Python Programming

1) object oriented
2) interpretted
3) general purpose

Van Guido Rossum in 1990s in Netherlands

$gedit file.py

```
print ("Hello World")
```

$python3 file.py

Comment styles

line comment → # multiline → """ """

Performing Math Operations

Write a program that evaluates $\frac{10.5 + 2 \times 3}{45 - 3.5}$

```
print ((10.5 + 2 x 3)/(45-3.5))
```

→ Errors

Syntax, Runtime, Logical

Reading input from keyboard

```
variable = input ("Enter a value: ")
```

↳ value is string

```
variable = real (input ("Enter value: "))
```

or

```
variable = int (input ("Enter value: "))
```

Variables are used to reference the values that may be changed in the program.

Simultaneous assignment

var1, var2, ... varn = exp1, exp2, ... expn

Swap 2 numbers

```
x, y = eval(input("Enter 2 numbers:"))
x, y = y, x
print("number after swapping:", x, y)
```

Mathematical/Arithmetic Operators

→ +, -, *

→ / → float division 5/2 = 2.5

// → integer division 5//2 = 2

** → exponential → 2**2 = 4

% → remainder

→ Write a program to obtain minutes and remaining seconds for an amount of time in seconds

→ Assume a runner runs 14km in 45 minutes and 30 seconds. Write a program that displays the average speed in miles per hour.

→ Write a program that reads an integer between 0 and 1000 and adds all the digit in the integer

Write a program to compute distance between 2 points

Reads subtotal and gratuity rate and compute the gratuity rate

Subtotal = 10
gratuity = 15%
gratuity = 1.5
subtotal = 11.5

Augmented Assignment Operator

i += 8 ⇐ += ⇒ i = i + 8

-= i = i - 8 ⇒ i -= 8

*= i = i * 8 → i *= 8

/=

//=

%=

Comparison Operator

>, <, >=, <=, ==, !=

Boolean Variables

True False

>>> print (true) ⇒ 1

>>> print (false) ⇒ 0

>>> print (bool(0)) ⇒ false

>>> print (bool(4)) ⇒ True

Other than 0 all are true

Generating random numbers

→ randint (a, b) function can be used to generate the random integer between a and b inclusively.

>>> import random

>>> random.randint (0, 9)

>>> random.randint (0, 1) → 1

>>> random.randint (0, 1) → 0

>>> random.randrange(0,1)
→ displays 0 always

>>> random.random()
→ displays floating point number between 0.0 and 1.0

Program

1) Write a program to generate 2 single digit numbers randomly and display it to a student as a question "What is 1+7?" After student types an answer the program displays a message to indicate whether it is true or false.

→

```
import random
print("What is ", random.randint(0,7), "+", random.randint(0,7))
if (
```

⇒

```
import random
num1 = random.randint(0,9)
num2 = random.randint(0,9)
answer = eval(input("What is " + str(num1) + " + " + str(num2) + "? "))
print("num1+", num2, "=", answer, "is", num1 + num2 == answer)
```

first condition

comparison

This will return the value

Selection

if statement

```
if boolean expression :
    statement(s)
```

Two ways of if statement

```
if boolean_expressions :
    statement(s) for the true case
else :
    statement(s) for the false case
```

Nested if

```
if i > k :
    if j > k :
        print ("i and j are greater than k")
else :
    print ("i is less than or equal to k")
```

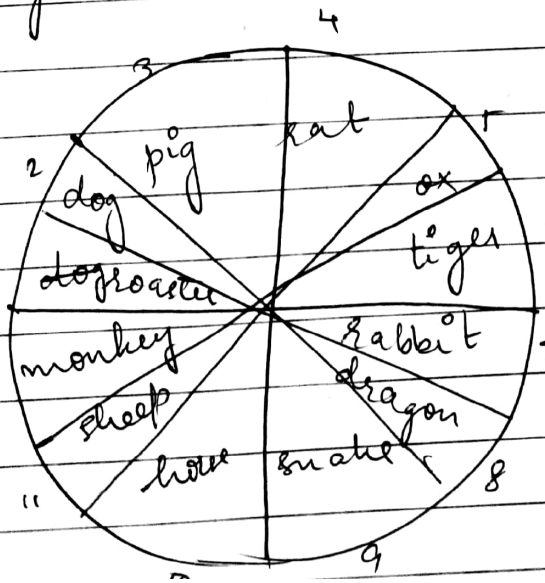
if-elif-else

```
if score >= 90 :
    grade = 'A'
elif score >= 80 :
    grade = 'B'
elif score >= 70 :
    grade = 'C'
elif grade >= 60 :
    grade = 'D'
else :
    grade = 'F'
```

