Python day-1

Date
Page

→ To Install any Module in Python,
you have to go on your cmd
↓
& just write pip install module-name
↓
Now the Installing Process will Start

→ After you can Import them in your Python's cmd

→ So there is also Built-in Modules Exist which you do not have to Install it.

⊙ Python Package Manager

→ So the pip is your Package Manager.

→ they will Provied Either Built-in or any other Modules for your Efficient.

Modules
- Internal Modules which you already have.
- External Modules which you have not, you have to Install it.

your First Program:

print ("Hello World")

↓ ↓

function String

→ , end = " " it is a new line character.

Python day-2

Date
Page

① Print:

- print ("Hello")

→ It is the you can write No, Strings, Sentence etc.. In Op Screen

- print (5) # Integer. / Number.

- print ("your String") or print ('your String') or print ("""Multiple line String""")

→ String is Collection of Char (A, b, !, d) Inside " " ( ) , or ' '.

→ you can use ### Single quotes ~~or~~ Inside double quots or

double quots Inside Single quots

"I 'm' ching"

'I "m" ching'.

→ ("I m" Chirag")

↳ this will Cause Error.

↳ the Solution is,

- Escape Sequence :- ("I M:" chirag ")

↓

Escape Sequence

→

\'	Single quote.
\"	double quote.
\n	New line
\t	for tab Space
\b	for Back space
\\	double Slash.

→ New line Example:

Print ("Line A \n Line B")

→ Tab Space

print ("Line A \t Line B")

→ Back Spar.

print ("Line 123 \b")

→ double Slash

→ Print ("this is double back Slash \\")

Comments :-

Example: # this file for med.

This program V is 3.10.0 [f(x)] + [1]

Excessive

→ This is double \\ back slash.

→ This is Mountain /\/\/\/\.

→ This is awsome \t"\"\" \\"

Raw String :-

→ Any Print function to close the Escape Sequence Raw String is used.

→ How to write Raw String?

Print (r" Line A \n Line B ")

Line A \n Line B

Scanned with CamScanner

Python day-3

Date
Page

- **Variable** :- you can store your data at single name different datatype like String, float, Integer, double

num

→ Dynamic Programming language

- **Variable declaration rule:**

→ Number not allow in variable name Means Every Variable name should start with latter. or _ underscore

```
1num=2 ✗
1=2 ✗
```

→ Special character is not allowed. like @, #, $, &, ', ;.

- **String Concatination:**

→ More the one String will Merge with Each other is known as---

Q.
```
f_name = "med"
l_name = "patel"
full_name = f_name + l_name
print (full_name)
```

medpatel.

Eg-2

```
f_name = "meet"
l_name = "patel"
space = " "
full_name = f_name + space + l_name
print (full_name)
```

meet patel

Eg-3

```
f_name = "chirag"
l_name = "Joshi"
print (f_name + " " + l_name)
```

meet patel

Each

```
print (f_name + 3)
```

Error: str obj first and int not concate

Eg->

```
print (f_name * 3)
```

meet meet meet

① User Input:

Eg-1

```
num = input ("Enter your num:")
num1 = input ("Enter your num1:")
print = (num 1 + num)
```

Enter your num: 5
Enter your num1 = 10
510

Eg-2

```
num = input ("Enter your num:")
num1 = input ("Enter your num1:")
inum = int (num)
inum1 = int (num)
print (inum + inum1)
```

Enter your num: 5
Enter your num: 10
15
Same for Ex - ③

Eg-3
```
num = int(input("enter your num : "))
num1 = int(input("enter your num1:"))
print(num + num1)
```

Eg-4 type() you use this function for check your variable datatype

```
print(type(name)) # class 'int'
```

@ Variable - Scope:

```
def: x = y = z = 1
print(x) # 1
print(y) # 1
print(z) # 1
```

-6	-5	-4	-3	-2	-1
p	y	t	h	o	n
0	1	2	3	4	5

@ String Indexing

```
name = "python"
```

p	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

```
print(num[0]) # p
print(num[1]) # y
print(num[2]) # t
print(num[3]) # h
print(num[4]) # o
print(num[5]) # n
```

```
print(num[-1]) = n
print(num[-2]) = o
print(num[-3]) = h
print(num[-4]) = t
print(num[-5]) = y
print(num[-6]) = p
```

