python is a simple and easy to learn language which feels like reading simple english. This Pseudo Code nature of python makes it easy ~~cad~~ to learn and understand by beginner.

1) A module is a file containing code written by somebody else (usually) which can be imported and used in programs

2) Pip is the package manager for python
3) REPL - Read evaluate print loop

Single line Comment
''' Multi line
Comments '''

In VS
ctrl + /
Can make comments select and unselect

4) print(''' Hey
pal
Howdy ''')

★ playsound module (can help us to play song)
from playsound import playsound
playsound (' full path use double backslash ')

5) import Os
print(Os.listdir())

* Chapter 2
Variables and Numbers

```
a = " Akbar "
b = 512
c = 45.32
```

A variable is the name given to a memory location in program.

Keywords : Reserved words in python
Identifier : class function variable name

* Data Type
| Integers, floating point
| string, Boolean , None |

String ex → ' Akbar '
" Akbar " ''' Akbar '''

Code →
```
a = ''' harry '''
# printing the variable
print(a)
# printing the type of variable
print ( type (a) )
```
Output
```
class <str>
```

* Rules of naming variable

Date: / /
Page No.: TELCO

Operators
1) Arithmetic +, −, *, /
2) Assignment = += −=
3) Comparison == >, >= <, !=
4) Logical and or not

Type() function and Typecasting
Type function is used to find the data type of a given variable in python

a = "3534" → a = int(a)
print (a + 5)

Str(31) = "31" Int to Str Conversion
Int ("31") = 32 Str to Int Conversion
float(32) = 32.0 Int to float Conversion

☆ Input function
This function allows the user to take input from keyboard as string

a = input ("Enter name")
→ The output will always be String.

shift + 1
terminal clears

Date:
Page No.:

* chapter 3 String

* A data type in ~~String~~ Python
String is a sequence of ~~quote~~ characters enclosed in quotes.

```
a = " Akb "
print (b)
print (type(b)) .
```

| space ke type tab bhi we then include hoga |

* STRING SLICING
A string in python can be sliced for getting a part of the string

Note - We can access the string but can't change via indexing
(# name [0] = "ak" doesn't work)

```
print (name [0:3])
```

Output Akb $\overset{\text{"}}{\text{include}}$ hoga $\longrightarrow$ exclude hoga

* Negative indices
-1 (correspond to length -)
-1 of Akbar ⇒ r

```
A k b a r
0 1 2 3 4
-5 -4 -3 -2 -1
```

Date: / /
Page No. TELCO

* Slicing with skip value
we can provide a skip value as a
part of our slice like this
0 1 2 3 4 5 6
word = "amazing"
word [1:6:2] = mzn
↓
skip value

(strng = var name hai

* String functions
1) print (len (strng))
2) String endswith ("bar")
print (strng.endswith ("bar"))
3) print (strng.count ("O"))
4) print (Strng. capitalize ())
5) ~~print (strng. find (~~ ~~))~~
5) print (Strng. find ("upon"))
6) print (strng. replace ("Akbar",
"Replaced word"))

* Escape sequence
\n (new line) \t (tab)
\' (single quote) \\ backslash

Chap 4 - List and Tuples

Python list are containers to store a set of values of any data type

friend = ["Apple", "Akash", "7", "Earl"]
print(friend[1])

Note - we can create a list of item with diff type
print(friend[0:2])

* List indexing / slicing.
* List method
Consider the following list
L1 = [1, 8, 7, 2, 21, 15]

1) L1.sort(): update list to [1, 2, 7, 8, 15, 21]
2) L1.reverse: " " [15, 21, 2, 7, 8, 1]
3) L1.append(8): add 8 to end of list
4) L1.insert(3, 8): will add 8 at index 3
5) L1.pop(2): will delete at element index 2
6) L1.remove(21) will remove 21 from list

* Tuples
t = (1, 2, 4, 5)
print(t[0]), Can't update the value of tuple.
→ Can't change the value in tuple.

