Lazy

Python

range $\longrightarrow$

range(5) $\Rightarrow$ [0, 1, 2, 3, 4]

range(1, 5) $\Rightarrow$ [1, 2, 3, 4]

range(1, 10, 2) $\Rightarrow$ [1, 3, 5, 7, 9] for

shell

```
for i in [w1 w2 w3 w4]
         [{1..10}]
do
  ==
done
```

Python

```
for i in [listofelements]:
    line1
    line2
    line3
```

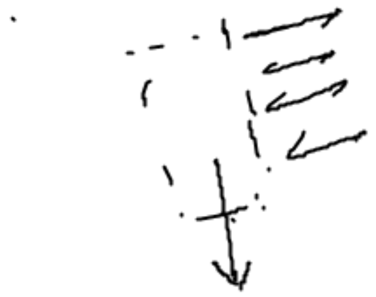
indentation

while shell

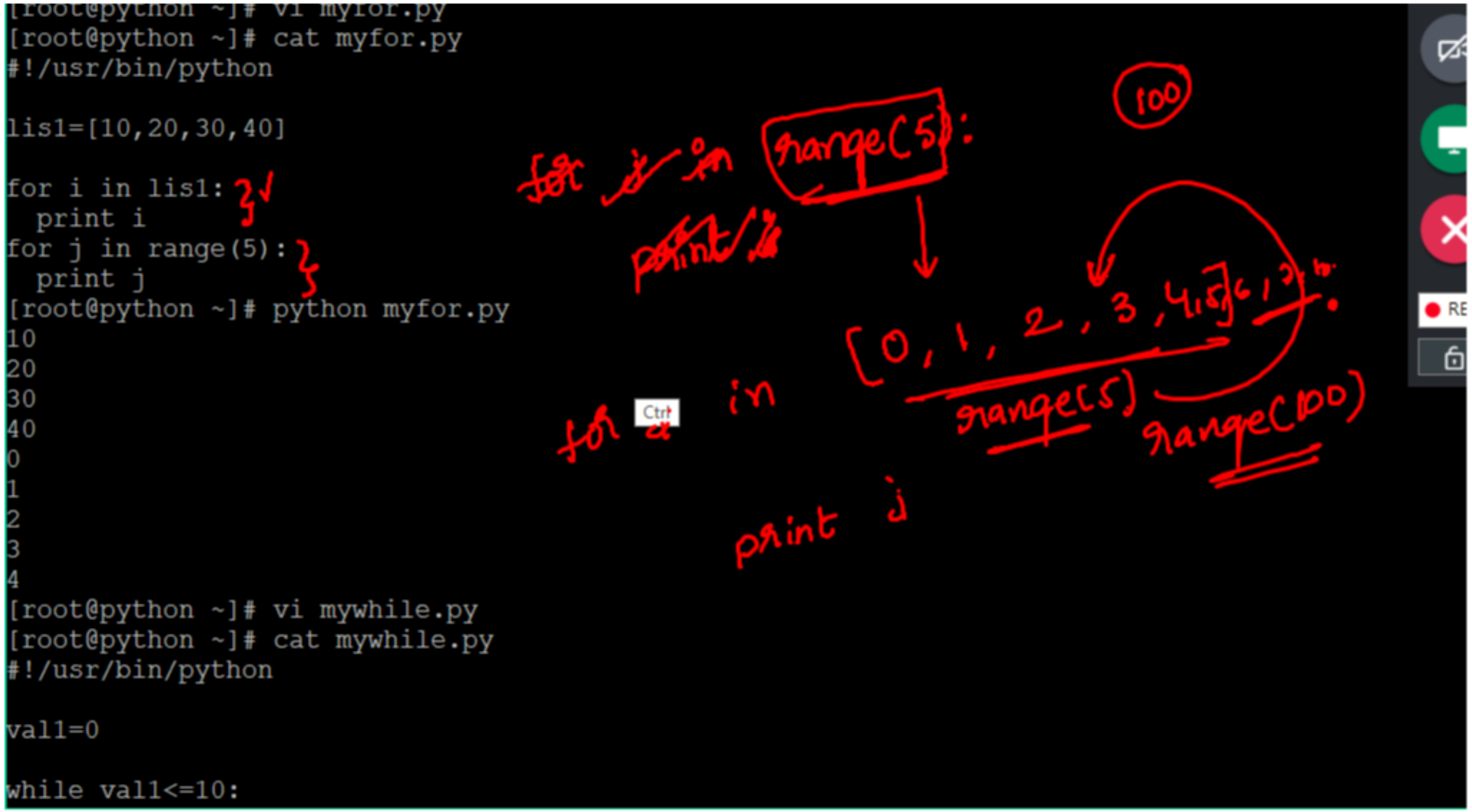
```
while cond.
do
        ≡
done
```

Python

```
while cond. :
```



indentation



```
[root@python ~]# vi myfor.py
[root@python ~]# cat myfor.py
#!/usr/bin/python

lis1=[10,20,30,40]

for i in lis1:
  print i
for j in range(5):
  print j
[root@python ~]# python myfor.py
10
20
30
40
0
1
2
3
4
[root@python ~]# vi mywhile.py
[root@python ~]# cat mywhile.py
#!/usr/bin/python

val1=0

while val1<=10:
```

Ide $\Rightarrow$ integrated developement Environment

$\downarrow$

[vs code
atom
intelli'id] tools supports pgmig languages

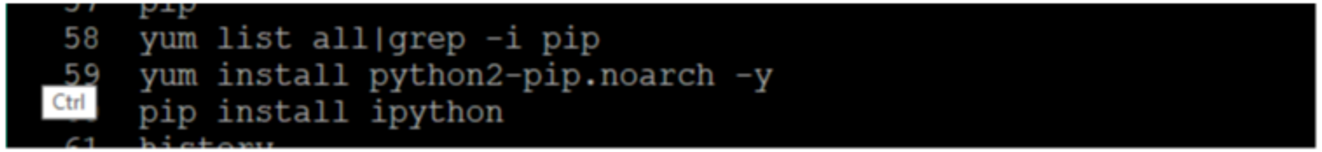
$\rightarrow$ Pycharm $\Rightarrow$ Python ide [windows]

$\rightarrow$ Ipython $\Rightarrow$ Python ide [linux] install /update /uninstall s/w linux

$\boxed{\text{yum}} \rightarrow$??