Write a program to perform the addition subtraction ~~mathe~~ mul, div by taking input from the user

```
print ("Menu")
print ("
num1 = eval (input (print ("Enter number 1")))
num2 = eval (input (print ("Enter number 2")))
print ("Menu")
print ("1)Add \t 2)Sub \t 3)Mul \t 4)Div \t ")
choice = eval (input (print ("Enter option")))
if choice == 1:
    print ("Sum of num1 & num2 ", num1+num2)
elif choice == 2:
    print ("          ", num1 - num2)
elif choice == 3:
    - - -
elif choice == 4:
    - - -
else
    print ("Invalid choice")
```

Write a program to find out Chinese Zodiac Sign



2018 % 12 = 2

```
year = eval (input (print ("Enter the year:")))

zodia_sign = year % 12
 if zodiac_sign = = 0 :
      print ("monkey")
 elif zodiac_sign = = 1 :
      print ("Roaster")
 elif zodiac_sign == 2 :
      print ("dog")
 elif zodiac_sign = = 3 :
      print ("pig")
 elif zodia_sign = = 4 :

 elif   zs  = =  5

 elif   zs  - - = 6
```

e

Logical operator

and

or

not

1) Write a program which checks whether a number is divisible by 2 and 3, by 2 or 3 and by 2 or 3 but not both

Conditional Expression

exp if bool_exp1 else exp2

y=1 if x>0 else y=-1

max=num1 if num1>num2 else num2

→ ~~Associativity~~ Associativity is left to right in Python

Loop statements

while loop ~~condition~~ continuation condition :
    statement(s)

for i in range(initial value, end value)
    #loop body
for var in sequence :
    #loop body

for i in range(4,8):
    print(i)
→ 4,5,6,7 will print one less than the x value

for v in range(3,9,2): (3 is start, 9 is end, 2 is increment)
→ op: 3 5 7

for x in range(0,9):
    print(x)
→ 0 1 2 3 4 5 6 7 8

print() → jump to next line

print(i, end = ' ') → avoid \n

Files

file variable = open(filename, mode)

- r
- w
- a
- rb } read binary
- wb } write binary

```
outfile = open("file1.txt", w)
outfile.write("Barack Obama\n")
outfile.write("GeorgeBush\n")   → next line for printing
outfile.write("Bill.Clinton")
outfile.close()
```

Testing a files existence

```
import os.path
if os.path.isfile("file1.txt"):
    print("file1.txt exists")
```

Reading data

1)
```
infile = open("file1.txt", r)
print("(1) using read():")
print(infile.read())   → brings characters from file & prints
infile.close()
```

2)
```
infile = open("--", r)
print("(2) using read(number):")
s1 = infile.read(4)
print(s1);

s2 = infile.read(10)
print(s2)
print(repr(s2))   → even create spaces will
```

1 —
2 —
3 —

displays 4 characters

continues from previous printed

Scanned by CamScanner

3) infile = open("file1.txt", 2)
print("(3) using readline():")
line1 = infile.readline()
line2 = infile.readline()
line3
line4 =
print(repr(line1))
:
(repr(line4))
infile.close()

→ So for 4th will print blank

4) infile=open("file1.txt", 2)
print("(4) using readlines())
print(infile.readlines())
infile.close()

} will read all lines at once

Reading all data from a file

1) line = infile.readline()
while line != '':
# Process the line
# Read next line
line = infile.readline()

2) for line in infile: → operator in python
Process in line

Write a program to copy the contents of one file to another file. Also, display the number of lines and characters copied.