~~#~~ t1 = () # empty tuple
t1 = (1,) # with single element
t1 = (1, 2, 3) # with multiple element

Tuples method
1) a.count(1) : a.count(1) will return no of times 1 occur in a

2) a.index(1) = a.index(1) will return a first occurence of 1 in a.

* Chapter 5 - Dictionary and Sets
Dictionary is a collection of key value pairs

```
my dict = {
        "Fast" : "In Quick manner",
        "Harry" : " A Coder"
        }
    print ( my dict = [ 'Fast' ] )
```
output : In a quick manner,

Syntax :
```
    a = { "key" : "value" ,
          "marks" : "100" ,
          "list" : [1, 2, 3]
        }
```
* They are unordered, mutable, indexed
cannot write duplicate value

Dictionary Methods

1) print(list (mydict.keys()))
= print the keys of dictionary

2) print(mydict.values())
= print the keys of the dictionary

3) print(mydict.items())
= print the (key,value) for all contents of dictionary.

4) Update_dict = {
"Ankur" = "Smart"}
}
mydict.update(Update_dict)

5) print(mydict.get("harry"))
agar harry nahi hoga to return none karega.

* SETS
a = {1, 3, 4, 5}
print (a)

Set is a collection of non-repetitive element

empty set
b = set()
print(type(b))
1) b.add (4) ;

We cannot add list in set, but we can enter tuple, we cannot add dictionary

* Properties
, unordered, unindexed, no way to change items in set , can't contain duplicate value

* Oprations (methods)
S = {1, 8, 2, 3}

1) len(S) : returns 4 , the length of set
2) ~~S.remove (8)~~ : update the set s and remove 8 from set
3) S.pop () Remove arbitrary element from set and return element removed
4) ~~S. clear()~~ : Empties the set
5) S.union({8, 11}) : Returns a new set with all item from both set {1, 8, 2, 3, 11}
6) S. intersection ({8, 11}) Return a new set which containing only item in both set {8}

Chapter 6 - Condition expression

We must be able to execute instructions on a condition(s) being met. This is what conditions are for.

* If else and elif in python.
  If else and ~~if~~ elif statements are a multiway decision taken by our program due to certain condition in our code.

Syntax:

```
if (condition1):        // true hai
    print("yes")
elif (condition 2):     // if cond 2
    print("No")              is true
else :
    print("Maybe")
```

Code

```
a = 22
if (a>9) :
```

* Relational Operators
  Relational operators are used to evaluate condition inside the if statement

Some ex of relational operator are:
== , >= , <=

* Logical operator
and   or   not

* ~~[illegible]~~ Is and In

a = [45, 56, 6]
print (45 in a)
output = True

(In)

(Is)
a = none
if (a is none):
print (yes)
else:
(No)

* Chapter 7 Loops

* Two loops
while loop, for loop

```
i = 0
while i<10:
print ("Yes")
 i = i+1
```

evaluates to true, the body of loop is executed otherwise no.

```
while condition:
# body of loop
```

In while loop, the condition is checked first, if it

, 9]
in l
also

* For loop
A for loop is used to iterate through a sequence like list, tuple or string (iterables)

```
l = [1,2,8]
for item in l:
 print(item)
```

b) Range function
The range function in python is used to generate a sequence of numbers. We can also specify the start stop and step size.

```
For i in range (2,8):
 for i in range (1, 10, 2):
                       step size.
```

* For loop with else

```
for i in range(10):
   print(i)
else:
   print("condition is false")
```

— Relational Op
Relational operators
to evaluate

when loop ~~exlunt~~

```
l = [1, 7, 8, 9]
for item in l
    print (item)
else:
    print ("Done")
```

Break statement ~~stat~~

```
for i in range ~~a~~(10):
    print(i)
    if i = = 5:
        break
```

Note - Jab tak successfully for loop execute nahi hoga to else kaam nahi karega

Continue statement ~~std~~

```
for ~~c~~ i in range(10):
    ~~[illegible]~~ if i = = 5:
        continue
    print(i)
```

[5 print nahi hoga]

Pass statement ~~t~~
It instructs to do nothing.
Null statement kahte h.