$\boxed{\text{pip}} \rightarrow$ python package manager

ipython $\rightarrow$ package in python



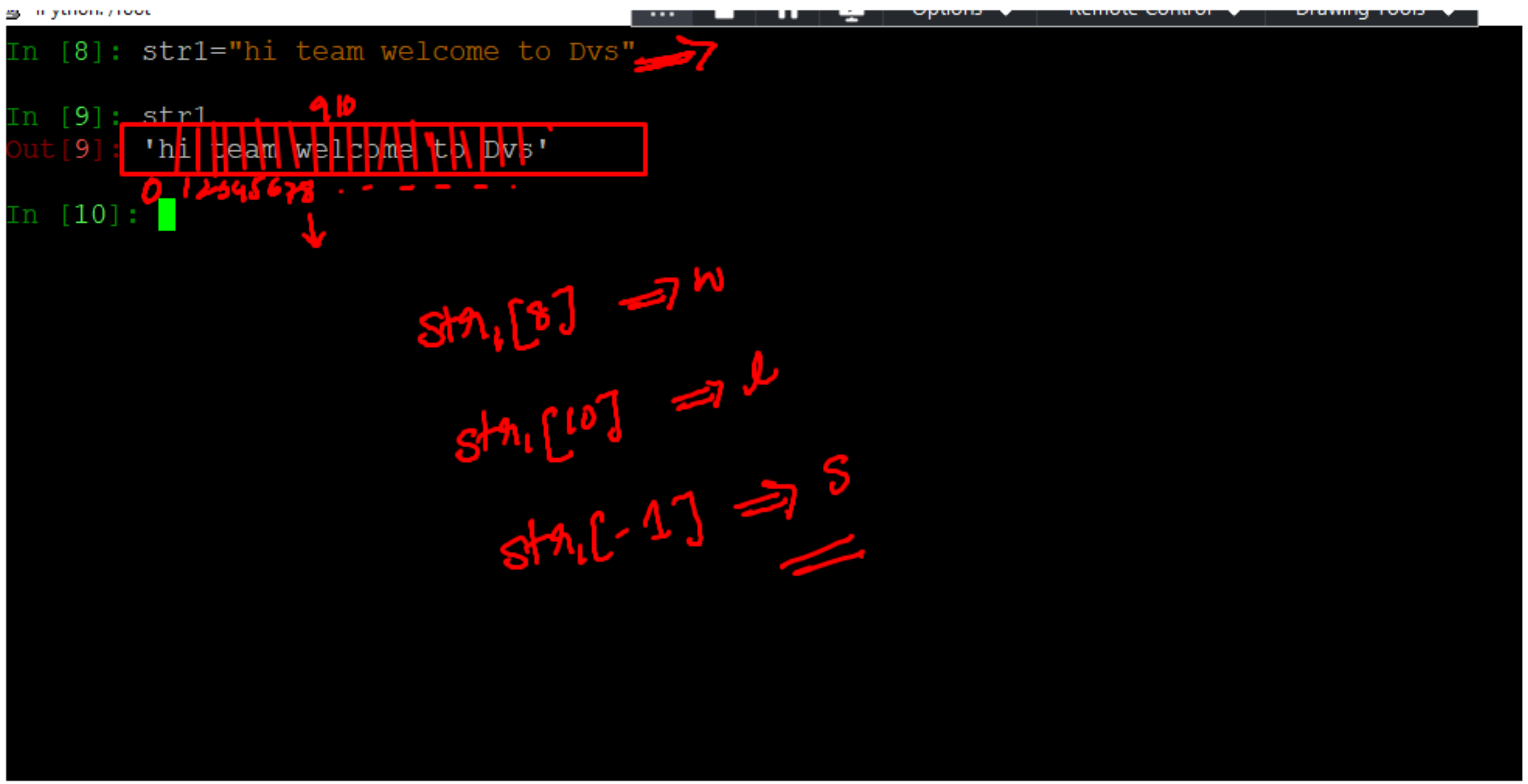
```
 57  pip
 58  yum list all|grep -i pip
 59  yum install python2-pip.noarch -y
 [illegible]  pip install ipython
 61  history
```

