Practical 1

Jumping Rivers

Practical 1

The aim of this practical is to review our knowledge of functions, for loops and if statements.

Basic functions

Consider the following simple function

```
v = 5
Fun1 = function() {
    v = 0
    return(v)
}
Fun1()
```

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

Default arguments:

```
Fun2 = function(x = 10) {
    return(x)
}

Fun3 = function(x) {
    return(x)
}
```

1. Why does

   ```
   Fun2()
   ```

   work, but this raises an error

   ```
   Fun3()
   ```

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

if statements.

```
Fun4 = function(x) {
    if (x == 5) {
        y = 0
    } else {
        y = 1
    }
    return(y)
}
```

Change `Fun4` so that it:

- returns 1 if x is positive;
- returns -1 if x is negative;
- returns 0 if x is zero.

Change `Fun4()` so it errors if x is positive

for loops.

```
total = 0
for (i in 1:5) {
    total = total + i
}
total
```

The `for` loop above calculates

$$\sum_{i=1}^{5} i = 1 + 2 + 3 + 4 + 5$$

1. What is the final value of `total` in the above piece of code?
2. Change the above loop to calculate the following summations:

$$\text{(i)} \sum_{i=1}^{20} (i + 1)$$

$$\text{(ii)} \sum_{j=-10}^{15} j$$

1. Rewrite the two loops using the `sum()` function. For example, the for loop in the first example can be written as `sum(1:5)`

More functions, for loops and signalling conditions:

```
a = 2
total = 0
for (blob in a:5) {
    total = total + blob
}
```

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)`

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(1,5)`.

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

4. Change `Fun5` so that it messages the user the total after each iteration and stops the function if the total has surpassed 50.

Solutions

The solutions can be viewed via

```
library("jrAdvPackage")
vignette("solutions1", package = "jrAdvPackage")
```