Worksheet 1

Allen B. Downey and Keith L. Mannock

September 10, 2017

These are the programming exercises for the first session which you should complete before we next meet. There are usually several ways to solve most programming exercises so just get something to work first and then refine it (refactor).

Always experiment when you are trying out a new feature and do look at other questions and exercises above and beyond the worksheets. Programming is all about practice and experience, and you don't get that by doing the minimum.

Whenever you are experimenting with a new feature, you should try to make mistakes. For example, in the "Hello, world!" program, what happens if you leave out one of the quotation marks? What if you leave out both? What if you spell `print` wrong?

This kind of experiment helps you remember what you read; it also helps when you are programming, because you get to know what the error messages mean. It is better to make mistakes now and on purpose than later and accidentally.

General

1. In a print statement, what happens if you leave out one of the parentheses, or both?
2. If you are trying to print a string, what happens if you leave out one of the quotation marks, or both?
3. You can use a minus sign to make a negative number like`-2`. What happens if you put a plus sign before a number? What about `2++2`?
4. In math notation, leading zeros are ok, as in `02`. What happens if you try this in Python?
5. What happens if you have two values with no operator between them?

Python as a calculator

Start the Python interpreter and use it as a calculator.

1. How many seconds are there in 42 minutes 42 seconds?
2. How many miles are there in 10 kilometres? Hint: there are 1.61 kilometres in a mile.

3. If you run a 10 kilometre race in 42 minutes 42 seconds, what is your average pace (time per mile in minutes and seconds)? What is your average speed in miles per hour?

4. We've seen that `n = 42` is legal. What about `42 = n`?

5. How about `x = y = 1`?

6. In some languages every statement ends with a semi-colon, `;`. What happens if you put a semi-colon at the end of a Python statement?

7. What if you put a period at the end of a statement?

8. In math notation you can multiply x and y like this: xy. What happens if you try that in Python?

9. The volume of a sphere with radius r is $\frac{4}{3}\pi r^3$. What is the volume of a sphere with radius 5?

10. Suppose the cover price of a book is $24.95, but bookstores get a 40% discount. Shipping costs $3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

11. If I leave my house at 6:52 am and run 1 mile at an easy pace (8:15 per mile), then 3 miles at tempo (7:12 per mile) and 1 mile at easy pace again, what time do I get home for breakfast?

Functions

1. Write a function named `right_justify` that takes a string named `s` as a parameter and prints the string with enough leading spaces so that the last letter of the string is in column 70 of the display.

```
>>> right_justify('monty')
                                                                 monty
```

   Hint: Use string concatenation and repetition. Also, Python provides a built-in function called `len` that returns the length of a string, so the value of `len('monty')` is 5.

2. A function object is a value you can assign to a variable or pass as an argument. For example, `do_twice` is a function that takes a function object as an argument and calls it twice:

```
def do_twice(f):
    f()
    f()
```

   Here's an example that uses `do_twice` to call a function named `print_spam` twice.

```
def print_spam():
    print('spam')

do_twice(print_spam)
```

Type this example into a script and test it.

3. Modify `do_twice` so that it takes two arguments, a function object and a value, and calls the function twice, passing the value as an argument.
4. Copy the definition of `print_twice` from earlier in this chapter to your script.
5. Use the modified version of `do_twice` to call `print_twice` twice, passing `'spam'` as an argument.
6. Define a new function called `do_four` that takes a function object and a value and calls the function four times, passing the value as a parameter. There should be only two statements in the body of this function, not four.

Supplemental

Note: This exercise should be carried out using only the statements and other features we have learned by the end of the first session (notes, 1 through 3).

1. Write a function that draws a grid like the following:

```
+ - - - - + - - - - +
|         |         |
|         |         |
|         |         |
|         |         |
+ - - - - + - - - - +
|         |         |
|         |         |
|         |         |
|         |         |
+ - - - - + - - - - +
```

Hint: to print more than one value on a line, you can print a comma-separated sequence of values:

```
print('+', '-')
```

By default, `print` advances to the next line, but you can override that behavior and put a space at the end, like this:

```
print('+', end=' ')
print('-')
```

The output of these statements is `'+ -'`.

A `print` statement with no argument ends the current line and goes to the next line.

2. Write a function that draws a similar grid with four rows and four columns.