~~infile = open ("file1.txt")~~

→ import os.path → import sys
if os.path.isfile ("file1.txt"):
print ("file1.txt exists")

infile = open ("file1.txt", r);
for line in file:
lines = infile.readlines()

infile = open ("file2.txt", w)

→ import os.path
import sys

f1 = input ("Enter a source file ").strip()
f2 = input (" Enter a destination file ").strip()

if os.path.isfile(f2):
print (f2 + " already exits")
sys.exit()

infile = open (f1, "r")
outfile = open (f2, "w")

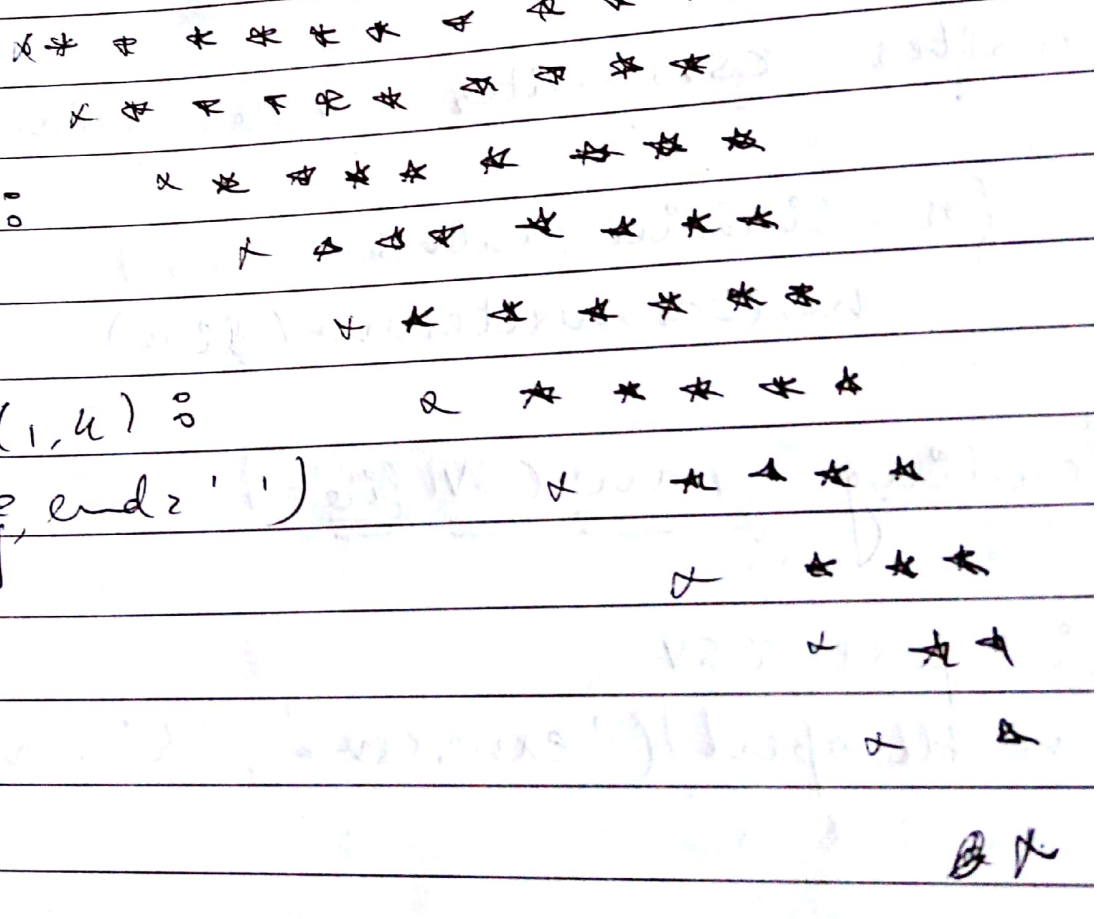
countlines = countChars = 0
for line in infile:

```
countline += 1
countchar += len(line)  # characters
                        # lines
print(countline, "Lines count", countchar,
      "characters copied")

infile.close()
outfile.close()
```