string ✓

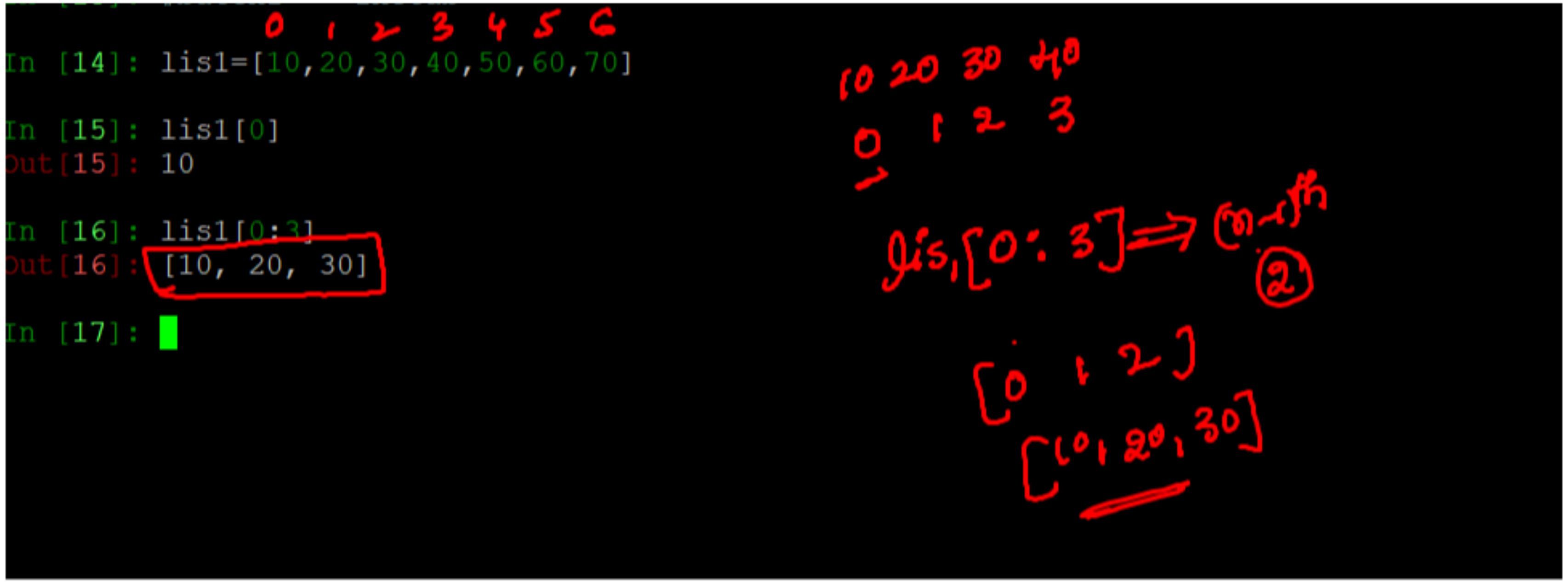
list ✓

tup ✓

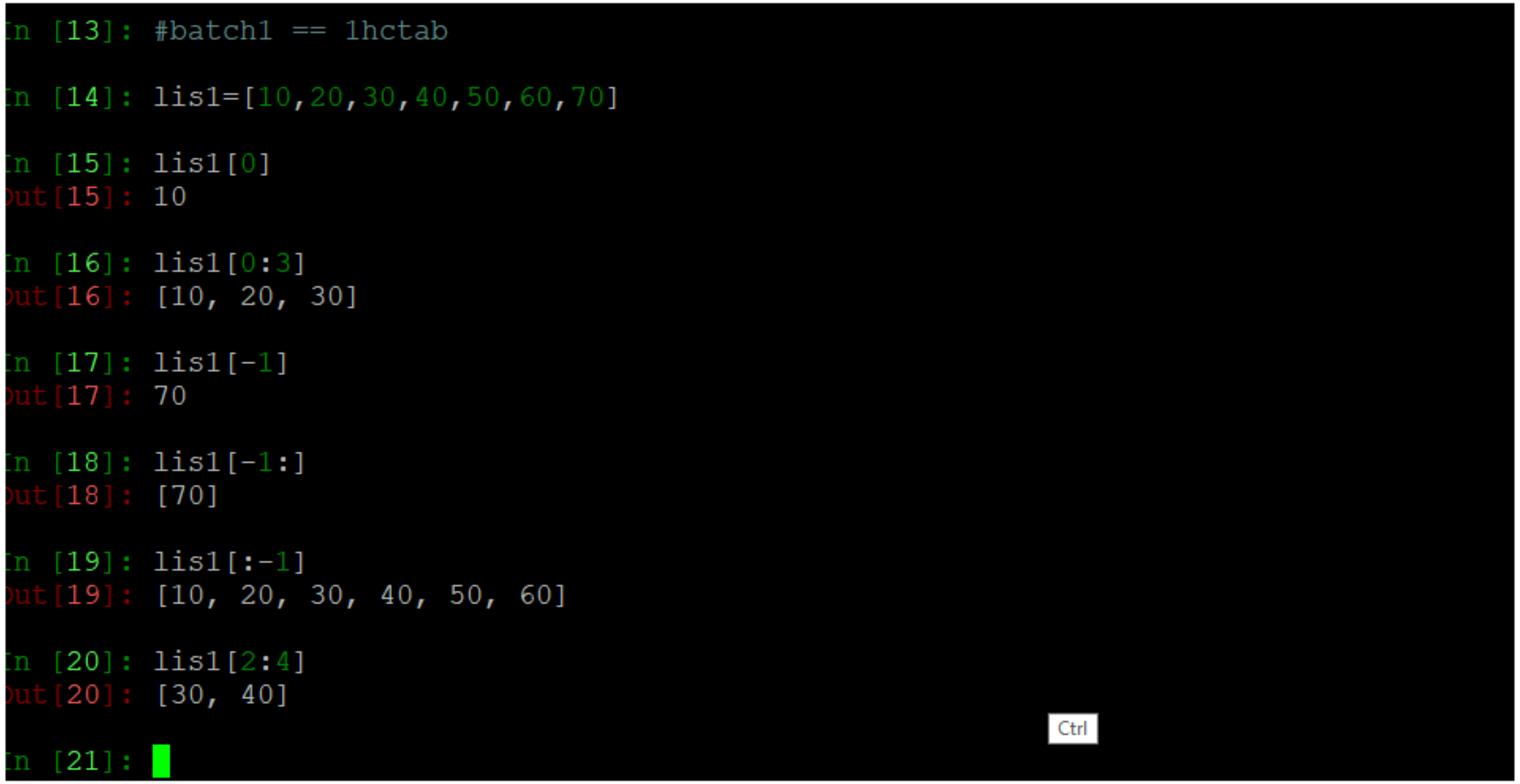
dict ✓



```
In [8]: str1="hi team welcome to Dvs"

In [9]: str1
Out[9]: 'hi team welcome to Dvs'

In [10]:
```