Python day-4

next
0 1 2 3 4

String Slicing / Selecting:

name = "python"

-6	-5	-4	-3	-2	-1
p	y	t	h	o	n
0	1	2	3	4	5

Basic Syntax of String Slicing:

[start value : stop value]

[start value : stop value : step value]

print (name [0:4])
pyth (start, stop)

print (name [4:3])
yt (start, stop)

print (name [0:6:2] # pto

print (name [0:6:3] # ph

Ex-2

p	y	t	h	o	n		i	s		a		m	a	s	s	i	v	E		
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	

l	a	n	g	u	a	g	e	.	
20	21	22	23	24	25	26	27	28	

→ print(name[0:29]) # Original String
print(name[0:30:2]) # pto smsielnus.
print(name[0:30:4]) # posmilel.
print(name[0:29:]) # also return original string

Ex-3 Sth = string Slicing

-14	-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
s	t	n	i	n	g		s	l	i	c	i	n	g
0	1	2	3	4	5	6	7	8	9	10	11	12	13

print(Sth[3:6]) # ing
print(Str[4:10]) # ng sli
print(Str[:]) # string slicing
print(Str[7:]) # Slicing
print(Str[-7:]) # Slicing
print(Str[-9:-3]) # g slic
print(Str[2:10:2]) # r n
print(Str[-1:-1:-1]) # nothing will print
print(Str[-11:-4]) # ing sli

Python day :- 6

Date
Page

★ String Method

① len() :- It defines the length of the entire String.

② lower() :- It Convert the whole String in the lower Case Character.

③ upper() :- It Converts the whole String in the upper Case Character.

④ Title() :- It Convert the all String in such way that

Ex. Word

↑ first character in upper case. → Rest of the Character in lower Case.

⑤ Count() : they will Count the no of character which is present in your string.

: At the other hand, Count() is Case Sencitive.

⑥ Strip() :- they will help to Skip unnecessary Spaie in your string

:- Syntax :-

⑦ lstrip() :- they will help to skip unnecessary spaces from the left side of your string.

⑧ rstrip() :- they will help to skip unnecessary space from the right side of your string.

⑨ Replace() :- they will help to change the one character or collection of characters to your string inplace of old one.

Syntax : print(name.replace("chirag", "meet"))
↑ old ↑ new

⑩ find() :- Search the string for a specified value and Return the position of where it is found.

⑪ Center() :- Return a Centered String

Syntax :- print(name.center(20))
↑
it is necessary to give proper length How Much you want to Centralize

* Conditional Statement *

(1) If Statement :-

→ It is a One type of Conditional terminology that Checkes that your Given Condition will Full-fill the Requirment or not.

⊙ Syntax :-

```
if condition:
```

↑ keyword

```
    # your
      code
```

If is true then Execuate this part ← otherwise not

⊙ Example :-

```
age : int ( input (" Enter your age:") )
if age > 14 :
- print (" you are eligable ")
```

② Else Statement:

→ your if is False then this part will Execute

Example:

```
if age >= 14 :
        print ("your age is 14 or above")
else:
    print ("your age is bellow 14")
```

③ Nested if-else

Example:-

```
win = 27
ei = int( input ("Enter your
if win == ei :
    print ("you win !!!")
else :
    if win > user input :
        print ("your number is High")
    else :
        print ("your number is low")
```

Python day-8

Date
Page

* Conditional Statement *:

⑥ if-elif-if:

→ the elif keyword is Python way of saying "if the previous condition were not true, then try this condition".

- Example:-

```
# show ticket price
# 0 to 3 free
# 3 to 10 150
# 10 to 60 250
# 60 above 200

age = int(input("Enter your age:"))
if 0 == age or age < 0:
    print("your ticket price: Free")
elif 0 < age <= 3:
    print("your ticket price: Free")
elif 3 < age <= 10:
    print("your ticket price: 150 $")
```

```
elif 10 < age <= 60 :
    print (" your ticket price : 250 $ ")

else :
    print (" your ticket price : 200 $ ").
```

⑥ And key word :-

→ It is a logical operator & it is used to combine two Conditional Statement.

