PYTHON

~ UJJWAL MITTAL

CONCEPT1

VARIABLES IN PYTHON

ITS does not require any type of command to declare a variable

```
x=5
y='name'
print(x)
print(y)
```

For Finding The Data Type

```
print(type(x))
print(type(y))
```

Assign Multiple Value In One Line

```
x,y,z='hello','how','are'
print(x)
print(y)
print(z)
```

Assign Same Value To Multiple Variable

```
a = b = c = 'string'
 print(a)
 print(b)
```

print(c)

Function Ke Inside Variable Local Hai And Bahar Global Hoga .Agar Hume Function Ke Inside Ke Variable Ko Global

Banana Hai Then We Declare It With "Global" Keyword

y = "india" # here it acts as global

def firstfun():

y ="america" # here y is local

print(y + "is world power")

firstfun() // america is world power

print(y + "is world power") # india is world power

to make local variable global in function

```
y = "india"                          # here it acts as global
def firstfun():
     global y
     y ="america"                    # here y is local
     print(y + "is world power")

firstfun()                           # america is world power
print(y + "is world power")             # america is world power
```

MINI-CONCEPT

1. **"//" gives the absolute value of division (integer division)**
2. **"**" x**3 means x*x*x**

```
x,y,z = 7,2,2
print(x/y/z)                    // 1.75
print(x//y//z)                  // 1
print(x%y%z)                    //1
print(y**3)                     // 8
```

CONCEPT 2(conditions)

LETS DO IF ELIF AND ELSE

```
x=20
y=30
if x>y:
   print("x is greater")
elif x==y:
   print("both are equal")
else:
    print("y is greater"
```

```
# for finding the largest number among three's
def find_largest(num1,num2,num3):
    if num1>num2 & num1>num3:
        print(str(num1) + "is largest")
    elif num2 > num1 & num2>num3:
        print(str(num2) + "is largest")
    if num3>num1 & num3>num2:
        print(str(num3) , "is largest")

num1=int(input("enter the first number:"))
num2=int(input("enter the second number:"))
num3=int(input("enter the third number:"))
find_largest(num1,num2,num3)
```

```
def grade_system(marks) :
    if marks > 90:
        print("well done\nyou passed with AA grade")
    elif marks>80:
        print("very good\n you passed with AB grade")
    elif marks>70:
        print ("good\n you passed with B grade")
    elif marks>60:
        print("you passed with C grade")
    elif marks>33:
        print("you passed with D grade")
    else:
        print("you are FAIL")

marks=int(input("enter your score:"))
grade_system(marks)
```

concept 3 { f string}

see we have two methods to print a statement having variable that are

1. **Using + symbol**
2. **Using , comma**

For example

```
x = 'modi'
y = 60
z = 'Gujarat'
print(x+" is the prime minister "+"he is "+ str(60)+"years old "+"he is
from "+ z)
print(x, "is the prime minister"," he is ", str(60),"years old .he is from
", z)
```

but in above example its difficult to write above syntax like commas then double quotes and if there are 10-12 variable then it become more difficult that why concept of f string come in existance

```
print(f"{x} is the prime minister of India. he is {y} years old. he is from
{z}")
```

so here put f in starting and write everything in one double quote and whereever ther is variable put in {} …

CONCEPT 4 (STRINGS and list)

String is collection one or more character put in single quote or double. ex:- S=''this is a string".

MINI-POINTS:

1. Index of string starts **from 0.**
2. If we start from last of the string then it start with **-1.**
3. **String*n** = stringstringstring……..n
4. In slici**ng string[start : stop : step]**
5. **String.lower(),string.upper(),string.isupper(),string.islower(),string.replace('kisko',' kisese')**
6. For finding a substring **string.find('substring')** . this will give the index of first substring alphabet.
7. **Substring in string** returns true or false**.**
8. **String.capitalize() this** make first letter capital of every name in string line

```
string='ujjwal_mittal'
ans = len(string)
print("after finding len")
print("hello")
print(ans)
print(string*3)      // Ujjwal_mittal Ujjwal_mittal

print(string[5])
print(string[-2])
print(string[-1])
print(string[7:12])
print(string[2:6])
print("hi i am ujjwal mittal.\ni am persuing by bachelor of degree in
mathmatics and computing")
print(string.upper())
```

```
print(string.lower())
print(string.isupper())
print(string.islower())


string1 = string.replace('ujjwal','hi')
string2 = string.replace('mittal','world!')
print(string1)      // hi_mittal
print(string2)       //Ujjwal_world
```

9. **string is immutable** . Ex:

s = "hello"

s[0] = 'y' ◊ gives an error

s = 'y'+s[1:len(s)] ◊ is allowed, s bound to new object

LISTS

Both same and mixed type of data are allowed to store in lists.