LAB Write a program to display the following pattern

```
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5
```



```
for i in range(1,7):
    k = 2*k - 1
    print()
for j in range(1,4):
    print(j, end = ' ')
```

Writing to CSV file

first_name	last_name	city
John	Deonza	Mangalore
Ram	Saxena	Delhi

```
import csv
data = ["first_name, last_name, city".split(","),
        "John, Deonza, Mangalore".split(","),
        "Ram, Saxena, Delhi".split(",")]

with open('exw.csv', 'w', newline='')
writer = csv.writer(csvfile, delimiter=',')
                                as csvfile:
                                    for line in data:
                                        writer.writerow(line)
OR
writer = csv.writer(csvfile, delimiter=',')

    for line in data:
        writer.writerow(line)

csvfile = open('exw.csv', 'w', newline='')

writer = csv.writer(csvfile, delimiter=',')

    for line in data:
        writer.writerow(line)
```

Reading from CSV file

```
import csv
with open('exw.csv', 'r', newline='') as
                                    csvfile:

read csv
read_csv = csv.reader(csvfile, delimiter=',')
```

```
for column in read CSV:
    print (column)
#print (column[0]) //to check what will be printed
```

OR

```
csvfile = open ('exm.csv', 'r', newline = ',')
```

OR

```
readcsv = csv.reader (csvfile, delimiter = ',')

for column in read CSV:
    print (column)
```

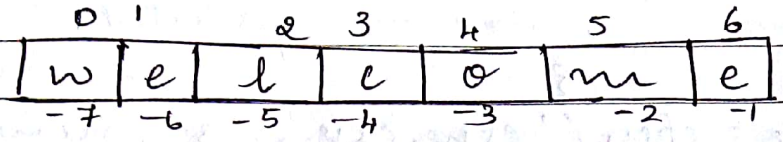
Strings

```
s = str ()  or  s = " "
s1 = str ("welcome")  or  s1 = "welcome"

>>> s = "welcome"
>>> len (s)
→ 7

>>> max (s)
→ o        based on ASCII value

>>> min (ws)
→ w
```

Scanned by CamScanner

Index Operator

welcome



s[index]

```
>>> s = "welcome"
>>> for i in range(0, len(s), 2):
        print(s[i], end = ' ')
```

>>> s[-1] → 'e'

>>> s[-1+len(s)] → s[6] → 'm'

>>> s[2] → 'l'

>>> s → welcome

slicing operator [start : end]

```
>>> s = "welcome"
>>> s[1:4]
```

→ 'elc' [start : end - 1]

>>> s[: 6] → s[0 : 6]

'welcom'

>>> s[4 :] → s[4 : 7]

→ ome

>>> s[1 : -1]

→ 'elcom'

Concatenation (+) and repetition (*) Operators

```
>>> s1 = "welcome"
>>> s2 = "Python"
>>> s1 + "to" + s2
```

→ welcome to Python

```
>>> s1 * 3
```

→ welcomewelcomewelcome

also can be given as 3 * s1

The in and not in

```
>>> s1 = "welcome"
>>> "come" in s1
```

→ true

```
>> "come" not in s1
```

→ false

Comparing strings

```
>>> "green" == "glow"
```

→ false

```
>>> "green" != "glow"
```

→ true

```
>>> "green" > "glow"
```

→ true

```
>>> "green" >= "glow"
```

→ true

} ascii values are same

>>> "gloa" > "glow"
→ false

} ascii of gloa is less than glow

Compares the ascii values

Iterating a string

s1 = "welcome"

```
for ch in s1:
    print(ch)
```

Testing the strings

>>> s = "abc123"
>>> s.isalnum()
→ true

} for alphanumeric

>> s.isalpha()
→ false

>>> s.isdigit()
→ false

>>> s.isidentifier()
→ true

>>> s.istoken()
→ true

>>> s.isupper()
→ false

>>> s.isspace() → checks only for spaces like s = " "
→ false

Searching for substring

```
>>> s = "welcome"
          0 1 2 3 4 5 6
>>> s.endswith("me")
→ true

>>> s.startswith("G")
→ false

>>> s.find("come")
→ 3

>>> s.rfind("e")          // rfind not defined
→ 6

>>> s.count("e")
→ 2

>>> help str
→ gives list of string manipulation function with syntax
```

