Week 2

Introduction to the python

Console : check comands
Input/output seperation

Editor : Where we type our code - python code

* Printing Statements in python

>>> print (variable/number/whatever)

Eg:-
```
>>> a = 2
>>> b = 3
>>> c = a+b
>>> print (c)
5
```

Note Here we need to observe

In [8] >>> c
Out [8] : 5

but In [8] >>> print(c)
5

* Variables in python

a = apple
b = ball
c = cat

```
>>> print (" Hello c ")
>>> Hello c
```

```
>>> print ("Hello", c)
>>> Hello cat
```

Note: Variables are mutable - latest value of the variable will be displayed.

* Our first program.

```
name = input ("Enter the name")
print ("Hello world", name)
```

▷ click on run

output: Hello world prateek.

Changing the input to int: (I/P to different datatypes)

```
value = input()
valueint = int(value)
```
} Explicit conversion.

```
value = int(input())
```
} direct conversion.

* Discount calculation

```
cost = int(input("Enter the cost"))      # method 1
cost = input("Enter cost")               # method 2
d = int(cost)

ansum = 0.9 * d
print("The cost after discount is", answer)
```

* if condition

Syntax:-

```
if (condition):
    .statement1
elif (condition):
    statements
else:
    statements.
```

* Introduction to loops

Loop - Entity which helps to perform things continuously

→ different looping statements Ex:- for, while.

Syntax:

```
for (key) in range (variable/number):
    do something!!.
```

key may be any variable considered.

```
>>> for x in range (10):
        print ('hi')
```

print 'hi' 10 times. (0 to 9) times so the range calculates things according to · from and upto (0 to n-1)

>>> for x in range (0, 201, 2):
    print (x)

* Sum of numbers - loop

```
sum = 0
for number in range (3):
    sum = sum + number
print (sum)
```

>>> 3

Week 3 Cool ideas - Part 1

★ Lists Part 1 - Introduction

data structure - way of organizing data.

listname = [variables]

★ List manipulation - Part II

Indexing start from 0

ⓐ listname.append("item")	("0:[")
ⓑ listname.insert(index, "object").	(1,"0:[")
ⓒ listname.count(ages) - len(ages).	
ⓓ listname.sort() - ascending order	
ⓔ listname.reverse()	

★ Slicing - to extract subset of a list.

#listname[start : end + 1]

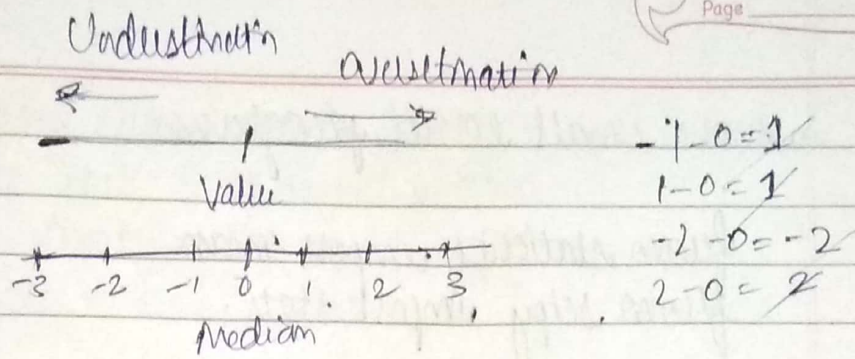
["from but not including"]

default start index = 0
default end index = len(list).

★ Fizz Buzz problem.

★ Crowd computing.

→ proposed by Francis Galton
→ Ox weight judging competition.
→ 800 tickets
→ weight of ox = 1198 lbs
→ median of all estimates = 1207 lbs



→ . Overestimation − Underestimation = 0

→ Other aggregate measure − These aggregate measures

• Trimmed mean

→ 10% trimmed mean

→ Remove 10% smallest and 10% largest values from sample

→ calculate sample of mean.