List is mutable means data can be updated or changed.

```
first_list=['ujjwal',19072004,18,'mittal']
print(first_list)
print(len(first_list))                                    //4
print(first_list.index(18))
first_list.extend(['baddi',12511])          // now here become 6 elements
print(first_list)
print(first_list[2:4])              // this is slicing
```

for list having interger element there many function like (sort , max , min, append, insert,extend,remove,pop,

1. **, append:- to add element in the last**
2. **Insert :- to add element at a specific index number ex:- name.insert(2,5) {mens insert 5 at index 2}**
3. **Extend:- to add more than one element together in the last**
4. **Remove;- remove element from the last**
5. **Pop:- remove elements from specific index**

CONCEPT 5 (MODULE)

RANDOM MODULE:- already written code for a specific task to perform by other person we just need to import it.

```
a="ujjwal"
print(a+ " is a collage student")
print(a.lower())
print(a.upper())
```

```
print(a.islower())                                  #true
print(a.isupper())                                  #false
print("\n")
print(a.upper().isupper())                        #true
print(len(a))                                     #6
print(a.index("wa"))                               #3

#hey this is we starting functions
ans=-3
print(abs(ans))                                 #3
print(pow(3,3))                                 #27
print(max(25,5))                                #25
print(min(5,9))                                 #5
print(round(10.555))                            #11
print(round(5.223))                             #5

from math import*
print(sqrt(81))
print(ceil(4.7))
print(floor(4.7))
```

RANDOM MODULE :- by importing this module we have many function related to randomness (helpful in making funny program like head and tail , luck game etc..)

- **For a file name as my-module ➔> import my_module**
  In this case we need to call our function with module name i.e my_module.fun()
  If module name is lengthy import it like ➔> import my_module as m
  Function call as print(m.fun())

```
import math
print(math.ceil(3.7))
print(math.floor(4.7))

import math as m
print(m.ceil(5.6))
```

- **If we donot want to use module name while calling we need to import i.e**
  1. **from math import***
  2. **From math import ceil , floor**

```
from math import ceil,floor
print(ceil(3.7))
print(floor(4.7))                    give error in sqrt as we haven't imported it
print(sqrt(4))
```

```
from math import*
print(ceil(3.7))
print(floor(4.7))
print(sqrt(5.2))
```

CONCEPT 6 {INDEX ERROR}

it is an error when we try to excess the index than our list index

CONCEPT 7 { TUPLES() }

1. **Tuple is ordered sequence of items same as list.**
2. **Indexing in start from 0 and indexing from end start with -1.**
3. **Tuples are Fast than lists..**
4. **Tuple are inmutable. It means element cannot be added or remove once the tuple is created..**
5. Repition of **tuple tup1*2**

Use () brackets for tuple and [] for lists

```
tup1=('ujjwal',19,5.11,'b.tech')
print(tup1)
print(tup1.index(19))
print(tup1.index(5.11))
print(len(tup1))
# using the slicing method

tup2=tup1[0:2]
print(tup2)

tup3=tup1[0:3]
print(tup3)

print(type(tup1[2]))                    // float
print(type(tup1[3]))
print(tup1.index(19))                   // str

rep_tup = (1,4,5)
print(rep_tup*2)        // 1,4,5,1,4,5

max(rep_tup)
min(rep_tup)
len(rep_tup)

new_tup = sorted(rep_tup)
```

CONCEPT 7 { SETS }

SET is collection of set and created using {}..

1. **The elements in set are not in order.**
2. **That's why there is no indexing in set.**

```
3. set1={10,5,9,'python','c','java'}
   set2=set()
   print(set1)
```

OUTPUT :- {'c', 5, 9, 10, 'java', 'python'}

Therefor in sets output is not in sequence and for creating an empty set set2=set() this is the only method.

NOW LETS START OPERATION OF SETS

1. union and intersection of set is same as in mathematics as in union no repetition of elements and in intersection common element got in output..

```
2. set1={1,2,3,4,5}
   set2={4,5,6,7,8}
   print(set1.union(set2))
   print(set2.union(set1))

   set1.update(['ujjwal','pinku'])
   print(set1)                                  #{1, 2, 3, 4, 5,
   'ujjwal', 'pinku'}

   print(set1.intersection(set2))               # {4, 5}

   print(set1.difference(set2))                 #  {1, 2, 3, 'pinku',
   'ujjwal'}

   print(set1.difference(('pinku')))
```

refer google for disjoint, subset, deleting the set

CONCEPT 8 {DICTIONARIES }

Dictionary is ordered , mutable and does not allow duplicates.

