Solutions 2

Jumping Rivers

Question 1

Consider the following simple function

```
v = 5
def Fun1():
    v = 0
    return v

Fun1()
```

```
## 0
```

1. Why does the final line return 0 and not 5.

```
print('Fun1 uses the local variable v')
```

```
## Fun1 uses the local variable v
```

2. Delete line 3 in the above piece of code. Now change `Fun1()` to allow `v` to be passed as an argument, i.e. we can write `Fun1(5)`. Call this function to make sure it works.

```
def Fun1(v):
    return v

Fun1(10)
```

```
## 10
```

Question 2

Consider the two functions defined below:

```
def Fun2(x=10):
    return x

def Fun3(x):
    return x
```

1. Why does

```
Fun2()
```

```
## 10
```

work, but this raises an error

```
Fun3()
```

```
print("Fun3 expects an argument x, but \
we haven't given one and there is \
no default.")
```

```
## Fun3 expects an argument x, but we haven't given one and there is no default.
```

2. Change `Fun2` so that it returns `x*x`.

```
def Fun2(x=10):
    return x*x
```

Question 3

```
a = 2
total = 0
for blob in range(a, 5):
    total = total + blob
total
```

```
## 9
```

1. In the code above, delete line 1. Now put the above code in a function called `Fun5`, where `a` is passed as an argument, i.e. we can call `Fun5(1)`

```
def Fun5(a):
    total = 0
    for blob in range(a, 6):
        total = total + blob
    return total
Fun5(1)
```

```
## 15
```

2. Alter the code so that the `for` loop goes from `a` to `b`, rather than `a` to 5. Allow `b` to be passed as an argument, i.e. we can call `Fun5(1,6)`.

```
def Fun5(a, b):
    total = 0
    for blob in range(a, b):
        total = total + blob
    return total
```

3. Change `Fun5` so that it has default arguments of `a = 1` and `b = 10`.

```
def Fun5(a=1, b=10):
    total = 0
    for blob in range(a, b):
        total = total + blob
    return total
```

4. The `range()` function also has a step argument, so to create the sequence `1, 3, 5` we would write `range(1, 6, 2)`. Alter the code such that `Fun5()` can now go up in steps of `c`. Allow `c` to be passed as an argument.

```
def Fun5(a=1, b=10, c=1):
    total = 0
    for blob in range(a, b, c):
        total = total + blob
    return total
```