag = 1
        break
    i = i+1
if (flag == 0) : print ('There is no repetition')
```

- Run the experiment, on the computer, and say it is indeed true ⟹ Paradox is truth.

L4.4 Naive Search in a list

```
import random
l = []
for i in range (10000):
    l.append (random.randint (1, 10,00,000))
flag = 0
a = int (input ())  → while (a != -1)
for i in range (len(l)):
    if (a == l[i])
        print ("Found")
        flag = 1
        break
if (flag == 0):
    print ('Not Found')
```

L4.5 The obvious sort

```
l = [12, 10, 7, 18, 6, 42, 8, 35]
l.sort()
print(l)
```

Try logic sheet on paper

```
x = []
while len(l) > 0
min = l[0]
for i in range (len(l)):
    if l[i] < min:
        min = l[i]
x.append (min)
x.remove (min)
```

```
print (l)
print (x)
```

CLASSTIME Pg. No.
Date / /

L4.6 Dot Product

- `l = [...]` `l = random.sample(list(range(1000)), 100)` (import random)

```
sum = 0
for i in range(len(l)):
    sum = sum + l[i]
print(sum)
```

- Dot prod.

$x = [1, 7]$
$y = [8, 6]$

$DP = (1 \times 8) + (7 \times 6)$
$= 8 + 42 = 50$

```
sum = 0
for i in range(len(x)):
    sum = sum + (x[i] * y[i])
```

⇒ This will work only when 2 lists are of same length.

L4.7 Matrix Addition

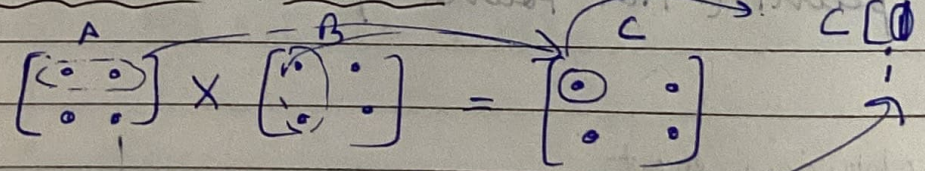
dim=3

$u_1 = [1, 2, 3]$ $s_1 = [1, 2, 1]$
$u_2 = [4, 5, 6]$ $s_2 = [6, 2, 3]$
$u_3 = [7, 8, 9]$ $s_3 = [4, 2, 1]$

A = [] B = []
A.append(u_1) B.append(s_1)
(u_2) (s_2)
(u_3) (s_3)

```
C = [[0,0,0], [0,0,0], [0,0,0]]
for i in range(dim):
    for j in range(dim):
        C[i][j] = A[i][j] + B[i][j]
```

L4.8 Matrix Multiplication - 1



A[0] . B[0]

row column

$(A[0][0] \cdot B[0][0]) + (A[0][1] \cdot B[1][0])$
i k k j i k k j

k = k + 1

L4.9 Matrix Multiplication - 2

- a_1, a_2, a_3 b_1, b_2, b_3 $\Rightarrow$ A=[] B=[]
  dim = 3 C=[[0,0,0], [0,0,0], [0,0,0]]

C[i][j] is the dot prod. of i^{th} row of A and j^{th} colm of B

```
for i in range (dim):
    for j in range (dim):
        for k in range (dim):
            C[i][j] = A[i][k] * B[k][j] + C[i][j]
print(C)
```

- import numpy
  X = numpy.mat(A)
  Y = nump.mat(B)
  print(X * Y)

L4.10 Conclusion

- try w/o looking into video
- practicing programmes
- get familiarised with these things

Week 5

L5.1 Intro to functions

- def add(a,b):
  ans = a+b
  print(ans)

  add(1,6)

- def sub(a,b)
  ans = a-b
  print(ans)

  sub(10,8)

- we're creating functions
- ~~dist~~ def discount(cost,d):
  ans = cost - (cost * (d/100))
  print(ans)

  discount(100,2)

- def add(a,b):
  ans = a+b
  return ans

  ⇒ Functions help in mind.

- def discount(cost,d)
  ans = cost - (cost * (d/100))
  return ans

x = c = int(input())
y = disc = int(input())
print(discount(x/c, y/disc))

L5.2 More eqs. on functions

- def list_min(l)
  mini = l[0]
  for i in range(len(l)):
  if (l[i] < mini):
  mini = l[i]
  return mini

  l = [1,2,3,-10,4,6]
  print(list_min(l))

  def first_ele(l):
  return l[0]

  x = [1,2,3]
  first_ele(x)

  ] same for list max too.

- def list_appendbefore(l, z):
```
    newl = []
    for i in range(len(z)):
        newl.append(z[i])
    for i in range(len(l)):
        newl.append(l[i])
    return newl
```
for append and first take (l[i]) then (z[i])

print(list_appendbefore(l, z))

- def list_average(l)
```
    sum = 0
    for i in range(len(l))
        sum = sum + l[i]
    ans = sum/len(l)
    return(ans)
```
⟹ Modular Approach to Programming

L5.3 Sorting using functions