• Wisdom of Crowds. Collective opinion > Expert Opinion

Calculating the gems-trimmed mean.

```
Estimates = [... . ---- -]
Estimates.sort()
tv := int(0.1 * len(Estimates))
Estimates = Estimates[tv:]  → Estimates = Estimates[:len(Estimates)-tv]
# for i in range(len(Estimates)):
    print(Estimates[i])

print(mean(Estimates))
```

→ to do this.

from statistics import mean.

P.T.O

→ more small easier program -

```
from statistics import mean
from scipy import stats.

Estimates = [ ]
Estimates.sort()

m = stats.trim_mean (Estimates, 0.1)
print (m)
```

• Plotting no

```
import matplotlib.pyplot as plt
plt.plot (1,2,3,4).
```

} y axis

Similarly 2 cartesians :-

```
plt.plot ([1,2,3,4], [1,4,9,16])
```

(x, y)

Labelling axises :-

```
plt.ylabel (" - - - ")
plt.xlabel (" - - - - ")
```

Different fashion for plotting :-

```
plt.plot (x, y, 'ro')      - red dot
plt.plot (x, y, 'r--')     - red dashes.
plt.plot (x, y, 'bs')      - blue squares.
plt.plot (x, y, 'g^')      - green triangles.
```

```
# plotting the Estimates list.
import stat  # import mean
import matplotlib.pyplot as plt
Estimates = [...]

import statistics
import matplotlib.pyplot as plt
Estimates = [                    ]
y = []

Estimates.sort()
tv = int(0.1*(len(Estimates)))
Estimates = Estimates[tv:]
Estimates = Estimates[:len(Estimates)-tv]
for i in range(len(Estimates)):
    y.append(5)

plt.plot(Estimates, y, 'r--')
plt.plot([statistics.mean(Estimates)],[5],'ro')
plt.plot([statistics.median(Estimates)],[5],'bs')
plt.plot([375],[5],'g^')
```

Jumbled words – permutation.

```
import random
def play():
    p1name = input('player1, Please enter your name')
    p2name = input('player2, Please enter your name')
    pp1 = 0
    pp2 = 0          # points = 0
    turn = 0
    while(1):
        picked_word = choose()
        qn = jumble(picked_word)
        print qn
```

#player 1

```
if turn % 2 == 0:
    print(p1name, 'Your turn').
    ans = input ('What is on my mind ?')
    if ans == picked_word:
        pp1 = pp1 + 1
        print ('Your score is :', pp1)
    else:
        print ('Better Lunck next time. I thought:',
                picked_word)
    c = int (input('press 1 to continue 0 to end'))
    if c == 0:
        frank (p1name, p2name, pp1, pp2)
        break.
```

#player 2

```
else:
    print (p2name, 'Your turn')
    ans = input ('What is my mind ?').
    if ans == 'picked_words:
        pp2 = pp2 + 1
        print ('Your score is :', pp2)
    else:
        print ('Better Luck next time. I thought:',
                picked_word)
    c = int (input('Press 1 to continue 0 to quit'))
    if c == 0:
        frank (p1name, p2name, pp1, pp2)
        break.

turn = turn + 1
```

```
def choose():
    words = ['rainbow', 'computer', 'science', 'programming',
             'mathematics', 'player', 'condition', 'reverse', 'water']
    pick = random.choice(words)
    return pick

def jumble(word):
    random.sample(word, len(word))
    jumbled = "".join(random.sample(word, len(word)))
    return jumbled

def thank(p1n, p2n, p1, p2):
    print(p1n, 'Your score is:', p1)
    print(p2n, 'Your score is:', p2)
    print('thanks for playing')
    print('Have a nice day')
```

- Evolution is a random process
- Genetic algorithm.

★ Theory of evolution — File handling.

