13.4. Tuples as Return Values

Functions can return tuples as return values. This is very useful — we often want to know some batsman's highest and lowest score, or we want to find the mean and the standard deviation, or we want to know the year, the month, and the day, or if we're doing some ecological modeling we may want to know the number of rabbits and the number of wolves on an island at a given time. In each case, a function (which can only return a single value), can create a single tuple holding multiple elements.

For example, we could write a function that returns both the area and the circumference of a circle of radius r.

Save & Run Original - 1 of 1 Show in CodeLens

```
def circleInfo(r):
    """ Return (circumference, area) of a circle of radius r """
    c = 2 * 3.14159 * r
    a = 3.14159 * r * r
    return (c, a)

print(circleInfo(10))

```

```
(62.8318, 314.159)
```

Activity: 1 -- ActiveCode (ac12_3_1)

Again, we can take advantage of packing to make the code look a little more readable on line 4

Save & Run Original - 1 of 1 Show in CodeLens

```
def circleInfo(r):
    """ Return (circumference, area) of a circle of radius r """
    c = 2 * 3.14159 * r
    a = 3.14159 * r * r
    return c, a

print(circleInfo(10))

```

```
(62.8318, 314.159)
```

Activity: 2 -- ActiveCode (ac12_3_2)

It's common to unpack the returned values into multiple variables.

Save & Run Original - 1 of 1 Show in CodeLens

```
def circleInfo(r):
    """ Return (circumference, area) of a circle of radius r """
    c = 2 * 3.14159 * r
    a = 3.14159 * r * r
    return c, a

print(circleInfo(10))

circumference, area = circleInfo(10)
print(circumference)
print(area)

circumference_two, area_two = circleInfo(45)
print(circumference_two)
print(area two)
```

```
(62.8318, 314.159)
62.8318
314.159
282.7431
6361.71975
```

Activity: 3 -- ActiveCode (ac12_4_5)

Check your Understanding

tuples-3-1: If you want a function to return two values, contained in variables x and y, which of the following methods will work?

- [ ] A. Make the last two lines of the function be "return x" and "return y"
- [x] B. Include the statement "return [x, y]"
- [x] C. Include the statement "return (x, y)"
- [x] D. Include the statement "return x, y"
- [ ] E. It's not possible to return two values; make two functions that each compute one value.

Check me | Compare me

✔ Correct.
B. return [x,y] is not the preferred method because it returns x and y in a mutable list rather than a tuple which is more efficient. But it is workable.
C. return (x, y) returns a tuple.
D. return x, y causes the two values to be packed into a tuple.

Activity: 4 -- Multiple Choice (question12_4_1)

Define a function called `information` that takes as input, the variables `name`, `birth_year`, `fav_color`, and `hometown`. It should return a tuple of these variables in this order.

Save & Run | 5/14/2021, 2:30:55 PM - 3 of 3 | Show in CodeLens

```
1 def information(name,birth_year,fav_color,hometown):
2     f=(name,birth_year,fav_color,hometown)
3     return f
4
```

Activity: 5 -- ActiveCode (ac12_3_3)

Result	Actual Value	Expected Value	Notes
Pass	('Sar...uis')	('Sar...uis')	Testing that information returns the correct tuple on input ('Sarah', 1996, 'purple', 'St. Louis')

Expand Differences

You passed: 100.0% of the tests

Define a function called `info` with the following required parameters: `name`, `age`, `birth_year`, `year_in_college`, and `hometown`. The function should return a tuple that contains all the inputted information.

Save & Run | 5/14/2021, 2:32:04 PM - 2 of 2 | Show in CodeLens

```
1 def info(name, age, birth_year, year_in_college, hometown):
2     f = (name, age, birth_year, year_in_college, hometown)
3     return f
4
5
6
```

Activity: 6 -- ActiveCode (ac12_3_4)

Result	Actual Value	Expected Value	Notes
Pass	('Tin...oit')	('Tin...oit')	Testing the function info on input: name='Tina', age=20, birth_year=1996, year_in_college='sophomore', hometown='Detroit'.

Expand Differences

You passed: 100.0% of the tests

You have attempted 7 of 6 activities on this page

13.3. Tuple Assignment with Unpacking"> 13.5. Unpacking Tuples as Arguments to Function Calls">

ple Assignment with Unpacking">

13.5. Unpacking Tuples as Arguments to Function Calls">Next Section - 13.5. Unpacking Tuples as Arguments to Function Calls

| Back to top