```
def obvious_sort(l)
    x = []
    while(len(l) > 0):
        mini = l[0]
        for i in range(len(l)):
            if l[i] < mini:
                mini = l[i]
        x.append(mini)
        l.remove(mini)
    return x
```
Do we want 2 functions → find min → then append and remove

L5.4 Matrix multiplicaⁿ using funcⁿ

```
def initialize_mat[dim]:
    C = []
    for i in range(dim):
        C.append([])
    for i in range(dim):
        C[i].append(0)
    return(C)
```
→ Organize → modular } funcⁿ

C[0].append([0,0,0])

for i in range(dim):

```
def dot_product(u, v):
    dim = len(u)
    ans = 0
    for i in range(dim):
        ans = ans + (u[i] * v[i])
    return ans

def row(M, i):
    dim = len(M)
    l = []
    for k in range(dim):
        l.append(M[i][k])
    return l

def column(M, j):
    dim = len(M)
    l = []
    for k in range(dim):
        l.append(M[k][j])

def mat_mul(A, B):
    dim = len(A)
    C = initialize_mat(dim)
    for i in range(dim):
        for j in range(dim):
            C[i][j] = dot_product(row(A, i), column(B, j))
    return C
```

scope of var

25.5 Theoretical Intro to Recursion

Comp. Interest 10% → $1 + \frac{10}{100}$

2000 — I year → (2000)(1.1) = 2200 f(1)

II yr → (2200)(1.1) = 2420 f(2)

III yr → (2420)(1.1) = 2662 f(3)

$f(n) = (2000)(1.1)^n = f(n-1) \times 1.1$

CLASSTIME Pg. No.
Date / /

Sum(n) = sum(n-1) + n n! = (n-1)! x n

L5.6 Recursion - An Illustration

```
n = 10 , ans = 0
for i in range (n):
    ans = ans + (i+1)
print (ans)
```
} naive method

```
def sum(n):
    ans = 0
    for i in range (n):
        ans = ans + (i+1)
    return ans
```
} func^n method

```
def sum (n):
    if (n==1):
        return 1
    else:
        return n + sum (n-1)
```
} recursion method

```
def comp (p, n):          # interest = 10%.
    if (n==1)
        return p*(1.1)
    else:
        return (comp (p, n-1))*(1.1)
```

```
def fac (n):
    if (n==1)
        return 1
    else:
        return (fac(n-1))* n
```

L5.7 Types of Func^n Arguments

```
def add(a, b, c):
    return (a+b-c)
```
→ Positional Arguments

print (add (20, 30, 40))

print (add(a=20, b=30, c=40)) → Keyword Argument

⇒ Error ⇒ missing positional arguments

```
def add (a, b=20, c=30):
    return (a+b-c)
```
→ Default arguments

print (add (40))

CLASSTIME Pg. No.
Date / /

L5.8 Scope of Variables

- call-by-value & scope-of-variable
- global & local variable → but def-(x) parameter only when it is called
  - ↳ it is accessible throughout the program
  - ↳ keyword ⇒ global use it inside func. def then it will take value from anywhere in that code.

```
def myfunc()
    global x
    x = x * 2
    print(x)
x = 5
myfunc()
print(x)
```

L5.9 Type of Func^ns

- print() → it is also a func^n
- print(), input(), len() ⇒ In-built func^ns
  → part of python itself
- log(), sqrt(), random(), randrange(), calendar(), month() → Library Func^ns
  ↳ only work when library is imported
- upper(), lower(), strip(), count(), index(), replace()
  ↳ String Methods / Func^ns
- def Square(x) ⇒ User-defined func^ns
- parantheses (()) is always followed by the func^n

L5.10 Tutorial on func^ns

- cal. func^n → no. of upper case letters, lower case letters, total no. of charac., & no. of words