- Example :-

```
name = "meet"
age = 14
if name == "m" and age > 10 :
    print (" you can watch the movie .. ")
else :
    print (" you can not watch the movie .. ")
```

you can use "or"

⑦ Or key word :

→ it is a logical operator and it is used to combine two Conditional Statements

→ It work like from the Both Condition if any one Condition true at atime then your Code will Execute.

⑤ in keyword:

→ the in key word is used to check if a value is present in a sequence [list, range, string].

- Example:

```
name = "meet"
if 'e' in name:
    print ("yes, m is part of name")
else:
    print ("no, m is not part of name")
```

⑥ check empty or not:

⊙ Example:

```
name = "chi"
if name:
    print ("yes")
else
    print ("No")
```

Python day-9

Date
Page

* loops *

① Python has 3 type of loop

② While loop:

→ with the while loop you can execute a set of statement as long as condition true

- Example:

```
i=1
while i<6:
    print(i)
    i=i+1
```

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

0	2=5	i=i+1
1	2=5	1=0+1
2	2<=5	2=1+1
3	<=5	3+2-1
4	<=5	4=3+1
5	<=5	5=4+1
6	<=5 X	6=5+1

- Example:

```
i=0
while i<=5:
    print("hello world")
    i=i+1
```

```
#  hello world
   hello world
   hello world
   hello world
   hello world
   hello world
```

- Example:

```
total =0
i=1
while i<=5:
    total =total+i
```

```
i = i+1        #  15
print (total)
```

total	i	i<=5	total = total+i	i = i+1
0	1	1<=5 ✓	1 = 0+1	1=0+1
1	2	2<=5 ✓	3 = 1+2	2=1+1
3	3	3<=5 ✓	6 = 3+3	3=2+1
6	4	4<=5 ✓	10 = 6+4	4=3+1
10	5	5<=5 ✓	15 = 10+5	5=4+1
15	6	6<=5 X		

● Example :

```
i = 1                  ai = int (input (" enter the num:"))
total = 0              i = 1
                       total = 0
                       while i <= ai_input :
                             total = total + i
                             i = i + 1
                       print (total)
```

#

```
Enter the num : 05
15
```

② For loop:-

→ A For loop is used for iterating over a sequence.

→ With for loop we can execute a set of statements, once for each item a list, tuple, set etc...

* Functions *

→ to Avoide the Iterative terminology we will use the "functions".

© DRY → Don't Repeat your Self:

© Dry Means:

```
a = 1
b = 2
c = a+b
Print (c) # 3
d = 3
e = 1
f = d+e
Print (f) # 4
```

} So it is Iterative Process
it is against function.

---

* How we Resolve this:-

→ to Resolve we use function like

- def add (a,b): // function define
  return a+b
  Print (add (a,b))

here def = keyword → use for Create the function
return = keyword → It will Return the Value of any kind of operation.
↳ it is a Calling Process of function

Ex-1

```
a = 1
b = 2
def add (a, b):
    return a+b
Print (add (a,b))
```

3

Creating function:

→ In python a function is define using def keyword

```
def myfunc():
    Print("Hello from a function")
```

#

Calling a function:

→ To call a function, use the function name follow by Parenthesis:

```
def myfunc():
    Print("Hello from function")
myfunc()
```

Hello from function

Use A function: & Use function inside A function:

→ Now we will use the function to check the took No is even or odd.

```
a = int(input("Enter the a no:"))
b = int(input("Enter the b no.:"))
def greater(a, b):
    if a > b:
        return a
    else:
        return b
Print(greater(a, b))
```

```
a = int(input("Enter the a no.:"))
b = int(input("Enter the b no:"))
c = int(input("Enter the
```

```
def greatest(a, b, c):
    """ if a>b and b>c:
            return a
        elif b>a and b>c:
            return b
        else:
            return c """
    temp = greater(a, b)
    return greater(temp, c)
print(greatest(a, b, c))
```

---

* default Parameter:

→ the following example shows how to use full a default Parameter.