Converting strings

```
>>> s = "welcome to python"

>>> s.capitalize()
  Welcome to python
```

} capitalize the first letter of first letter

```
>>> s2 = s.title()
>>> s2
  Welcome To Python
```

>>> s1 = 'New England'
>>> s2 = s1.swapcase()
>>> s2
→ new england

>>> s1 = s1.upper()
>>> s1
NEW ENGLAND

>>> s5 = s1.swapcase()
→ NEW ENGLAND

>>> s6 = s1.replace("England", "Heaven")
>>> s6
New Heaven

>>> s1
New England

Programs

1) Write a program to count and display the number of capital letters in a string.

→ st = eval(input("Enter the string"))
count = 0

~~s2 = s1.isupper()~~

~~s3 = s2()~~

~~if s1.isupper~~
for i in st:
    if isupper():
        count += 1
print("No of upper case letters in a string", count)

2) Write a program to check whether the given string is palindrome or not

→ s1 = input ("Enter the string")

- ~~rev = [::-1]~~
- rev = s1[::-1] // gives the string reverse

if s1 == rev:
    print ("string is a palindrome")
else:
    print ("string is not a palindrome")

3) Write a program to count the number of each vowel in a string.

→ s1 = input ("Enter the string")

4) Write a program to remove all punctuation from the string provided by the user.

p = '''!()-[]{};:\,<>./?@#$%^&*_~"'

Skipping white space characters from the string

```
>>> s = " welcome to Python "
        ↑ space
>>> s1 = s.lstrip()
→ 'welcome to Python '
```
→ removes space on left side of the string

```
>>> s2 = s.rstrip()
→ ' welcome to Python'
```
→ you remove the 'It' from this

```
>>> s3 = s.strip()
→ 'welcome to Python'
```
→ removes white space from both ends

List

→ 1) List can store a collection of data of any size. (string etc collection of character)
2) It is a sequence of elements defined by the list class.
3) Contains methods for creating, manipulating and processing lists
4) List is mutable (string is immutable)

Creating the lists

```
→ list1 = list()    # empty list
```
list1 ↓ name of the list

→ list 2 = list([2, 3, 4])

In Python no concept of array, hence we have list.

→ list 3 = list (["red", "green", "blue"])

→ list 4 = list (range (2,5)) # range within list

→ list 5 = ("abcd")

OR

→ list1 = [] #empty list
→ list 2 = [2, 3, 4]
→ list 3 = ["red", "green", "blue"]
→ list 4 = [2, "three", 4] # you can mix different datatypes

Functions for list

>>> list 1 = [2, 3, 4, 1, 32]

>>> len (list 1) # length of list 1
→ 5

>>> max (list 1) #maximum value
→ 32

>>> min (list 1) #minimum value
→ 1

(In string based on ascii values, but here the actual value)

```
>>> sum(list1)
→ 42

>>> import random
>>> random.shuffle(list1)        # randomly shuffles the list1 values
>>> list1
→ [24,1,2,32,3]
```

Index operator

```
>>> list1[-1]                    #negative index
→ 32

>>> list1[-3]
→ 4
```

List slicing [start:end]

```
      0  1  2  3  4  5
>>> list1=[2, 3, 5, 7, 9, 1]
>>> list1[2:4]
→ [5,7]

>>> list1[:2]
→ [2,3]

>>> list1[3:]                    # 6-1
→ [7,9,1]                       len(str)-1

>>> list1[1:-3]                  # -3+len(list1)
→ [3,5]                         -3+6 = 3

>>> list1[-4:-2]                 # -4+-26 = -62
→ [5,7]                         -2+6 = 4
```