Ex:

```
l = [1, 7, 9]
for item in l.
    pass:
```

f string

★ print(f" {num} X {i} = {num*i}"

★★ Ch-8 Functions

A function is a group of statements performing a specific task.

```
def percent (marks):
    p = ((marks [0] + marks [1] + marks [2]
        + marks [3]) / 400) * 100
    return p

marks 1 = [45, 78, 11, 16]
percentage 1 = percent (marks1)
```

Syntax,
```
def func1():
        print("Hello")
```

for function call, func1()

★ Types of fnc
1) Built in len(), print(), range()
2) User defined

★ Function with argument
Can accept some values it can work with.

def get (
g =
ret
a = get

★ Default pa
we can hav
in a funct
def g
(f
2
g

★ Recursion
Recursion
Used to c
formula
fact
d
fo

Date / /
Page No.

```
def greet (name):
    gr = "Hello" + name
    return gr
a = greet ("Harry")
```

* Default parameter value
we can have a value as default argument in a function

```
    def greet (name = "Stranger"):
        print ("Good day", name)

    greet ("Harry")
    greet ()
```

*

* Recursion
Recursion is a function which calls itself
Used to directly use a mathematical formula as function.

factorial(n) = n x factorial (n-1)

```
    def factorial_iter (n) :
        product = 1
    for i in range (n) :
        product = product * ( i +1)
    return product

    f = factorial_iter (5)
    print (f)
```

```
print("Hello", end="")
print("Hi", end="")
print("are", end="")
print("you", end="")
```

Sum of n = (Sum of n-1) + n

Sum 5 ... 5

1+4+3+2+...

```
this = "      Harry is a boy      "
print(this)
print(this.strip())
```

* Chapter 9, File I/O

The random access memory is volatile and all its content are lost once a program terminate. In order to persist the data forever we use file.

A File is data stored in a storage device. A python program can talk to the file by reading content from it and writing content to it.

Types
i) Text (.txt)
ii) Binary (.jpg .dat)

Opening a fil
Python has a
It takes 2

Open
Open
built in

f = open
data
print
f.close

* Other me
reading
* Mode
r -
a -
'rb'
'r1'

Writing
In or
it
we
write

Opening a file.
Python has an open() function for opening file
It takes 2 parameters : filename and mode

```
open("this .txt", "r")
↓            ↓          ↓
open is a   filename   mode of opening
built in function      (read mode)
```

```
f = open('sample.txt', 'r')
data = f.read()          # open f.read(5) Range
print(data)                to first five range.
f.close()
```

* other method
1) readline : for single line print only
* Mode
r - read   w - write
a - appending   + - for updating
'rb' will open for read in binary mode
'rt' for text mode

Writing file.
In order to write to a file, we first open it ~~to~~ in write or append mode after which we use the python f.write() method to write to the file

```
f = open("this.txt", "w")
f.write("This good")
f.close()
```

* With statement
    with open('sample.txt', 'r') as f:

Note - we dont need to open or close

* Chapter 10 Object Oriented Programming

- Creating object is an approach of program. This concept focus on using reusable code; (DRY) principle

```
class Number:
    def sum (self):
        return self.a + self.b
num = Number()
num.a = 12
num.b = 34
s = ~~s~~ num.sum()
print (s)
```

Class
A class is a blueprint for creating object
Class Employee
  # method & variable

Object
- An object is an instantiation of a class.
- Object of a given class can invoke the methods available to it without revealing the implementation details to the user.

modelling - [illegible]

