```
#PYTHON(CHP1)
"""
    1. PRINT STATEMENT.
         Syntax:
         print("Helloworld")




    2. COMMENTS.
       i.Single line comment - we write #
                               before the text.

      ii.Multiline line comment - we wrap our
                                  text with thriple
                                  quotes.




    3. VARIABLES.
        Variables - They are container where we store Data values.
        Syntax:
               Variable_name = value
               a=5





    4. DATATYPES.
        1.STRING - "sam is a good boy"
        2.INTEGERS - 50
        3.FLOAT - 32.5
        4.BOOLEAN - True/False


    NOTES: To change Lines in Python we use '\n'.






    5. STRINGS

       i. STRINGS CONCATINATION.
        a="Rohit"
        b="Pandey"
        print(a+b)----->Rohit Pandey


       ii.STRING FUNCTIONS.

          1.lower() - converts the entire string
                      into lowercase.
```

Syntax:
```
s="Ram"
s.lower()
```

2.upper() - converts the entire string into uppercase.

Syntax:
```
s="Ram"
s.upper()
```

3.islower() - checks that your string is in lowercase or not.

Syntax:
```
s="Ram"
s.islower()
```

4.isupper() - checks that your string is in uppercase or not.

Syntax:
```
s="Ram"
s.isupper()
```

5.len(variablename) - finds the strings length.

Syntax:
```
s="Ram"
len(s)
```

6.index(value) - finds the index of the first occurence of the given value.

Syntax:
```
s="Ram"
s.index('a')
```

7.replace(old,new) - replaces old value with new value.

Syntax:
```
s="Ram"
```

```
                s.index('R','S')



       8.split() - converts a string into
                   a list of strings.

                  Syntax:
                     s="Ram is a good boy"
                     words = s.split()

                 words->["Ram","is","a","boy"]



       9.str() - converts a number to a string.

               Syntax:
                 s=5
                 str(s)

   iii. ACCESS ELEMENTS FROM STRING.
        Here String index starts from 0.
        Syntax:
               Stringname[indexvalue]
```

```
 6. NUMBERS.

       i.Functions in numbers.

           1.abs(value) - returns a absolute value
                          of the given value.

                     Syntax:
                          a=-5
                          print(abs(a))--->(5)


           2.pow(a,b)  - returns a powers of a number
                         from the two parameters.

                     Syntax:
                          a=5
                          print(pow(a,2))--->(25)
```

3.max(a,b) - returns a max number
from the given two parameters.

Syntax:
```
a=5
b=10
print(pow(a,b))--->(10)
```

4.min(a,b) - returns a min number
from the given two parameters.

Syntax:
```
a=5
b=10
print(pow(a,b))--->(5)
```

5.round(value)/ceil(value) - returns a round
of value of the
given value.

Syntax:
```
a=5.77
print(round(a))--->(5)
```

6.sqrt(value) - returns a square root
of the given value.

Syntax:
```
a=36
print(sqrt(a))--->(6)
```

7.int() - converts a string to number.

Syntax:
```
s="5"
int(s)
```

NOTES: Import math module in python
before using this functions.
Syntax:
```
from math import *
```

```
   7.TAKING INPUT.

       i.NUMBERS - int(input("Enter:"))
       ii.Strings - input("Enter:")
       iii.List - eval(input("Enter:"))
                      OR,
                   list(input("Enter:"))




   8. OPERATORS IN PYTHON

      1.Arithmetic Operator - +,-,%,/,//,**.
      2.Assignment Operator - +=,-=,*=,/=,//=,**=
      3.Comparison Operator - ==,!=,<,>,<=,>=.

      4.Logical Operator - i.AND(and) - T+T=T , T+F=F
                          ii.OR(or)- T+T=T, T+F=T, F+F=F
                         iii.NOT(not)-  T=F,F=T.

"""
#lecture done till 1:03:00
```

```
#PYTHON(CHP2)
"""

  1. LIST.
         List - It is actually a Datastructure
                which stores a list of information.
                It can stores values of all datatypes
                together.It is mutable.

        Example:
             a=[1,2,"Rohitt",true];


          i.len(listname) - returns length
                            of the given list.

                Syntax:
                    a=[1,2,3]
                    len(a)------>3


         ii.ACCESSING LIST ELEMENTS.
             listname[indexvalue]

              NOTEs - indexing starts from 0
                      from LHS.But from RHS
                      it starts from -1.




        iii.LIST SLICING.
            listname[start:stop:step]




         iv.UPDATING LIST.
            listname[indexvalue]=new value




         v.Joining list.
            a=[1,2]
            b=[3,4,5]
            print(a+b)




         vi.LIST FUNCTIONS.

             1.extend(list) - It helps in joining
                              two list by taking a
                              parameter as list
```

and then joining it
with other list.