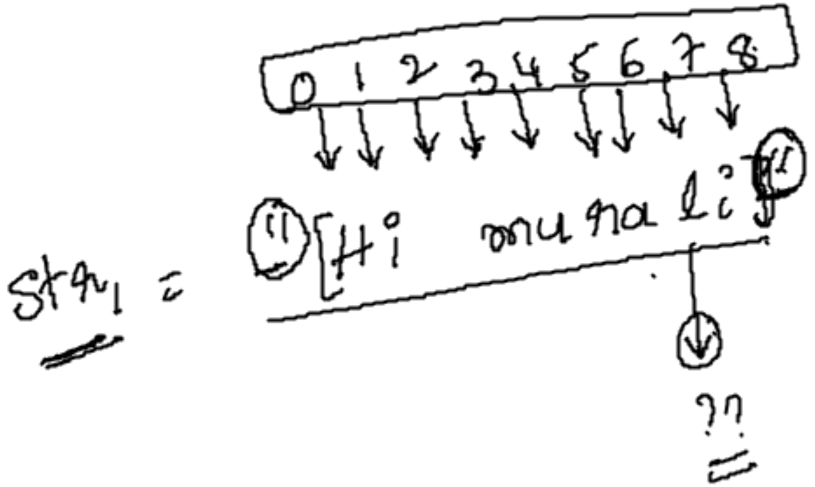


```
           0   1   2   3   4   5   6
In [14]: lis1=[10,20,30,40,50,60,70]

In [15]: lis1[0]
Out[15]: 10

In [16]: lis1[0:3]
Out[16]: [10, 20, 30]

In [17]:
```



```
n [13]: #batch1 == 1hctab

n [14]: lis1=[10,20,30,40,50,60,70]

n [15]: lis1[0]
ut[15]: 10

n [16]: lis1[0:3]
ut[16]: [10, 20, 30]

n [17]: lis1[-1]
ut[17]: 70

n [18]: lis1[-1:]
ut[18]: [70]

n [19]: lis1[:-1]
ut[19]: [10, 20, 30, 40, 50, 60]

n [20]: lis1[2:4]
ut[20]: [30, 40]

n [21]:
```