```
sentence = input()
uletters = upper(sentence)
print(uletters)
```

```
def upper(s):
    upper = 0
    for c in s:
        if (c.upper()):
            upper += 1
```

CLASSTIME Pg. No.
Date / /

```
l_letters = lower(sentence)
print(l_letters)

def lower(s)
    lower = 0
    for c in s
        if (c.islower()):
            lower += 1
    return lower

chars = character(sentence)
print(chars)

def character(s)
    chars = 0
    for c in s:
        chars += 1
    return chars

nwords = no_of_words(sentence)
print(nwords)

def no_of_words(s):
    words = 1
    for c in s:
        if (c == ' '):
            words += 1
    return words
```

- Write a python code → area & perimeter of circle & rec.

```
a = input('what is the shape ?')
b = input('what do u want? A or P')
if (a == 'circle' and b == 'area'):
    c = int(input('radius'))
    print(c_area(c))
elif ... circle, perimeter
elif ... rec., area
elif ... rec., perimeter
elif (a == 'exit' or b == 'exit'):
    print('You want to exit or put proper polygon')
    Exitloop

def c_area(a)
    return (π*a**a)
def r_area(l,b)
    return (l*b)
def r_perimeter(l,b)
    return 2*(l+b)
def c_perimeter(a)
    return (2*π*a)
```

- Write a python code, whether input coordinates form Δ or not.

```
x1, y1   x2, y2   x3, y3  => float(input())
d1 = distance(x1, y1, x2, y2)
d2 = "        (x2, y2, x3, y3)
d3 = "        (x3, y3, x1, y1)

def distance(x, y, a, b):
    return
    (((x-y)**2)+((a-b)**2))**1/2
```

```
if d1 > d2:
    if (d1 > d3):
        isTeria (d1, d2, d3)
    else:
        isTeria (d3, d1, d2)
elif d2 > d3:
    isTeria (d2, d1, d3)
else
    isTeria (d3, d1, d2)
```

```
def isTeria (max, a, b)
    if ((a+b) > max):
        print ('Δ')
    else:
        (not Δ')
```

Eg. (0,0), (0,1), (1,0) ⇒ Δ
(2,3), (3,2), (2,3) ⇒ not Δ

Week - 6

CLASSTIME Pg. No.
Date / /

66.1 Lists and Sets:

- l = list(range(10))
- l[0] ⇒ 0
- l.append('ABC')
- 5 in l
  ⇒ True

Car	bus
Mileage (5) ✗	Mileage ✗ (50) ✓
Mobile	

Trade off

- y = {1, 7, 6, 2, 8, 1} ⇒ type(y) ⇒ set
  print(y) ⇒ {1, 7, 6, 2, 8} # repetition not allowed
  2 in y ⇒ True
- s = set(range(10))
- to create list & set ⇒ list takes ~~more~~ less time
- when finding 5th ⇒ set takes no time almost
- import sys
  l = [], l_1 = [0], l_2 = [0, 1]
  sys.getsizeof(l,)
- x = list(range(100)) s = set(range(100))
  sys.getsizeof(x) ⇒ 856 (s) ⇒ 8408
- x[0] ⇒ 0 s[0] ⇒ error
- z = {1, 2, 3, 4, 5}
  ⇒ z.add(6)

66.2 dictionaries

- l , len(l) , l.append(100) , l[3] = 100
- d = {}
  d['a'] = 123
  d['b'] = 246
  ⇒ for search d['b']
- malgudi = ['gt', 'mas', ...]
- d = {}
  for x in ~~malgudi~~s :
  d[x] = 0
  for x in malgudi :
  d[x] = d[x] + 1