→ If we call the function without agument, it will use the default value.

```
def user_info(name = 'unknown', age = "none"):
    return f"your name is {name} and your age is {age}."
print(user_info('meet', 22))
```

argument

o/p:- your name is meet & your age is 22

But if we will not put any argument -

o/p:- your name is unknown & your age is none.

---

* LIST *

→ List is a data structure
→ List is a order Collection of data.
→ List denoted by [].

Example list = [1, 2, 3, 4]
Print (type [list])

mix = [1, "two", "three", 4, 8, 5]

#assign
mix[1] = 2
mix[2:] = [3, 5] # [1, "two", 3, 5]
Print (mix[-1])

⊙ assigning

→ mix = [1, "two", "three", 4.0, 5]
mix[1] = 2 # [1, 2, "three", 4.0, 5]
mix[2:] = [3, 5] # [1, 2, "three", 3, 5]
Print (mix[-1]) = 5

⊙ add the data in list:

① append → mix.append ("meet") # [1, "two", "three", 4.0, 5, "meet"]
mix = ["meet", "NISARG", "Pathik"]

② insert → mix.insert (1, "2") # [1, "2", "three", 4.0, 5]

③ Extend
stud 1 = ['Meet']
stud 2 = ['Vinod', 'Pathik']
stud 1.extend (stud 2)
Print (stud 2) # ['meet', 'Vinod', 'Pathik']

Scanned with CamScanner

Remove data from list:

① pop() → it will Remove Element like last in first out.

```
Stud 1 = ['meel', 'Vinod', 'Pathik', 'misang']
Stud 1.pop()
Print (Stud 1)   # ['meel', 'Vinod', 'Pathik']
```

② Remove(): → it will Remove Specified item.

```
a = ["apple", "banana", "cherry"]
a.Remove ("banana")
Print (a)
```
["apple", "cherry"]

→ in the pop(), if you don't Specify item, it will Remove the last item.

```
a = ["apple", "banana", "cherry"]
a.pop()
Print (a)
```
["apple", "banana"]

③ del() it will used for Remove Specified index.

```
a = ["apple", "banana", "cherry"]
del a[0]
Print [a)
```
["banana", "cherry"]

④ del() :- it will used for also delet the list Completely

```
a = ["apple", "banana", "cherry"]
del a
```
#

⑤ clear. it will Empties the list. list still Remains, but has no Content

```
a = ["apple", "banana", "cherry"]
a.clear()
Print(a)
```
= []

⊙ Loop List :

① for loop :

```
a = ["apple", "banana", cherry"]
for x in a :
    Print(x)
```

```
# apple
  banana
  cherry
```

```
a = ["apple", "banana", "cherry"]
for i in range (len (a)):
    Print (a[i])
```

```
# apple
  banana
  cherry.
```

(ii) while loop :

```
a = ["apple", "banana", "cherry"]
i = 0
while i < len (this.a) :
    Print (a[i])
    i = i+1
```

```
# apple
  banana
  cherry
```

List Comprehension :

Shortest syntax for loop.

```
a = ["apple", "banana", "cherry"]
[Print(x) for x in a]
```

```
# apple
  banana
  cherry
```

Syntax =

```
List = [Expression for item in iterable
        if condition == True]
```

⊙ ~~Another~~ Copy list :

① Copy() : will make a copy of list

```
a = ["apple", "banana", "cherry"]
b = a.copy()
Print(b)
```

```
# ['apple', 'banana'
  , 'cherry']
```

② List() will make a copy.

```
a = ["apple", "banana", "cherry"]
b = list(a)
Print(b)
```

```
# ['apple',
    'banana',
    'cherry']
```