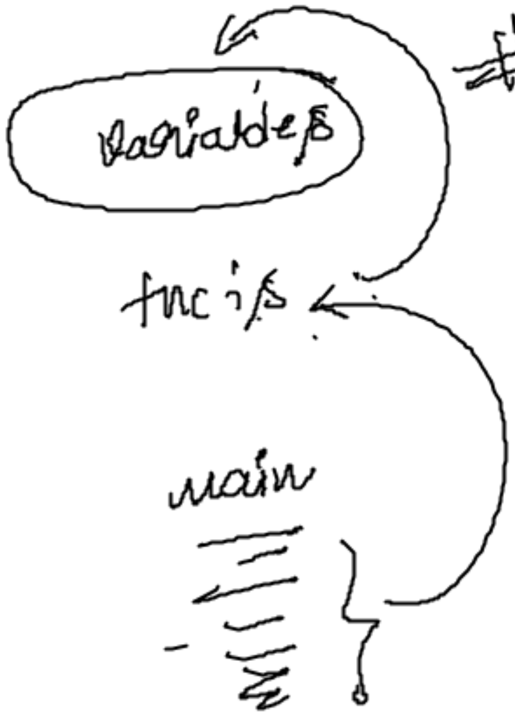
string



str1[7] ⇒ 'l'

ipython str1.(tabbutton)

Functions



Python myprgm.py => execute

Functions

??

```
#!/usr/bin/python
a=10 ; b=30

print a+b

a=40 ; b=100
print a+b

a=50 ; b=200
print a+b
```

{ a=x ; b=y
print a+b }

{ ~~a=x ; b=y~~
~~print a+b~~ }

{ ~~a=x ; b=y~~
~~print a+b~~ }

x=10 , y=30

x=40 y=100

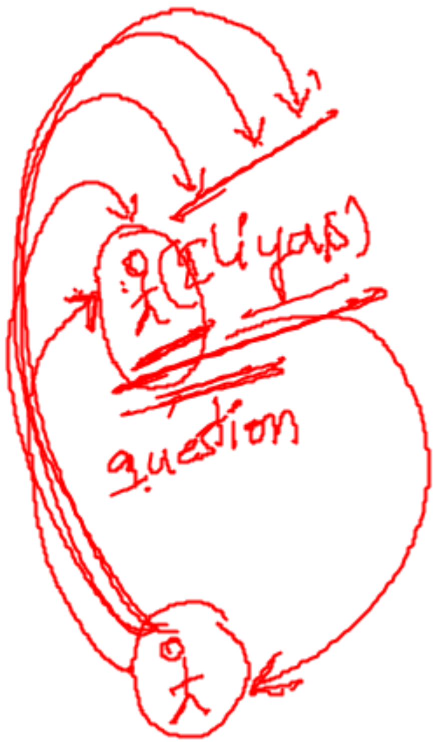
x=50 y=200

```
#!/usr/bin/python

def myfn(x, y):
    print x + y

myfn(10, 40)
myfn(100, 200)
```

once

1 2 3 4 5 --- --- 99

o/p
50
300



syntax;

fnc) declaration
```
def    fnname():
    ______
    ______
    ______
```

fnc) calling
```
fnname()
fnname()
```

types of fnc's

Required Arguments

Keyword "