Syntax:
```
a=[1,2,3]
b=[4,5]
a.extend(b)---->[1,2,3,4,5]
```

2.append(value) - takes a value as parameter and adds it at the end of the list.

Syntax:
```
a=[1,2,3]
b=4
a.append(b)
```

3.insert(index,value) - takes two parameters as index and v alues and add the value at that paticular index.

Syntax:
```
a=[1,2,4,5]
b=3
a.insert(1,b)
```

4.remove(listitem) - removes the listitem taken as the parameter.

Syntax -
```
a=[1,2,3,5]
a.remove(5)
```

5.clear() - It removes all elements from a list.

Syntax:
```
a=[1,2,3,4,5]
a.clear()
```

```
6.pop() - It removes the lastelement from
          end of the list.

     Syntax:
        a=[1,2,3,4,5]
        a.pop() ------>[1,2,3,4]



7.index(listvalue) - It returns the index of
                     list value given as
                     parameter.

    Syntax:
        a=[1,2,3,4,5]
        a.index(5)----> 4



8.count(value) - It takes a value as parameter
                 and counts its occurence in a
                 list.

    Syntax:
        a=[1,2,1,1,3]
        a.count(1)---->3



9.sort() - It helps in sorting the list.

    Syntax:
        a=[1,2,5,4,3]
        a.sort()---->[1,2,3,4,5]



10.reverse() - It helps in reversing a list.

    Syntax:
        a=[1,2,3,4,5]
        a.reverse()---->[5,4,3,2,1]


11.copy() - It helps in copying a list.

    Syntax:
        a=[1,2,3,4,5]
        b=a.copy()
```

## 2.TUPLES.

```
  Tuples - It is used to store multiple values
           same as list but tuples are immutable
           i.e. we cannot change its values.

    Syntax:    a=(1,2,3)
```

```
    i.len(listname) - returns length
                        of the given list.

            Syntax:
                a=[1,2,3]
                len(a)------>3
```

```
    ii.ACCESSING TUPLE ELEMENTS.
       tuplename[indexvalue]
```

```
    iii. tuple(parameters) - It takes parameters
                             sequentially and makes
                             it a tuple of elements.

            Syntax:
                    a=tuple(12345)
                    print(a)------>(1,2,3,4,5)
```

## 3.FUNCTIONS.

```
   fUNCTION - It is a block of code which only
              runs when it is called.

      Syntax:
             def sum(a,b):
                 return a+b

             #main
              x=4
              y=5
              z=sum(x,y)
              print(z)
```

```
4.CONDITIONAL STATEMENTS.

   Example:
     a=10
     if(a>0):
        if(a%2==0):
            print("Positive Even No.")
        else:
            print("Positive Odd")
     elif(a<0):
         print("Negative")
     else:
         print("Neutral")


"""
#Completed till 2:07:00
```

```
#PYTHON(chp3)
"""

   1. DICTIONARY.

      Dictionary - It is a datastructure in
                   Python that store values
                   in Key-Value pair.

            Example:
                a={1:"rohit",2:"ram",3:"rock"}




       i. ACCESS DICTIONARY VALUES.

            a. dictname[key]
            b. dictname.get(key,defaultvalue)


      ii. UPDATING DICTIONARY VALUES.
            dictname[key]= new value


     iii. DELETING FROM DICTIONARY.
            dictname.pop(key)


      iv. ADDING ELEMENTS TO DICTIONARY.
            dictname[new key]= new value




    2. WHILE LOOP.

        Syntax:
              initialize
              while(condition):
                  statement
                  reinitialize


        Example:
             n=12345
             sum=0
             while(n>0):
                 d=n%10
                 sum+=d
                 n//=10
             print(sum)
```

3. FOR LOOP.

Syntax:

```
for i in list/string:
    statement

     OR,

for i in range(start,stop,step):
    statements
```

Example:

```
a=[1,2,3,4,5]
for i in a:
    print(i)

      OR,

for i in range(1,10):
    print(i)
```

4. 2D LIST & NESTED LOOPS.

2D LIST - It is list of datavalues within a list.