Join Two List

```
a = ["a", "b", "c"]
b = [1, 2, 3]
c = a + b
Print (c)
```

```
# ['a', 'b', 'c', 1, 2, 3]
```

```
a = ["a", "b", "c"]
b = [1, 2]
for x in b:
    a.append(x)
Print(a)
```

```
# ['a', 'b', 'c', 1, 2]
```

```
a = ["a", "b", "c"]
b = [1, 2, 3]
a.extend(b)
Print(a)
```

```
# ['a', 'b', 'c', 1, 2, 3]
```

* Tuple *

→ It is a Immutable.. you can Create once and datas can't be changes
→ It is denoted by ().
→ tuple is Immutable as same as Strings.
→ i.e.:- stud = ("meet", "Vimod", "pathik")

```
Print (stud)
#
("meet", "Vimod", "pathik")
```

→ No update, No delete, No insert, No pop, No remove

→ TUPLE ARE FASTER THEN LIST:-

⊙ use a Method in tuple:

① Count(), ② Index(), ③ len(), ④ Slicing,

⊙ tuple in looping:

```
day = ('mon', 'Tue', 'wed', 'Thu', 'Fri', 'SET', 'SUN')
for i in days:
    Print (i)
#
('mon', 'Tue', 'wed', 'Thu', 'Fri', 'SET', 'SUN')
```

⊙ Concate tuple to One Value:

```
days = ('mon')
Print (type (days))
#
<class 'str'>
```

↳ that Not return tuple

for that you have to add ','

```
days = ('mon',)
Print (type (days))
#
<class 'tuple'>
```

① tuple without Parathesis :

Example Stud_Name = 'chirag', 'meet', 'vinod', 'pathik'
Print (type(Stud_Name))
#
<class 'tuple'>

② tuple unpacking :

Example stud = ('meet', 'Vinod', 'pathik')
S1, S2, S3 = (stud)

Print (S1) # meet
Print (S2) Vinod
Print (S3) Pathik

③ List Inside tuple :

Ex batch = ('chirag', ['meet', 'Nisarg', 'pathik', 'vinod'])
0 1
tuple element
List Element

④ Print (batch) # ('chirag', ['meet', 'Nisarg', 'pathik', 'Vinod'])
batch[1].remove('Nisarg')
batch[1].append('Vaij')
Print (batch[1]) # ['meet', 'pathik', 'Vinod', 'Vaij']

Eg-2 num = [1, 2, 3, 4, 5, 6, 7, 8, 9]

Print(min(num)) # 1

Print(max(num)) # 9

Print(sum(num)) # 43

* Remove item from tuple:

→ You can not directly remove the item from tuple, you have to convert it in to list, After removing from the list you have to convert back to list in to tuple..

```
a = ("apple", "banana", "cherry")
y = list(a)
y.remove("apple")
a = tuple(y)
print(a)
# ('banana', 'cherry')
```

⑥ function return the value in tuple:

```
def add_mul(a, b):
    add = a+b
    mul = a*b
print(add_mul(5,5)) # (10, 25)
print(type(add_mul(5,5))) # <class 'tuple'>
```

⑦ tuple using range function :-

Example

```
num = tuple(range(1,4))
print(num)
# (1, 2, 3)
```

⑧ del():

→ used to delete the tuple completly.

```
a = ("apple", "banana", "cherry")
del a
print(a) # error
```

⑨ update the tuple

①
```
x = ("apple", "banana", "cherry")
y = list(x)
y[0] = "kiwi"
x = tuple(y)
print(x)
```
("apple", "kiwi", "cherry")

(convert in list)

⑩ add item

```
a = ("apple", "banana")
y = list(a)
y.append("cherry")
a = tuple(y)
print(a)
# ('apple', 'banana', 'cherry')
```

loop in tuple:-

①
```
a = ("boy", "girl", "men")
for x in a:
    print(x)
```
boy
girl
men

② Loop through the index No:-

```
a = ("boy", 'girl', 'men')
for i in range(len(a)):
    print(a[i]):
```
apple
banana
cherry

② Using a while loop:

```
a = ("apple", "banana", "cherry")
i = 0
while i < len(a):
    print(a[i])
    i = i + 1
```

```
# apple
  banana
  cherry
```

* Join 2 tuples:

① with +

```
t1 = ("a", "b", "c")
t2 = (1, 2, 3)
t3 = t1 + t2
print(t3)
```

```
# ('a', 'b', 'c',
  1, 2, 3)
```

② Multiply

```
a = ("apple", "banana")
b = a * 2
print(b)
```

```
# ('apple', 'banana'
  'apple', 'banana')
```

* Dictionaries *

→ these are used to store Value in key : Value pairs.