+, * in/not in operator

```
>> list 1 = [2, 3]
>> list 2 = [1, 9]
>>> list 3 = list 1 + list 2
>>> list 3
→ [2, 3, 1, 9]

>>> list 4 = 3 * list 1        # replicating
>>> list 4
  [2, 3, 2, 3, 2, 3]

>>> list 2 = [2, 3, 5, 2, 33, 21]
>>> 2 in list 2
→ True                        # if present

>>> 2 not in list 2            # if not present
→ False
```

Traversing elements in for loop

1)
```
for i in list 2 :
    print (i)
```

2)
```
for i in range (0, len (list 2)) :
    print (list 2 [i])
```

Comparison

Comparing lists

```
>>> list1 = ["green", "red", "blue"]
>>> list2 = ["red", "blue", "green"]
>>> list1 == list2
→ false
```

because it compares each position element with the other corresponding element

```
>> list1 != list2
→ True
```

```
>>> list2 > list1
→ True
```

~~text~~ lexicographic comparison, ex: g comes before r

```
>>> list2 >= list1
→ True
```

```
>>> list2 < list1
→ false
```

```
>>> list2 < list1
→ false
```

List Comprehension

```
>>> list1 = [x for x in range(5)]
>>> list1
→ [0, 1, 2, 3, 4]
```

```
>>> list2 = [0.5 * x for x in list1]
→ [0, 0.5, 1, 1.5, 2]
```

>>> list3 = [x for x in list2 if x < 1.5]
→ [0, 0.5, 1]

9) Write a program to write and read the numeric data into a text file.

```
from random import randint
outfile = open("num.txt","w")
for i in range(10):
    outfile.write(str(randint(0,9))+" ")
outfile.close()

infile = open("num.txt","r")
s = infile.read()
numbers = [eval(x) for x in s.split()]
for number in numbers:
    print(number, end=" ")
infile.close()
```

int → convert to string and then write

eval → convert into int

LAB

Write a program to accept a string and display the resultant string in reverse order. The resultant string should contain all the characters at the even position of the accepted string ignoring blank spaces.

→ i/p: NMAMIT NITTE

0 1 2 3 4 5 6 7 8 9 10 11

o/p: TIIAN

Consider 2 strings s1 and s2 and display the merged string. The merged string should be capital letters from both the strings in the order they appear.

→ str1: I like C
str2: Mary likes Python

Merged string: I@LCMLP

Given a string containing both upper case and lower case letters. Write a program to count the number of repeated characters and display the maximum count of a character along with character.

→ i/p: ABaBcbGc

o/p: A 2
B 3
C 2
G 1

List methods

```
>>> list1 = [2, 3, 4, 1, 32, 4]
>>> list1.append(19)        // appends 19 to the list1
>>> list1
→ [2, 3, 4, 1, 32, 4, 19]
>>> list1.count(4)          // number of times 4 occures
→ 2
>>> list2 = [99, 100]
>>> list1.extend(list2)     // extends list1 by list2
→ [2, 3, 4, 1, 3, 2, 4, 19, 99, 100]
   0  1  2  3  4  5  6  7   8   9
>>> list1.index(4)          // index value of 4 (the first appearance of 4)
→ 2
>>> list.insert(1, 25)      // inserting at the given position
→ [2, 25, 3, 4, 1, 3, 2, 4, 19, 99, 100]
>>> list1.pop(2)            // pops second element
        ↳ index
→ [2, 25, 4, 1, 32, 4, 19, 99, 100]
>>> list1.pop()             // can be used without parameter (but pops the last element)
→ [2, 25, 4, 1, 32, 4, 19, 99]
```

```
>>> list1.remove(32)        # removes the particular element
>>> list1
→ [2, 25, 41, 4, 19, 99]

>>> list1.reverse()
→ [99, 19, 4, 41, 25, 2]

>>> list1.sort()
→ [1, 2, 4, 4, 25, 19, 99]
```

Splitting a string into a list

```
items="Jane John Peter Swam".split()
→ ['Jane', 'John', 'Peter', 'Swam']

d = "08/12/2018".split("/")
→ ['08', '12', '2018']
```

Inputting lists

```
list1 = list()
val = eval(input("Enter the names"))
list1 = val.split(",")
```

