1 Basics

Exercise 1.1: The interpreter

Open the Python interpeter. What happens when you input the following statements:

(a) `3 + 1`

(b) `3 * 3`

(c) `2 ** 3`

(d) `"Hello, world!"`

Exercise 1.2: Scripts

Now copy the above to a script, and save it as `script1.py`. What happens if you run the script? (try: `python script1.py`). Can you fix this (hint: use the `print` function)

Exercise 1.3: More interpreter

Explain the output of the following statements if executed subsequently:

(a) `'py' + 'thon'`

(b) `'py' * 3 + 'thon'`

(c) `'py' - 'py'`

(d) `'3' + 3`

(e) `3 * '3'`

(f) `a`

(g) `a = 3`

(h) `a`

Exercise 1.4: Booleans

Explain the output of the following statements:

(a) `1 == 1`

(b) `1 == True`

(c) `0 == True`

(d) `0 == False`

(e) `3 == 1 * 3`

(f) `(3 == 1) * 3`

(g) `(3 == 3) * 4 + 3 == 1`

(h) `3**5 >= 4**4`

Exercise 1.5: Integers

Explain the output of the following statements:

(a) `5 / 3`

(b) `5 % 3`

(c) `5.0 / 3`

(d) `5 / 3.0`

(e) `5.2 % 3`

(f) `2001 ** 200`

Exercise 1.6: Floats

Explain the output of the following statements:

(a) `2000.3 ** 200` (compare with above)

(b) `1.0 + 1.0 - 1.0`

(c) `1.0 + 1.0e20 - 1.0e20`

Exercise 1.7: Variables

Write a script where the variable `name` holds a string with your name. Then, assuming for now your name is *John Doe*, have the script output: `Hello, John Doe!` (and obviously, do not use `print "Hello, John Doe!"`.

Exercise 1.8: Type casting

Very often, one wants to "cast" variables of a certain type into another type. Suppose we have variable `x = '123'`, but really we would like `x` to be an integer.

This is easy to do in Python, just use `desiredtype(x)`, e.g. `int(x)` to obtain an integer.

Try the following and explain the output

(a) `float(123)`

(b) `float('123')`

(c) `float('123.23')`

(d) `int(123.23)`

(e) `int('123.23')`

(f) `int(float('123.23'))`

(g) `str(12)`

(h) `str(12.2)`

(i) `bool('a')`

(j) `bool(0)`

(k) `bool(0.1)`

2 Control flow

Disclaimer: Some of the following problems are inspired by problems from www.projecteuler.net. Have a look if you are interested, there are some great challenges and Python is an excellent tool for solving them.

Exercise 2.1: Range

Type `range(5)` in the interpreter, what does the interpreter return? So what does `for i in range(5)` mean?

Let's also find out whether the interpreter can help us understand the object '`range(5)`' better. Type `type(range(5))` in the interpreter. More on this soon!

Exercise 2.2: For loops

Use a `for` loop to:

(a) Print the numbers 0 to 100

(b) Print the numbers 0 to 100 that are divisible by 7

(c) Print the numbers 1 to 100 that are divisible by 5 but not by 3

(d) Print for each of the numbers $x = 2, \ldots 20$, all numbers that divide x, excluding 1 and x. Hence, for `18`, it should print `2 3 6 9`.

Hint: see https://docs.python.org/2.7/library/functions.html#range.

Exercise 2.3: Simple while loops

Instead of using a for loop, use a while loop to:

(a) Print the numbers 0 to 100

(b) Print the numbers 0 to 100 that are divisible by 7

Exercise 2.4: While loops

Use a `while` loop to find the first 20 numbers that are divisible by 5, 7 and 11, and print them Hint: store the number found so far in a variable.

Pseudo-code:

```
number found = 0
x = 11
while number found is less than 20:
    if x is divisible by 5, 7 and 11:
        print x
        increase number found by 1
    increase x by 1
```

Exercise 2.5: More while loops

The smallest number that is divisible by 2, 3 and 4 is 12. Find the smallest number that is divisible by all integers between 1 and 10.

Exercise 2.6: Collatz sequence

A Collatz sequence is formed as follows: We start with some number x_0, and we find the next number in the sequence by

$$x_{i+1} = \begin{cases} x_i/2 & \text{if } x_i \text{ is even} \\ 3x_i + 1 & \text{if } x_i \text{ is odd} \end{cases}$$

If $x_i = 1$, we stop iterating and have found the full sequence.

For example, if we start with $x_0 = 5$, we obtain the sequence:

```
5 16 8 4 2 1
```

It is conjectured, though not proven, that every chain eventually ends at 1.

Print the Collatz sequence starting at $x_0 = 103$.

3 Functions

Exercise 3.1: Hello

(a) Write a function `hello_world` that prints `'Hello, world!'`

(b) Write a function `hello_name(name)` that prints `'Hello, name!'` where `name` is a string.