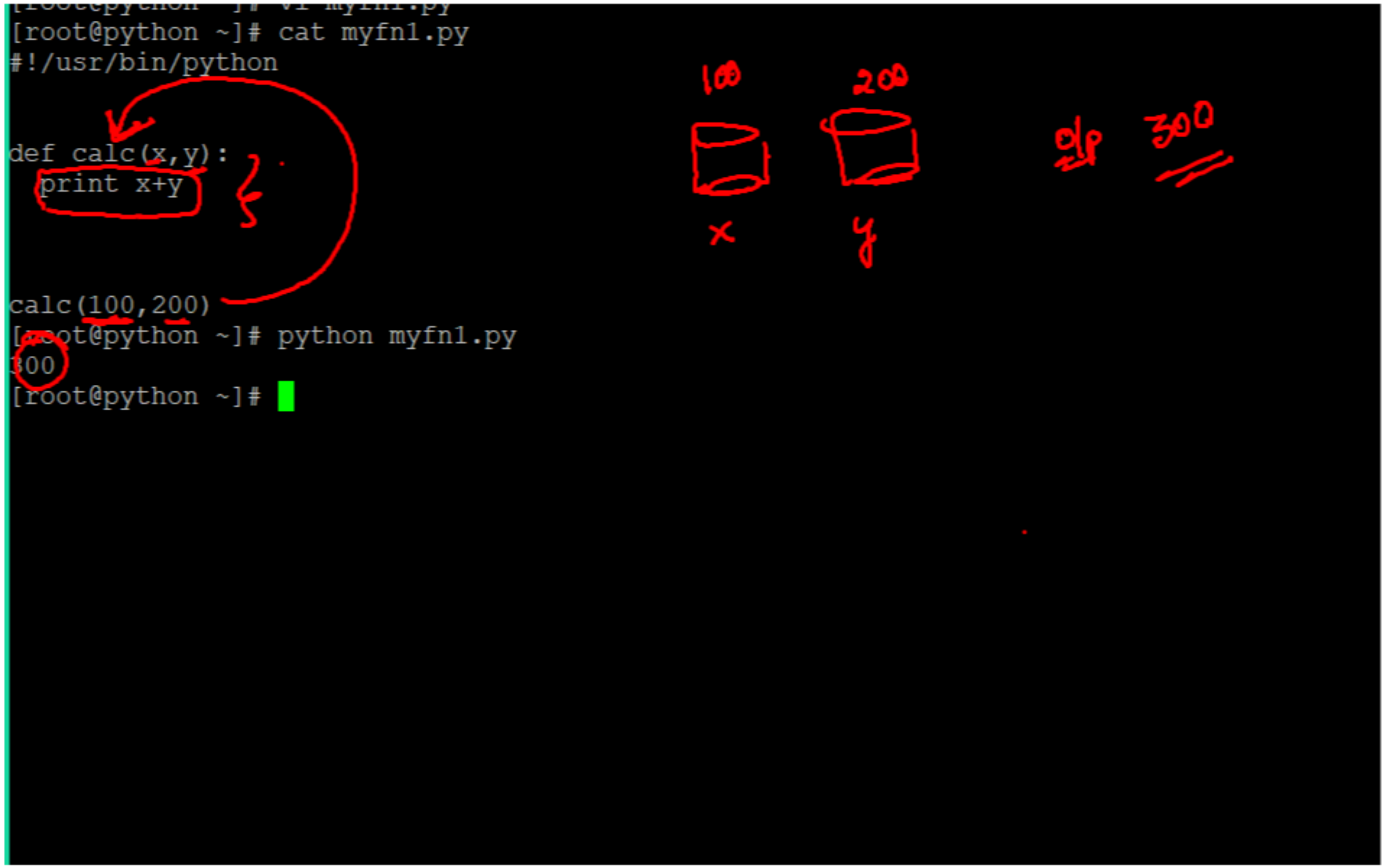
Default "

Variable Length "

{Arguments == parameters == values}

Required no. of Arguments

```
def    calc(x, y):
    print x+y

1   calc(10,20)   ✓

2   calc(30)      ✗
```



```
[root@python ~]# cat myfn1.py
#!/usr/bin/python


def calc(x,y):
  print x+y



calc(100,200)
[root@python ~]# python myfn1.py
300
[root@python ~]# 
```

o/p area is BTM price is 10,000,0000

```
def myflat(area, price):
{
    print "area is ", area, "price is", price*price
```

```
myflat("Btm", 10000)
```

o/p (Error)

```
myflat(2000, "HSR")
```

area

price