X

```
lst = []
print("Enter 10 ten numbers")
for i in range(10):
    lst.append(eval(input()))
```

OR

```
lst = []
```

```
s = input("Enter 10 numbers seperately by
    spaces in one line")
items = s.split()
lst = [eval(x) for x in items]
```

Two dimensional lists

1	2	3	4	5
6	7	0	0	0
0	1	0	0	0
1	0	0	0	8
0	0	9	0	3

(nested list)

```
matrix = [
          0 1 2 3 4
          [1, 2, 3, 4, 5], ← matrix[0]
          [6, 7, 0, 0, 0] ← matrix[1]
          [0, 1, 0, 0, 0], ← matrix[2]
          [1, 0, 0, 0, 8],          [3]
          [0, 0, 9, 0, 3]           [4]
         ]
```

```
matrix[0][0] = 1
matrix[4][4] = 3
```

Initialising the lists with input values

```
matrix = [ ]
rows = eval(input("Enter the number of rows"))
cols = eval(input("Enter the number of cols"))
for r in rows:
    matrix.append([ ])   #append empty rows
    for c in cols:
        ~~matrix.append(eval(input~~
        value = eval(input("Enter an element"))
        matrix[r].append(value)
print(matrix)
```

(1) ——[

import

from random import randint.

→ for random numbers

import random

}

!

(1) → matrix[r].append (random.randint(0, r

Printing lists

```
matrix = [[1, 2, 3], [1, 5, 6], [7, 8, 9]]
for row in range(len(matrix)):
    for col in range(len(matrix[row])):
        print(matrix[row][col], end=" ")
print()
```

OR

```
for row in matrix:
    for value in row:
        print(value, end=" ")
print()
```

Tuples
(values are fixed, not mutable)

→ Tuples are used for storing a fixed list of elements

→ Once a tuple is created you cannot add new elements, delete elements, replace elements or reorder elements in a tuple.

```
t1 = ()  # empty tuple
t2 = (2, 3, 5)
t3 = ("red", "green", "blue")
t4 = tuple([2*x for x in range(1,5)])

t4 = tuple("abc")
('a', 'b', 'c')

>>> t1 = ("red", "green", "blue")
>>> len(t1)
→ 3
>>> t2 = (2, 3, 4, 5, 6)
>>> max(t2)
→ 6
>>> min(t2)
→ 2
>>> sum(t2)
→ 20
>>> print(t2[0])          # can also use the index operator to access elements
→ 2
```

```
>>> t3 = t1 + t2
→ ('red', 'green', 'blue', 2, 3, 4, 5, 6)

>>> t2[2:4]
→ (4, 5)
>>> t2(-1)
→ 6
>>> 2 in t2
→ True

>>> for x in t1:
        print(x, end='')
    print()

>>> list1 = list(t2)
>>> list1
→ [2, 3, 4, 5, 6]

>>> t4 = tuple(list1)
>>> t5 = tuple(list1)
>>> t4 == t5
→ True
```

Dictionaries

→ stores collection of elements in form of key, value pairs.

creating dictionary

```
students = { }   #empty dictionary

students = {"4NM15CS010" : "John", "4NM15CS020" :
"Susan", "4NM15CS040" : "Peter"}
                  key          value
```

keys must be numbers or keys,
value can be anytime.

Adding to a dictionary

dictionaryName[key] = value

Ex: students["4NM15CS050"] = "Karthik"

Replacing a value

students["4NM15CS040"] = Girish
#replaces Peter

Deleting a value

del dictionaryName[key]

del students["4NM15CS050"]
#Karthik deleted

Printing

```
for v in students:
    for w in students:
        print(v+":"+w, end='')
print()

for key in students:
    print(key + ":" + students[key])
```

>>> len(students)
→ 3

>>> "4NM15CS010" in students
→ True

Equality Test

>>> d1 = {"red": 41, "green": 20, "blue": 10}
>>> d2 = {"green": 20, "red": 41, "blue": 10}
>> d1 == d2
→ True # elements aren't in ordered form, here order isn't considers, & compares only key values

