Taking multiple inputs from user in python:

There are two methods to accepts multiple values from the keyboard:

1.using split() method:

This function helps in getting a multiple inputs from user . it breaks the input by specific separator. If separator is not provided then any white space is a separator.

Syntax:

```
input().split(separator,maxsplit)
```

Example:

```
>>> x,y=input("enter two values:").split()

enter two values:67 78

>>> print(x)

67

>>> print(y)

78

>>> x,y=input("enter numbers:").split()

enter numbers:3456

Traceback (most recent call last):

  File "<pyshell#3>", line 1, in <module>

    x,y=input("enter numbers:").split()

ValueError: not enough values to unpack (expected 2, got 1)
```

```
>>>
```

Type-2:taking inputs at a time

Example:

```
>>> a,b=input("enter two values:").split()

enter two values:34 45

>>> print("first number {} and second number {}".format(a,b))

first number 34 and second number 45

>>>
```

Type-3:taking multiple inputs at a time

Example:

```
>>> a=list(map(int,input("enter values:").split()))

enter values:56 667 89 90 12 23 45 56

>>> print(a)

[56, 667, 89, 90, 12, 23, 45, 56]
```

2.using list comprehension:

We can create lists just like mathematical statements one line only. It is also used in getting multiple inputs from a user.

Example:

Type-1:taking three inputs at a time

```
>>> x,y,z=[int(x) for x in input("enter 2 values:").split()]

enter 2 values:34 45 56

>>> print(x)

34

>>> print(y)

45

>>> print(z)

56

>>>
```

Type-2:

```
>>> x,y=[int(x) for x in input("enter 2 values:").split()]

enter 2 values:45 56

>>> print("first number {} and second number {}".format(x,y))

first number 45 and second number 56
```

type-3:

```
>>> list=[int(x) for x in input("enter list of values:").split()]

enter list of values:12 13 14 15 16 27

>>> print(list)
```

```
[12, 13, 14, 15, 16, 27]

>>>
```

‘end’ parameter in print:

Print() comes with parameter called ‘end’. By default, the value of this parameter is ‘\n’. i.e the new line character. You can end a print statement with any character/string using this parameter.

Example:

```
print("welcome to",end="\n")

print("magneq software",end="")

print()

print("python",end="@")

print("magneq software")
```

‘sep’ parameter in print():

- ‘sep’ parameter is implemented in python 3.x
- It is used for formatting the output strings
- Print() function in python is space by default(softspace feature), which can be modified and can be made to any character or string as per our choice.

Example:

```
print('h','e','l','l','o',sep='#')

print('17','05','1990',sep='-')
print('h','e','l','l','o',sep='')


print('s','u',sep='',end='')

print('s')

print('17','05',sep='-',end='-1990\n')

print("sushma",'alla',sep='',end='@')

print("sowmya")
```

Control structures

Conditional statements:

1.if statement:

Syntax:

```
if <condition>:

   statements;
```

Example:

```
num=-34

if num>0:

   print("number is greater than zero")
```

2.if-else:

Syntax:

```
if <condition>:
    statements
else:
    statements
```

Example:

```
num=-34
if num>0:
   print("number is greater than zero")
else:
   print("number is less than zero")
```

Example-2:

```
num=35
if num%2==0:
   print("number is even")
else:
   print("number is odd")
```