```
l = [1, 2, 3, 4]
for x in l :
    print(x)
```

```
max = 0 , answer-word = ' '
d = {}
for x in s:
    d[x] = 0
for x in malgudi:
    d[x] += 1
    if (d[x]) > max:
        max = d[x]
        answer-word = x
```

- d = {}
  d['puneet'] = [90, 91, 92]
  d['sudhansan'] ⇒ 90, 91, 92
  d['sudhansane'][0] = 90

L 6.3 Tuples

- t = (2, 7, 18)
  ⇒ unchangeable but list can be ∆ed
  ⇒ t[0] ⇒ 2 , same as list
- t = tuple (range(10))
  Lmuse when you don't want to ∆ it
- import string
  print (string.ascii_lowercase)
- alpha = tuple (list(s)) ⇒ unchangeable
  x = 'abcd %## 123'
  l = list (x)
  u = []
  for p in l:
      if p in alpha:
          u.append (p)

u use alpha as a lookup to do other things

- l = list (range(10)) ⇒ l.__sizeof__() ⇒ 120
  t = tuple (range(100)) ⇒ t.__sizeof__() ⇒ 104

CLASSTIME Pg. No.
Date / /

11.4 More on Lists

- $l_1 = [1, 2, 3]$ $l_2 = [10, 20, 30]$
  $l_1 + l_2 \Rightarrow [1, 2, 3, 10, 20, 30]$
  $l_2 + l_1 \Rightarrow [10, 20, 30, 1, 2, 3]$ — concatenation
- $l = [1, 2, 3] * 5$
  print (l) — multiplication
- $l_1 = [1, 2, 3]$ $l_2 = [1, 2, 3]$ $l_3 = [1, 3, 2]$
  $l_1 == l_2 \Rightarrow T$ $l_2 == l_3 \Rightarrow F$ $l_3 > l_2 \Rightarrow T$
- Python list is mutable.
  $l = [1, 2, 4] \Rightarrow l[2] = 3 \Rightarrow$ print (l) $\Rightarrow [1, 2, 3]$
- $l_1 = [1, 2, 3]$
  $l_2 = l_1$ → $l_1[0] = 100$, print (l_1, l_2) | l_1 & l_2 both gets changed
- $l_1 = [1, 2, 3]$
  $l_2 = list(l_1)$
  $l_3 = l_1[:]$
  $l_4 = l_1.copy()$ | how to create a new memory location for the same and equal lists
- Mutable things ⇒ lists ⇒ call by reference
- $l = [1, 2, 3]$

List methods —

- l.append (4)
- l.remove (2)
- x = l.copy() ⇒ print (x, l)
- l.insert (2, 9) ⇒ 9 at 2nd index
- l.pop (0) ⇒ remove anything at other posn

- l.sort() → ascending order list
- l.reverse() ⇒ palindrome creation

CLASSMATE Date / /

L6.5 More on tuples

- Tuples are immutable, modification not possible
- t = (1, 2, 3) | packing
  print (t, type(t))
  x, y, z = t | unpacking
  print (x, y, z)
- t = (10,) | otherwise single element becomes int
  print (t, type(t))
- t = ([1, 2], [a, b]) | if list in tuple, then list can be modified
  t[0][0] = 10
  ↳ Hashable
- t_1 = (1, 2, 3) ⇒ hashable (immutable)
  t_2 = ([1, 2, 3],) ⇒ not hashable (mutable)

L6.6 More on dictionaries

- d = {'key' : 'value'} → everything
  ↳ immutable and hashable entities
- d = {0:0, 1:1, 2:4, 3:9, 4:16} } iterate over a dictionary
  print (d)
  for key in d:
  print (key, d[key])
- print (d.keys())
  print (d.values())
  print (d.items()) } internally uses tuples

L6.7 More on Sets

- set = {1, 2, 3, 4, 5}
- duplicate values aren't allowed
- A = {1, 3, 5}
  B = {1, 2, 3, 4, 5}
  print (A.issubset(B)) ⇒ True
  print (B.superset(A)) ⇒ True

- $A = \{1, 2, 3\}$ $B = \{3, 4, 5\}$

⇒ A.union(B) ou A | B

⇒ A.intersection(B) A & B

⇒ A.difference(B) A - B

CLASSTIME Pg. No.
Date / /

Week-8

L8.1 Intro. to week 8 intro to recursion

1) recursion
2) more programming (deep)
3) file handling

Eg. R(100) = one person and R(99)
R → runners

Eg. vessel(10) = clean vessel + vessel(9)
V(10) = 1 + V(9) ⇒ out sourcing
V(9) = 1 + V(8)

Eg. Covid infection is recursive
A(n) = sth. + A(n-1)

L8.2 Recursion: a simple question

L = [11, 12, 16, 2, 3 ... , 0, 5, 6]
check 0 in L → False / True

```
check 0 (L):
    if L(0) == 0
        return True
    else:
        check 0 (L[1: n-1])
```

Pseudocode

→ check the first element and outsource the check of the rest of the list

L8.3 Recursion: find 0 in the list

```
def check 0(L):
    if (len(L) == 0):
        return 0
    if (L[0] == 0):
        return 1
    else:
        return check 0(L[1: len(L)])
```