Represented in key:value pair and referred by key name.

Dictionaries cannot have two items with same key.

Whole dictionary is enclosed in curely brackets .

If key not found give an error

```
my_dict = {}          # empty dictionary

sample_dict = {'key_1': 3.14, 'key_2': 1.618, 'key_3': True, 'key_4':
[3.14, 1.618], 'key_5': (3.14, 1.618)}

sample_dict ['key_6'] = 'new_element'
sample_dict ['key_4'] = 'updated_element'
del(sample_dict['key_1'])

print(sample_dict)


        //  {'key_2': 1.618, 'key_3': True, 'key_4': 'updated_element',
'key_5': (3.14, 1.618), 'key_6': 'new_element'}
```

Indentation:- these are the spaces in the beginning of any for loop or function in python .indentation indicates a block of code in python ,hence it is important in python .As in other languages this block is represented by {} which is not in python..

MUTATION , ALIASING AND CLONING :

1.ANALOGY : attributes of a person . means same person known by many name like son father , husband,grandson and many more .ALL this name point to a same person.

When we equate x and y it actually act as a pointer and point to same list.

Y is the Alias of x changing one changes the other .Hence , sometime it act as drawback in program.

```
x = ["pink" , "red" , "black",]
y = x
y.append("yellow")
x.append("brown")

print(x)              //['pink', 'red', 'black', 'yellow', 'brown']
print(y)              // ['pink', 'red', 'black', 'yellow', 'brown']
```

2.CLONING A LIST:

SO, WE MAKE COPY OF LIST TO AVOID ALIAS IN PROGRAMM

```
cool = ["pink" , "red" , "black",]
chill = cool[:]
cool.append("yellow")
print(cool)            //['pink', 'red', 'black', 'yellow']
print(chill)           // ['pink', 'red', 'black']
```

3. SORTING IN LIST:

1. Sort() : no new list form same list is updated.
2. Sorted(): old list remains the same and one new list is formed.

```
3. roll = [10,5,9,1,7]
   new_roll = roll.sort()
   print(roll)                    #[1, 5, 7, 9, 10]
   print(new_roll)                #NONE


   roll1 = [10,5,9,1,7]
   new_roll2 = sorted(roll1)
```

```
print(roll1)                    #[10, 5, 9, 1, 7]
print(new_roll2)                #[1, 5, 7, 9, 10]
```

CONCEPT (LOOPS)

1. while loop

```
i=1
while i<6:
 print(i)
 i+=1
else:
 print("i is no longer less than 6")          // output 1 2 3 4 5  str

i=0
while i<=8:
 i += 1
 if i==4:
  continue
 elif i==7:
  break                                        // 1 2 3 5 6
 else:
  print("this is", i, "loop")
```

#2. FOR LOOP

Syntax : for I in range(50):

Print(2*i)

1. **to not go in next line use end=””**
2. **last number of range is excluded.**

```
row = int(input("enter row"))
for i in range(row):
    print()
    for j in range(i+1):
        print('+',end="")
```

```
nlis = [1,2,4,5,6,7,8,9,10,11,12,13,14]
for i in nlis:
    if i ==8:
        continue
    elif i ==11:
        break
    else:
        print(i)
```

CONCEPT 9 :- FUNCTION

1.PASS

```
def test():
    pass
```

when currently you have nothing to define use pass so that it does show error.

2.PLACEHOLDER

```
def test2(x):
   return x;

print(test2('python'))

def test2(a,b,c):
   return a, b ,c
m , n , b = test2("cpp", "java" , "js")
print(m)
print(n)
```

above is an example of placeholder as above function is returning 3 values which are placed in placeholder..

3.CALLING FUNCTION WITH PRINT AND WITHOUT PRINT

```
def is_even(x):
    print("with return ")
    return x%2 == 0

is_even(3)                          //WITH RETURN
print(is_even(3))             /* WITH RETURN
                                  FALSE  */
def is_even(x):
    print("without return ")

is_even(3)                        // WITHOUT RETURN
print(is_even(3))                /* WITHOUT RETURN
                                    NONE   */
```

```
def fun1():
    print("hello ji")
    return "return diya"
def fun2():
    print("python")

fun1()                          // hello ji
fun2()                          // python
```

```
print (fun1())                    /* hello ji
                                     return diya*/

print (fun2())                     /* python
                                     NONE*/
```

First if function call is outside print then no return statement get in output only get that you are printing in the function.