Here, we cannot use >, <, >=, <=, *

Methods

>>> tuple(students.keys())
→ ('4NM15CS010', ...) # only the key will be displayed

>>> tuple(students.value())
→ ('John', ...) # only the values

>>> tuple(students.items())
('4NM15CS010': 'John', ...) # both key and values

>>> students.get("4NM15CS010")
'John' # display particular key's value

```
>>> students.pop ("HNM15CS010")
'John'                    # John is popped

>> students.popitem()
                          # randomly selects one and pops

>>> students.clear()      # clears all items at once.
{}
```

Q1) Write a program that repeatedly prompts the user to enter a capital for a state upon receiving the user input the program reports whether the answer is correct. Assume that states and capitals are stored in a dictionary. The program prompts the user to answer all the states capitals and displays the total correct count. The users answer is not case sensitive.

Q2) With a given integer number n write a program to generate a dictionary that contains {i: i*i} such that it is an integral number between 1 and N both included and then the program should print the dictionary.

Q3) Write a program that accepts a sequence of comma seperated numbers from the console and generate tuple which contains every number.

Q4) Write a program that accepts a sentence and calculate the number of letters and digits

Sets

s1 = set() # creates an empty set

s2 = {2, 3, 5}

s3 = set([4, 5, 6])

s4 = set([2 * x for x in range(1, 10)])

s5 = set("abc") → {'a', 'b', 'c'}

list(s2)
tuple(s2)
} converts the set

```
>>> s1 = {2, 4, 10}
>>> s1.add(12)
  → {2, 4, 10, 12}
>>> max(s1)
  → 12
>>> min(s1)
→ 2
>>> sum(s1)
  → 28
>>> 2 in s1
→ True
>>> s1.remove(12)
```

subset and superset

```
>>> s1 = {1, 2, 4}
>>> s2 = {1, 4, 5, 2, 6}
>>> s1.issubset(s2)
→ True
```

>>> s2.issuperset(s1)
→ True

>>> s1={1, 2, 5}
>>> s2={2, 5, 6, 7}

>>> s1 | s2 or >>> s1.union(s2)
→ {1, 2, 5, 6, 7}

>>> s1 & s2 or >>> s1.intersection(s2)
→ {2, 5}

>>> s1-s2 or >>> s1.difference(s2)
→ {1}

>>> s1={1, 2, 4}
>>> s2={1, 4, 2}
>>> s1==s2
→ True

>>> s1!=s2
→ False

a) Write a program that prompts the user to enter a text file, reads words from the file and displays all the non-duplicate words in ascending order.

b) Write a program to count the keywords from a python source code file.

9) Write a program that prompts the user to enter a text file name and displays the number of vowels and consonants in the file. Use a set to store the vowels.

Identity Operator

→ is
→ is not

```
>>> x = 5
>>> y = x
>>> x is y
  true
```

```
>>> id(x)
1908671
>>> id(y)
1908671
```

→ The address will be the same, copies aren't created, both point same address object.

```
>>> x = 5
>>> type(x) is int
  True
>>> type(x) is not int
  False
```

C → continue
Python → pass

```
for i in range(1,10):
    if i%2 == 0:
        pass
    else:
        print(i)
```

→ 1, 3, 5, 7, 9

OS module

Accessing and manipulating files and directories on disk.

Functions

		What it does
1)	chdir(path)	- Changes the current working directory to path.
2)	getcwd()	- Returns the path of the current working directory.
3)	listdir(path)	- Returns a list of the names in directory named path.
4)	mkdir(path)	- Creates a new directory named path and places it in the current working directory.
5)	remove(path)	- Removes the file named path from the current working directory.
6)	rename(old, new)	- Renames the file or directory named old, to new.
7)	rmdir(path)	- Removes the directory named path from the current working directory.
8)	~~os.path~~	

os.path module function

8)	exists(path)	- Returns true if path exists and false otherwise.
9)	isdir(path)	- Returns true if path is directory and false otherwise
10)	isfile(path)	- Return true if path is file else false

ii) getsize(path) - Returns the size of the object named by path in bytes

8) Write a program to print all of the names of files in the current working directory that have a .py extension.