```
# return keyword
def d1(a, b):
    return a, b
d = d1(5,10)
print(type(d)) # <class 'tuple'>
print(d) # (5, 10)
```

```
# print statement
def d2(a, b):
    print(a, b)
d = d2(5,10) # 5 10
print(type(d))# <class 'NoneType'>
```

```
# yield keyword
def d3(a,b):
    yield a
    yield b
d = d3(10,20)
print(type(d)) # <class 'generator'>
print(next(d)) # 10
print(next(d)) # 20
print(next(d)) # StopIteration
```

```
# tuple
t = (1,2,3,4,5)
print(type(t)) #<class 'tuple'>

# generator comprehension
t = (i for i in range(10))
print(t) #<generator object <genexpr> at 0x0000003C61134AC8>
print(type(t)) #<class 'generator'>
print(list(t)) # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

# tuple comprehension explicitly
t = tuple(i for i in range(10))
print(t) #(0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
print(type(t)) #<class 'tuple'>
```

```
# generator comprehension
def show(a, b):
    while a<=b:
        yield a
        a+=1
result = show(1,5)
print(result) # <generator object show at 0x000000D26C292900>
print(next(result)) # 1
print(next(result)) # 2
print(next(result)) # 3
print(next(result)) # 4
print(next(result)) # 5
print(next(result)) # StopIteration

or
for i in result:
    print(i) # 1 2 3 4 5
```