Opening and read writing files.

```
with open("file1.text", "a") as myfile:
        print(myfile.read())
        myfile.write("I am fine")
myfile.close()
```

w - write mode
r - read mode
a - append mode
r+ - read & write

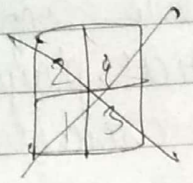
```
random.randint(1,5)
random.randrange(1,5)
random.random()
random.randrange(1,10,2)
```

Week 4

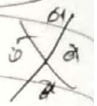
Cool Ideas (Part 2)

1, 2, 3, 4, 5, 6, 7, 8, 9 (China)

4	9	2
3	5	7
8	1	6



fact 1 is always in middle row last colum



Magic Square - Brute force → Hit and trial

Magic number = $M = n(n^2+1)/2$

for 3×3 $\Rightarrow M = 3(3^2+1)/2$

$= 3(10)/2$

$= 30/2 = 15$

Steps :-

1) Any magic square, 1 is located at $(n/2 ; n-1)$ (row, column)

2) Let position of 1 be (p, q) then 2 is located at $(p-1, q+1)$ position.

if row position beoms -1 then make $n-1$

if colum position beoms n then make it 0

3) If calculated position already contains a number then

decrement column by 2

increment row by 1

4) If anytime row position becomes -1 and column becoms n. then swatch to $(0, n-2)$

magic square computation program.

```
def magic_square(n):

    magicSquare = []
    for i in range(n):
        l = []
        for j in range(n):
            l.append(0)
        magicSquare.append(l)

    i = n//2
    j = n-1

    num = n*n
    count = 1

    while(count <= num):
        if(i == -1 and j == n):
            j = n-2
            i = 0
        else:
            if(j == n):
                j = 0
            if(i < 0):
                i = n-1

        if(magicSquare[i][j] != 0):
            j = j-2
            i = i+1
            continue

        else:
            magicSquare[i][j] = count
            count += 1
```

```
        i = i - 1
        j = j + 1

for i in range(n):
    for j in range(n):
        print(magicSquare[i][j], end = " ")
    print()

print("The sum of each row/column/diagonal is: "
                + str((n**2 + 1)*n//2))

magic_square(3)
magic_square(11)
magic_square(7)
```

---

* Dobble game – Spot the similarity

only 1 similar

only 1 similar

- Alphabets → Symbols
- Depict no of symbols
- only 1 common symbol.

program to find the common one.

```
import string
import random

symbols = []
symbols = list(string.ascii_letters)

card1 = [0]*5
card2 = [0]*5

pos1 = random.randint(0,4)
pos2 = random.randint(0,4)
```

```
samesymbol = random.choice(Symbol)
symbols.remove(samee.symbol).

if (pos1 == pos2):
    card2[pos1] = samesymbol
    card1[pos1] = samesymbol
else:
    card1[pos1] = samesymbol
    card2[pos2] = samesymbol
    card1[pos2] = random.choice(Symbols)
    symbols.remove(card1[pos2])
    card2[pos1] = random.choice(symbols)
    symbols.remove(card2[pos1])

i=0

while(i<5):
    if(i] = pos1 and i] = pos2):
        alphabet1 = random.choice(symbols)
        symbols.remove(alphabet1)
        alphabet2 = random.choice(symbols)
        symbols.remove(alphabet2)
        card1[i] = alphabet1
        card2[i] = alphabet2
    i=i+1

print(card1)
print(card2)

ch = input( )
if (ch == samesymbol):
    print("Right")
else:
    print("Wrong").
```

Week 5

Introduction to Dictionaries

>>> #key : #value (unique)

Syntax :-

dic_name ['keys'] = value

conv_factor['euro'] = 80
conv_factor['dollar'] = 60

print (conv_factor)
>>> {'dollar': 60, 'euro': 80}

list (conv_factor.keys())
list (conv_factor.values())

del conv_factor['dollar']

Dic = {keys : values}

- Audio Recognition

import speech_recognition as sr
audio_file = ("Soud.wav")

r = sr.Recognizer()

with sr.AudioFile(audio_file) as source:
    audio = r.record(source)

try:
    print ("Audio file contains " + r.recognize_google(audio))
except sr.UnknownValueError:
    print ("google speech recognition couldnt understand audio")