Noun →
Adjective →
Verb →

- Attributes
  An attribute
  native [illegible]
  Ex :-
      class E
      compa

instance → having
      having
    Employee

* Instance att
  Instance
  Assuming th
      h
      h

Note In
  Over

modelling a problem

Noun → class → Employee
Adjective → Attribute → name, age, Salary
Verb → method → getSalary(), increment()

* Attribute
  An attribute that belongs to the class
  rather than a particular object

Ex:-

```
class Employee
    company = "Google"        // specific to each class

instance
→ harry = Employee()        // object instantiation
    harry.company
  Employee.company = "Youtube"   // Changing class
                                    attribute
```

* Instance attribute
  Instance that belong to instance (object)
  Assuming the class from previous example

```
        harry.name = "Harry"
        harry.salary = "30k"   // adding instance
                                  attribute
```

Note Instance attribute take preference
  Over class attribute

* Self parameter
self refers to the instance of the class. It is automatically passed with a function call from an object.

harry.getSalary() → self is harry
equivalent to Employee.getSalary(harry)

function getSalary is defined as

```
class Employee:
    company = "Woofi"
    def getSalary(self):
        print("Salary no")
```

* static method
Sometimes we need a method which we don't require self. we can define a static method

```
@staticmethod
def greet():
    print("Hello")
```

** __init__() Constructor
__init__ is a special method which is first run as soon as the object is created.

It takes
function a
for ex

d

def

harry

Ch-11

Inherita
new c
Syntax!
class

Cla

we
add
obj
ac
k

It takes self argument and can also take further arguments
for eg

```
        Class Employee:
    def __init__(self, name):
            self.name = name
    def getSalary (self)
```

harry = Employee("Harry") ⇒ Object can be instantiated using constructor like this.

Ch-11 Inheritance

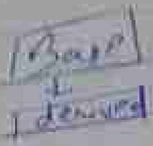
Inheritance is a way of ~~copying~~ creating a new class from an existing class

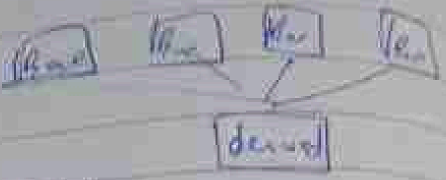
Syntax:

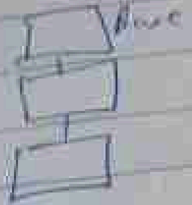
```
class Employee:          ⇒ base class
    # Code

class Programmer(Employee):  ⇒ derived class
    # code
```

We can use the method and attributes of Employee in Programmer object. Also we can overwrite or add new attributes and methods in programmer class

☆ Types

1) Single 

2) multiple 

3) multilevel 

☆ Super () method
Super method is used to access the method of a super class in the derived class
super().__init__()
↳ calls constructor of base class

super(). takebreath()

☆ Class Method
A class method is a method which is bound to the class and not the object of the class.
@classmethod decorator is used to create a class method.

```
@classmethod
def change salary (cls, sal):
    cls.salary = sal
```

* @property decorature
  consider the class

```
        class Employee
          @property
        def name (self):
            return self.ename
```

If e = Employee() is an object of class employee, we can print e.name to print the ename / call name() function

* @getter and @setter
  The method name with @property decorator is called getter method
  we can define a function + @name.setter decorator like below:

```
        @name.setter
        def name (self, value):
            self.ename = value.
```

* operator overloading in python
  operator in python can be overload using dunder method.
  these method are called when a given operator is used on a object.

Operator in python :
p1 + p2 → p1 __add__(p2)
- sub
// mul
truediv
~~/~~ floordiv
//

* other dunder (magic methods in python

1) __str__() → used to set what gets displayed upon calling str(obj)

2) __len__() → used to set what gets displayed upon calling __len__() or len(obj)

26/3/2
AZ

* Tag
(cr
Tag

2) B
Mf
A

T
c
a
C