Second , if function call in print and function does not return anything then it will return NONE keyword.

4. LOCAL AND GLOBAL VARIABLE

1. The variable declared inside a function is called local variable and the the scope is only inside the function.
2. The variable declared outside the function is global variable and the scope is throughout the programm. global variable can be used at both inside the function and outside the function.

5.When local and global variable name is same:

```
process = 'Continuous fermentation'
def fermentation(process_name):
  process = 'Batch fermentation'
  if process_name == process:
    return '0.5 g/L/h.'
  else:
     return '0.25 g/L/h.'

print('The productiovity in continuous fermentation is',
fermentation('Continuous fermentation'))

print('The productiovity in batch fermentation is', fermentation('Batch
fermentation'))
print(f'My favourite process is {process}.')

/*
The productiovity in continuous fermentation is 0.25 g/L/h.
The productiovity in batch fermentation is 0.5 g/L/h.
 My favourite process is Continuous fermentation.
                                              */
```

So inside function closest defined variable is accessed first

CONCEPT 11 (EXCEPTION HANDELING)

Exception refers to an error which our python programm face which disrupts the normal flow of execution. This can be due to an unexpected case which in not in the scope of our programe.

So, put block of code in try: and except: if exception happened

Main exceptions are ZeroDivisionError, NameError , ValueError.

Try / except

```
a = int(input("first number"))
b = int(input("second number"))
```

```
try:
  print(a/b)
except ZeroDivisionError:
 print("gives a division error")
```

Multiple exception handling in one programe.

```
try:
 a = int(input("first number"))
 b = int(input("second number"))
 print(a/b)
except ZeroDivisionError:
   print("gives a division error")
except ValueError:
  print("you should provide a number")
except :
  print("something went wrong")
```

try / except / else :

here else execute strongly only if no exception raised in the programe.

```
try:
 a = int(input("first number"))
 b = int(input("second number"))
 ans = a/b
except ZeroDivisionError:
   print("gives a division error")
except ValueError:
  print("you should provide a number")
except :
  print("something went wrong")

else:
 print(f"the division as there is no error is {ans} ")

// the division as there is no error is 5.0
```

Try / except / else / finally

Finally executed always either exception occurred or not .

```
try:
 a = int(input("first number "))
 b = int(input("second number "))
 ans = a/b
except ZeroDivisionError:
   print("gives a division error")
except ValueError:
  print("you should provide a number")
except :
  print("something went wrong")

else:
 print(f"the division as there is no error is{ans} ")

finally:
 print("programe at final stage")
```

first number 6 first number 56

second number 0 second number 2

gives a division error the division as there is no error is28.0

programe at final stage programe at final stage

CONCEPT 12 (TESTING AND DEBUGGING)

CONCEPT 13 (FILE HANDELING)

FOR FILE handeling r is for reading

w-writing

a- append (adding additional content)

```
= open('myfile.txt', 'r')
print(f)
# text = f.read()
# print(text)
# f.close()
```

here i just said to made a variable that open that file

now i made a variable that stores the thing inside the file by using variable name

name of opened file . see variable 'f' store the opened file b y using which we

read the content of file making variable text.

If no parameter from ‘r’, w’ and ‘a’ is passed by default r is passed in open function..

ARRAYS

ARRAY IS CONTAINER THAT HOLD FIX NUMBER OF ITEMS WHICH SHOULD BE OF SAME TYPE.

CREATING AN ARRAY :

- Import array or from array import(*)

- **(*) means it cover all features of an array.**

LAMBDA FUNCTIONS IN PYTHON

SMALL ANONYMOUS FUNCTION AND CAN TAKE ANY NUMBER OF ARGUMENTS BUT HAVE ONLY ONE EXPRESSION.

Lambda function are block of code that can be assigned to variable, passed as an argument, or returned from a function call in languages

CONCEPT 10 (OOPS)

Python is an object oriented programming which mainly focussed on objects.

OBJECT:object is simply a collectively set of data(variable) and methods(function) that act on those data.

Class definition begin with class keyword.

