Python Programming

Tuples Unpacking

Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)

* For unpacking

```
lst = 1, 2, 3, 4, 5
a, b, c, d, e = lst      # normal unpacking
a, _, _, _, _ = lst      # what If i don't care? use _ a common notation

# what if I am not sure from the total number? use *
    # * here refers to varying number of arguments
a, b, *c = lst
print(c)    # [3, 4, 5]

*a, b, c = lst
print(a)    # [1, 2, 3]

a, *b, c = lst
print(b)    # [2, 3, 4]

a, *b, c, d = lst
print(b)    # [2, 3]

# Although we can do the same with slicing
# but the * operator is more elegant and makes code simpler!

def f(*items):
    print(items)  # (1, 2, 3, 4)

f(1, 2, 3, 4)
```

Unpacking

```
3   lst = [1, 2, 3]
4   print(lst)      # [1, 2, 3]
5   print(*lst)     # 1 2 3   unpack first, then print: print received 3 arguments NOT 1
6
7
8   def f(a, b):
9       print(a+b)
10
11  #f(*lst)    f() takes 2 positional arguments but 3 were given
12
13  lst1 = [1, 2, 3]
14  lst2 = [4, 5, 6]
15  conc = [*lst1, *lst2]
16  print(conc) # [1, 2, 3, 4, 5, 6]
17
```

Deep unpacking

```
1
2   lst = 1, 2, (5, 6)
3
4   #ValueError: not enough values to unpack (expected 4, got 3)
5   #a, b, c, d = lst
6
7   print(len(lst)) # 3
8
9   # deep unpacking
10  a, b, (c, d) = lst
11  print(a, b, c, d)   # 1 2 5 6
12
13  t = 1, 2, 3, (4, (5, 6))
14  a, b, c, (d, (e, f)) = t
15  print(a, b, c, d, e, f) # 1 2 3 4 5 6
16
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."