

# 16 tuples

Monday, May 9, 2022 10:03 PM

## Tuples 1 motivation

Tuesday, April 26, 2022 10:15 PM

```
2 drinks = ["2% milk", "Diet coke, large", "strawberry milkshake"]
3 food = ["burger", "veggie burger"]
4 condiments = ["Ranch dressing", "ketchup", "wasabi"]
5
6 bag = [drinks, food, condiments]
7
8 #      0           1           2
9 # 0 ["2% milk",      "Diet coke, large", "strawberry milkshake"]
10 # 1 ["burger",      "veggie burger"]
11 # 2 ["Ranch dressing", "ketchup",      "wasabi"]
12
13 ...
14 looks for the food(string) in the bag(list of lists) and returns the loc
15 parameters : food(string), bag(list of lists)
16 return coordinates which list and at which index
17 ...
18 def find_my_food(bag, food):
19
20     #do a 'for' loop over each list
21     # and a 'for' loop within each list, looking for a match
22
23     loc1 = 0 #todo temp value : replace once 'for' loops are done
24     loc2 = 2 #todo temp value : replace once 'for' loops are done
25
26     location = "?? we want to return two things, loc1 and loc2....|"
27
28     return location
29
30 found = find_my_food(bag, "strawberry milkshake")
```

Tuples

You can return multiple values  
in a comma separated sequence

Demo [return (midX, midY)]  
From code w/ what type gets returned then?

Demo [midPoint = midXY(...)]

If we wanted it as 2 variables?

[mx, my = midPoint(p1x, p1y, p2x, p2y)]

A tuple is a <sup>» common separate</sup>  
sequence of values,  
(often enclosed in parens)

You can pack: unpack using assignment  
statements

V = (1, 4, "Handy") Pack  
(a, b, c) = V unpack

They can be passed into functions  
as arguments

But! They are immutable: You can't change them!

```

26     location = (loc1, loc2) # loc 1, loc2 bundled together
27     print(type(location))
28
29     return location
30
31 found = find_my_food(bag, "strawberry milkshake")
32
33 f1, f2 = found #unpack the tuple
34
35 print(bag[f1][f2])

```

## Tuples packing and unpacking

Tuesday, April 26, 2022 10:20 PM

```
1 my_tuple = (1,2,3) #packing a tuple
2
3 a,b,c = my_tuple #unpacking a tuple
4
5 #d,e = my_tuple #this won't work: need same num of variables
6           #as values in the tuple!
7
8 (f,g,h) = my_tuple #still just three variables
9           #this did not make a new tuple|
```

```
1 a = 1
2 b = 2
3 c = 3
4
5 v = a, a, c
6
7 print(v)
8
9 # What does this print?
10 # A: 1 2 3
11 # B: 1 1 3
12 # C: (1, 2, 3)
13 # D: (1, 1, 3)|
```

(answer is D)

```
3 def a():
4     print("a")
5 def b():
6     print("b")
7 def c():
8     print("c")
9
10
11 letters = [a,b,c]
12 a=7
13 print(a)
14
15 letters[0]()
16
17
18 #this will print :
19 # 7
20 # a|
```

They are sequences!

for i in	[1, 3]	list
	in (1,3)	tuple
	in range(1,5,2)	range
print(i)		

Write down what is in memory at each step:  
will any of the steps crash?

```
1 a,b, c = 7, 5, 3
2 (z,x) = c, b
3 w = (z,x)
4 print((x,z))
5 print(w)
6 v = (x, z, c, w)
7 print(v)
8 a, b = v
```

```
a,b,c = 7, 5, 3
(z,x) = c, b
w = (z,x)
print((x,z))
print(w)
v = (x, z, c, w)
print(v)
a, b = v
```

```
1 #lists are sequences
2 for letter in ["a","b"]:
3     print(letter)
4
5 #tuples are sequences!
6 for number in (1,3):
7     print(number)
8
9 #can you write a for loop that uses 'range'
10 #that prints the same numbers?
11
12 #A: range(1,3)
13 #B: range(3)
14 #C: range(1,3,2)
15 #D: range(1,5,2)
16
17 #answer:
18 for i in range(1,5,2):
19     print(i)
```

## Lists vs Tuples

Tuesday, February 23, 2021 12:45 PM

### Lists vs Tuples

Tuples are immutable → can't mutate  
change

Lists are mutable → can mutate

### Examples

A-tuple = (1, 2, 3)

A-list = [1, 2, 3]

A-tuple[1] = φ X Error!

A-list [1] = φ ✓ Ok!

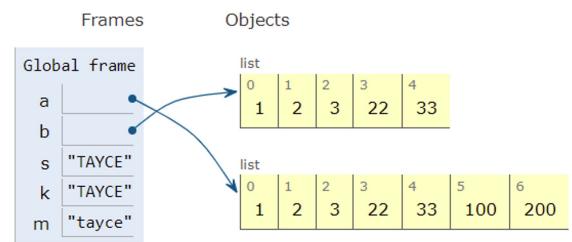
## List extend vs +

Tuesday, March 1, 2022 11:12 PM

Write code in Python 3.6

(drag lower right corner to resize code editor)

```
1 a = [1,2,3]
2
3 a.extend([22,33]) #extend mutates it in place
4
5 b = a          #just points to the existing list
6 #we'll do copies in an upcoming lecture
7
8 a = a + [100,200] # plus makes a whole new list!
9      # and the old list is left...as long as something
10     #is pointing to it it'll stay around
11     # (won't get garbage collected)
12     # more advanced CS classes go into how to know that
13     # it's still being used!
14 s = "TAYCE"
15 k = s #fresh copy, not pointing to the same one
16 s.lower() #this doesn't change anything, it returns something
17 m = s.lower() #this makes a new copy and saves it back
```



## QOTD

What does the following code print?

```
a = ["Tony", "Steve", "Natasha", "T'Challa", "Carol"]
b = a[2:3] + [a[4]]
b.extend(a[:2])
print(b[2], b[2:3])
```

## Return values vs effects

Wednesday, February 23, 2022 10:52 AM

### Return Values vs Effects

t.forward(100) ← Most turtle methods  
change the state of the  
turtle object

A s.lower() ← Most string methods don't change  
the original string → but  
return a new string

A =

so save it to a variable!

why?

strings are immutable!

s[0] = 2 X