Example:

```
a=[
    [1,2,3],
    [4,5,6],
    [7,8,9].
    [0]
]
```

i.ACCESS ELEMENTS FROM 2D LIST.

```
listname[row indx][column indx]

         OR,

 for row in a:
     for col in row:
          print(col)
```

```
   5. TRY/EXCEPT.

      Syntax:

            try:
                a=10/0
                num =12[]
                print(num)

            except ZeroDivisionError as err:
                print(err)
            except ValueError:
                print("Invalid Input")
            except:
                print("Error")


            In try block if some code is wrong
            then this except blocks come into
            play.

            Only Except block works for  all
            kind of errors, while others works
            for specific type of errors.

            Over here err, is the actual error
            what we are getting.

"""
#done till 3:12:00
```

```
#PYTHON(chp4)
#FILEHANDLING AND OOPs
"""
    1.OPENING AND CLOSING FILES IN PYTHON.

        Syntax:
            fh = open("filename.txt","r")

            Here,
                r = READ mode - we can only read.
                w = WRITE mode - we can only write.
                a = APPEND mode - we can only append
                                new things at end of file.

                r+ = READ & WRITE mode - we can read and write.



        Syntax:
            fh.close()






    NOTES: i. If we use write(w) mode to write,then it
              first overwrites the entire file and then
              writes the new stuff in it.

           ii.If we open a non-exising file in write(w)
              mode then it creates a new file with that
              name and writes in it.








    2. READING FUNCTIONS.

       i. readable() - It checks wheter a file
                    is readable or not i.e. it
                    is in (r , r+ or w+ mode) or
                    not.

            Example:
                    a=open("rohit.txt","r")
                    print(a.readable())----->TRUE




      ii. read() - It reads the entire file.
```

Example:
```
a=open("rohit.txt","r")
print(a.read())
```

iii. readline() - It first reads the first line of the line the cursor then moves to the next line. And if we use it again then it prints the next line.

Example:
```
a=open("rohit.txt","r")
print(a.readline())
```

iv. readlines() - It reads all the lines and then returns them in a list.

Example:
```
a=open("rohit.txt","r")
print(a.readlines())
```

```
['1.Rohit\n', '2.Sarbottam\n','3.Risav']
```

3.WRIING and APPENDING IN FILES.

i.write() - It helps to write in your file.

Example:
```
a=open("rohit.txt","a")
a.write("\n5.Sam")
```

ii.writelines([]) - It takes a list of items of to wrie in the file and then writes the file.

Example:
```
a=open("rohit.txt","a")
```

```
            a.writelines(["roh","pic"])




4.MODULES AND PIP.


    MODULES - It is Python file that we
              can import in our files to
              use their important functions.

              It is of two types--
              i.Builtin
              ii.Usermade

        Example:
            import random
            a=1
            b=5
            print(random.randint(a,b))

            It will run generate random
            numbers between a and b.




    PIP - It is a package manager which is used
          to install various Python Modules.

        Syntax:
              pip install modulename
              pip uninstall modulename







"""
```

```
#PYTHON(chp4)
#OOPs.
"""
   1.CLASSES & OBJECTS.

     CLASSES - A class is a user-defined blueprint
              from which multiple objects are created.

        For Example: Let us say, we want to make
                      a collection of dogs with
                     different breed and name.
                     If we use normal datastructures
                     then it lacks organizations
                     and here the class comes into
                     play.

                     We create a class Dog with
                     two properties(objects property)
                     breed and name.Through which
                     we can make multiple dogs(objects)
                     of different name and breed.

        Syntax:
              class Dog:
                    species = "mammal"
                    def __init__(self, breed,age):
                        self.breed = breed
                        self.age = age




     OBJECTS - An Object is an instance of a Class.
              An instance is a copy of the class
              with actual values.

        For Example: As class Dog was blueprints
                      which had two properties
                      breed and age. With the
                      help of this BLUEPRINT
                      class we can multiple REAL
                      Dogs(objects), which are not
                      blueprints.


    Syntax:
        dog1 = Dog('Doge',12)
        dog2 = Dog('Dobermann',15)
        print(dog1.breed,dog2.breed,dog1.species)

                    OR,

        from Pythonpractice import Dog
```

```
dog3 = Dog("farak",11)
dog4 = Dog("tom",18)
print(dog3.breed,dog4.breed,dog3.species)
```

2.OBJECT FUNCTIONS.

It is written within a class.

Syntax:
```
class Dog:
      species = "mammal"
      def __init__(self, breed):
          self.breed = breed
      def check(self):
          if(self.breed=="Doge"):
              return "Best dog."
          else:
              return "Not best dog."

dog1 = Dog('Doge',12)
print(dog1.check())
```

3.INHERITANCE

Inheritence - Here we have a concept of generic classes.One is the PARENT/BASE class from where CHILD/DERIVED classes  can inherit all the function and properties to use it for their own purposes.

And if they want they can also overwrite this functions while they are using it.

Example:
```
class Person:
   def name(self):
       print("Hello Rohit")
   def gender(self):
       print("I am male")
```

```
            class Boy(Person):
               def name(self):
                   print("Hello Sam")
               def prof(self):
                   print("Developer")


             boy1 = Boy()
             boy1.name()
             boy1.gender()
             boy1.prof()

"""
```