→ it is a Collection which is ordered, changeable and do not allow duplicates

i.e
```
thisdict = { "brand": "ford",
             "model": "End-even",
             "year": 1993 }
Print (thisdict)
```
#
```
{ 'brand': 'ford',
  'model': 'End-even',
  'year': 1993 }
```

① Dictionary Items :-

→ Dictionary items are ordered, changable, and does not allow duplicates

→ Items are Presented in key : Value pairs, can be Reffered to by using the key Name.

i.e
```
thisdict = { "brand": "ford",
             "model": "figo" }
Print (thisdict["brand"])
```
ford.

② Ordered & unordered.

→ When you say that dictionaries are Ordered, means that the item have a defined Order & that will Not change

→ Ordere as per Python 3.7. → as per 3.6, it is un ordered.

→ un order means that the item does not have defined order You can not Refer to Item by using an index.

⊙ Changable:-

→ it is changable means that we can change, add or remove item after the creation.

⊙ Duplicates Not Allowed:-

→ Can not have two item with same key:

i.e.:- a = { "brand": "ford",
"model": "figo",
"year": 2000,
"year": 2001 }
Print(a)

{ 'brand': 'ford',
'model': 'figo',
'year': 2001 }

⊙ Dictionary length:-

in above code
Print(len(a)) | # 3

⊙ Accessing Items:-

→ You can Access the item of a dictionary by Refering to its key name inside square brackets.

i.e. a = {"brand": "ford",
"model": "figo",
"year": 1964 }

~~X = this~~
X = a["model"] | # figo

② get()
X = a.get("model") | # figo

③ keys() → Returns all keys
X = a.keys()
→ dict_keys(['brand', 'model', 'year'])

① choose items :-

i.e. a = {"brand": "ford",
"model": "figo",
"year": 1964}
a["year"] = 2015
print(a)

{'brand': 'ford',
'model': 'figo',
'year': 2015}

② update() keyword

In above code ....
a.update({"year": 2020})

{'brand': 'ford',
'model': 'figo',
'year': 2020}

---

③ Add items :-

(i.e.) a = {"brand": "ford",
"model": "figo",
"year": 2020}
a["color"] = "red"
print(a)

{'brand': 'ford',
'model': 'figo',
'year': 2020
'color': 'red'}

④ update()

Above code ...
a.update({"color": 'red"})
Print (a)

Same o/p of above code

---

⑤ Remove items :-

① Pop() :- Remove item with specified key name.

i.e. a = {"brand":"ford",
"model":"figo",
"year":"1994"}

a.pop("model")
Print(a)

{'brand':'ford',
'year':1994}

② PopItem() :- Remove last inserted item

→ a.popitem()
Print(a)

{'brand':'ford',
'model':'figo'}

③ del() :- Remove the item with specified key name:

→ del a["model"]
Print(a)

{'brand':'ford',
'year':1994}

(OR)

del a
Print(a)

it will occure error.

⑤ clear() method empties the dictionary

→ a.clear()
Print(a)

{}

---

⑥ loop

i.e. for x in a:
Print(x)

brand
model
year

for x in a:
Print(a[x])

ford
figo
1994

⑦ Value() :- Return the Values:

for x in a.values():
Print(x)

ford
figo
1994

③ keys() → Return all the keys.

```
for x in a.keys():
    Print(x)
```

```
# brand
  model
  year
```

⑤ items() → Return key & Value:

```
for x, y in a.items():
    Print(x, y)
```

```
# 'brand' ford
  model figo
  year 1964
```

⊙ Copy Dictionary :-

① copy() :- will make a copy of dictionary.

```
my a = a.copy()
Print(my a)
```

```
# {'brand' : 'ford',
   'model' : 'figo',
   'year' : 1964}
```