(c) Explain the difference between the `print` and `return` keywords. What would change if instead of `print` you would use `return`?

Exercise 3.2: Polynomial

Write a function that evaluates the polynomial $3x^2 - x + 2$.

Exercise 3.3: Maximum

Write a function `my_max(x,y)` that returns the maximum of x and y. Do not use the `max` function, but use `if` instead in following two ways:

(a) Use both `if` and `else`.

(b) Use `if` but not `else` (nor `elif`).

Exercise 3.4: Primes

(a) Write a function `is_prime(n)` that returns `True` only if n is prime.

(b) Note that apart from 2 and 3, all primes are of the form $6k \pm 1$ (though not all numbers of the form $6k \pm 1$ are prime of course). Using this, we can improve the computation time by a factor 3. Update your function to use this.

(c) Write a function that returns all primes up to n.

(d) Write a function that returns the first n primes.

Exercise 3.5: Root finding

Suppose f is a continuous function and $f(a) < 0$ and $f(b) > 0$ for some known a and b. For simplicity, assume $a < b$. Then, there must exist some c such that $f(c) = 0$.

(a) Write a function `root(f, a, b)` that takes a function `f` and two floats `a` and `b` and returns the root `c`. Hint: check the sign at the midpoint of the interval.

(b) Remove the assumption that $a < b$, and that $f(a) < 0$ and $f(b) > 0$, if your current code relies on them.

(c) Add a check that prints
`'function evals have same sign'`
if $f(a) > 0$ and $f(b) > 0$ or if $f(a) < 0$ and $f(b) < 0$.

4 Lists

Exercise 4.1: Short questions

(a) Write a function that prints the elements of a list

(b) Write a function that prints the elements of a list in reverse

(c) Write your own implementation of the `len` function that returns the number of elements in a list.

Exercise 4.2: Copying lists

(a) Create a list `a` with some entries.

(b) Now set `b = a`

(c) Change `b[1]`

(d) What happened to `a`?

(e) Now set `c = a[:]`

(f) Change `c[2]`

(g) What happened to `a`?

Now create a function `set_first_elem_to_zero(l)` that takes a list, sets its first entry to zero, and returns the list.

What happens to the original list?

Exercise 4.3: Lists of lists

What is the difference between `a` and `b`:

```
a = [[]] * 3
```

```
b = [[] for _ in xrange(3)]
```

Exercise 4.4: Lists and functions

Write a function that takes a list and an index, and sets the value of the list at the given index to 0.

Exercise 4.5: Primes

In Section 3 you wrote a function that prints all primes up to n, and a function that prints the first n primes. Update these functions such that they return lists instead.

Exercise 4.6: List comprehensions

Let $i, j = 1, \ldots, n$

(a) Generate a list with elements `[i,j]`.

(b) Generate a list with elements `[i,j]` with $i < j$

(c) Generate a list with elements `i + j` with both i and j prime and $i > j$.

(d) Write a function that evaluates an arbitrary polynomial $a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$ using a list comprehension, where you are given `x` and a list with coefficients `coefs` (hint: use enumerate)

Exercise 4.7: Filter

In lecture we have seen how to implement `map` using list comprehensions. Implement `filter` using list comprehensions. Name your functions `myfilter` so you can compare with Python's standard filter.

Exercise 4.8: Flatten a list of lists

Consider having a list with lists as elements, e.g. `[[1,3], [3,6]]`.

Write a function that takes such a list, and returns a list with as elements the elements of the sublists, e.g. `[1, 3, 3, 6]`.

Exercise 4.9: Finding the longest word

Write a function that returns the longest word in a variable `text` that contains a sentence. While `text` may contain punctuation, these should not be taken into account. What happens with ties?

As an example, consider: "Hello, how was the football match earlier today???"

Exercise 4.10: Collatz sequence, part 2

Recall the Collatz sequence problem from Section 6. Our goal is to find the number $n < 1,000,000$ that leads to the longest Collatz sequence.

(a) Write a function that for any n, returns its Collatz sequence as a list

(b) Write a function that finds the integer x that leads to the longest Collatz sequence with $x < n$.

Exercise 4.11: Pivots

Write a function that takes a value `x` and a list `ys`, and returns a list that contains the value `x` and all elements of `ys` such that all values `y` in `ys` that are smaller than `x` come first, then we element `x` and then the rest of the values in `ys`

For example, the output of `f(3, [6, 4, 1, 7])` should be `[1, 3, 6, 4, 7]`

Exercise 4.12: Prime challenge

Write the function `primes(n)` that return a list with all prime numbers up to `n` using three (or less) lines of code.

Hint 1: Use lambda functions and list comprehensions.

Hint 2: Use the first two lines to define two helper (lambda) functions.