1. **CONSTRUCTOR:- It is a special method to create and initialize an object in the class.constructor is invoked automatically when an object is created in class. There are different type of constructor.**

We can say it is an function that called automatically. Main purpose is to initialize the values to different objects of that class.

```
class data:
    def __init__(self,arg1,arg2):
        data.first_var = arg1
        data.second_var = arg2

    def multi(self):
        ans = data.first_var * data.second_var
        return ans

item = data(10,5)
print(data.first_var)
print(data.second_var)
ans = data.multi                // here () complete differs the result
print(ans)                      // be smart
print(item.multi())




//
10
5
<function data.multi at 0x000001B6BF1987C0>
```

SELF PARAMETER

SELF PARAMETER IS REFERENCE TO THE CURRENT INSTANCES OF THE CLASS , AND IS USED TYO ACCESS VARIABLES THAT BELONGS TO THE CLASS.

IT DOES NOT NECESSARY TO ONLY GIVE NAME SELF YOU CAN GIVE OF YOUR ON CHOICE BUT IT CONSIDERED TO BE A GOOD PRACTISE.

CHILD CLASS IN DATA CLASS

THE WORD DATA MEANS THAT ANALYSIS is a python object that inherit all its attributes of data.

Analysis is a subclass of object and data is the superclass of analysis.

Class analysis (data):

ALGORITHMS

1. Bisection algorithm

For finding the square roots or cube root of numbers

```
# cube = 27
cube = 8120601
low = 0
high = cube
error = 0.01
guess = (low+high)/2

while(abs(guess**3 - cube)) >= error:
    if guess**3 > cube:
        high = guess
    else:
        low = guess
    guess = (low+high)/2
print(f"The approx cube root of the cube is {guess}")
```

Complexity is O(logn)

2. Iteration multiplication and recursive multiplication

In iteration there is multiplication of various steps using while and for loop but in recursive multiplication you reduce the process to smaller process.
There are two things in recursion.recursive case and recursive relation

```
# iteration for factorial
def iteration_fact(x):
    ans =1
    for i in range(x):
        ans*= i+1
    return ans

print(iteration_fact(6))
```

```
#recursive case of a factorial

def factorial(x):
    if x==1:
        return 1
    return  x*factorial(x-1)

print(factorial(3))
```

1. each recursive call creates its own scope/environment.means having same variable but they have different scope in different calls.

3.MULTIPLICATION INDUCTION

Simply to prove a statement find value for simplest case i.e 0 or 1 . then prove it true for a value K and K+1 .

```
#recursive function for multiplication
def multi(a,b):
    if b == 1:
        return a
    return a+ multi(a,b-1)


print(multi(5,5))
```

```
# recursive function for power
def power(a,b):
    if b == 0:
        return 1
    return a*power(a,b-1)

print(power(3,5))
```

```
# Famous Tower Of Hanoi question
def tower_of_hanoi(n):
    if n ==1:
        return 1
    return 2*tower_of_hanoi(n-1)+1

print(tower_of_hanoi(64))
```

4. RABBIT PAIR AND FIBBONACCI SERIES

when there are multiple recursive cases. No. of female in each month in pair of Rabbit is similar to fibbonacci series in which

```
def fibbo(x):
    if x == 0 :
        return 0
```

```
    if x == 1 :
        return 1
    ans =fibbo(x-1) + fibbo(x-2)
    return ans
print(fibbo(3))
```

5.PALIDROME

```
def isPalindrome(s):
 def toChars(s):
       s = s.lower()
       ans = ''
       for c in s:
        if c in 'abcdefghijklmnopqrstuvwxyz':
            ans = ans + c
            return ans
 def isPal(s):
    if len(s) <= 1:
        return True
    else:
        return s[0] == s[-1] and isPal(s[1:-1])
 return isPal(toChars(s))
```

EFFICIENCY OF PROGRAM:

Here we talk about the the best case , avg case and the worst case.

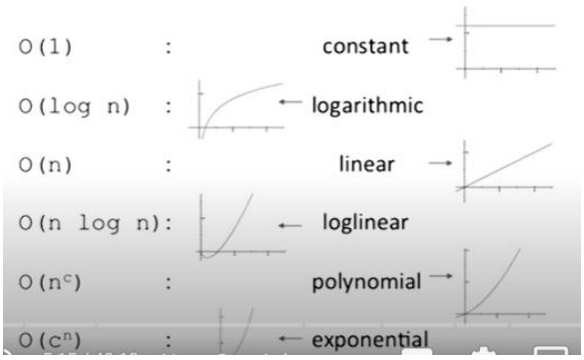
For example in searching a key in a list can have :

Best case when ket is at starting

Avg case when element in middle

Worst case when we transverse the whole list and case is not present.

O(1) < O(logn) < O(n) < O(n logn) < O(n2) < O(n3)< O(n^c) < O(c^n)



Complexities independent of input