② dict() :- will make a copy of dict() function.

```
my a = dict(a)
Print(my a)
```

```
# {'brand' : 'ford',
   'model' : 'figo',
   'year' : 1994}
```

⊙ Nested

i.e.

```
a = { "c1" : { "name" : "email",
               "year" : 2004 },
      "c2" : { "name" : "top",
               "year" : 2007 },
      "c3" : { "name" : "LMN",
               "year" : 2009 }
    }
Print(a)
```

```
# {'c1' : { 'name' : 'email',
            'year' : 2004 }

  'c2' : { 'name' : 'top',
           'year' : 2007 }

  'c3' : { 'name' : 'LMN',
           'year' : 2009 }
```

i.e. $c_1 = \{$ "name" : ABC,
"year" : 2005

$c_2 = \{$ "name" : "DEF",
"year" : 2001

$c_3 = \{$ "name" : "GHI",
"year" : 2002

myc = { "child1" : c_1,
"child2" : c_2,
"child3" : c_3 }

{'child1' : {'name' : 'ABC',
'year' : 2005}

'child2' : {'name' : 'DEF',
'year' : 2001}

'child3' : {'name' : 'GHI',
'year' : 2002}}

Other methods:

fromkeys() : Returns the specified key & value.

Setdefault() :- Returns the value of specified key.
if key does not exist : insert the key with specified value

SET

→ Sets are used to store multiple items in single variable

→ A set is a collection which is unordered, unchangable & unindexed.

i.e. set = {"apple", "banana", "cherry"}

print(set)

{'banana', 'cherry', 'apple'}

→ Set item : it is unordered, unchangable, do not allow duplicate value.

→ Unordered : that the item in a set do not have defined order

→ Unchangable : Meaning that we can not change the item after the set created.

" Once you created set, you can't change it's items, but you can remove items and add new items.

→ Duplication Not Allowed :

s = {"apple", "banana", "cherry", "apple"} # {'banana', 'cherry', 'apple'}

print(s)

→ length of set

thisset = {"apple", "banana", "cherry"} | #3

print(len(thisset)) | #<class 'set'>

print(type(thisset))

→ s = set(("apple", "banana", "cherry")) * set Constructor *

print(s)

{'cherry', 'apple', 'banana'}

⑥ Access Set Item :-

→ You cannot access items in set by Referring to an index or a key.

→ But you can loop through the set items using for loop.

→ You can ask if a specified value is present in a set by using in keyword.

i.e.
```
S = {"ABC", "DEF", "GHZ"}
for x in S:
Print (x)
```
DEF
GHZ
ABC

i.e.
```
S = {"ABC", "DEF", "GH2"}
Print ("banana" in S)
```
true

⑦ Addition

① add() : add one item to set.

```
S = {"ABC", "DEF", "GH2"}
S.add ("JKL")
Print (S)
```
{'JKL', 'ABC', 'GHZ', 'DEF'}

② update() : used to add the item from another set in to the current set.

```
a = {"apple", "banana"}
b = {"cherry", "mango"}
a.update(b)
Print (a)
```
{'apple', 'mango', 'cherry', 'banana'}

⑧ Remove

① Remove

S = {"
S.rem
Print

② discar

S.dis
Print

③ POP()

x = S
Print
Print

④ clea

S.
Pr

⑤ de

⊙ Remove ⊙

① Remove() :-

```
S={"ABC", "DEF", "GHIZ"}
S.remove("DEF")          # {'ABC', 'GHIZ'}
Print(S)
```

② discard() :-

```
S.discard("ABC")         # {'DEF', 'GHIZ'}
Print(S)
```

③ POP() :- It will Remove the list item.

:- Remember : Set is unordered, so you will Not know what item will get Removed.

```
X=S.POP()                # ABC
Print(X)
Print(S)                 # {'GHIZ', 'DEF'}
```

④ clean() :- that will empties the set :-

```
S.clean()
Print(S)                 # set()
```

⑤ del() :- that will delete the set Completely.

```
del S
Print(S)                 # Error.
```