python™ from scratch

A primer for scientists working with Next-Generation-Sequencing data

Chapter 1

Text output and manipulation

Chapter 1: text output and manipulation

In this unit you will learn

- how to write text output to the shell
- storing values in variables
- the notion of a string
- different ways of manipulating and searching in strings
- basic data types supported by python

Text output

- `print()` function writes output to the console

function

Collection of one or more individual commands, identified by a name. Can return one or more values.

- argument to `print()` is a string

argument

Value passed to a function to be used within the function.
A function can have zero to many arguments.

- example:

```
print("Hello, python world!")
```

Data types: string

- a string is a list of characters
- different kinds of quotes possible:

```
'Single quotes define strings.'

"So do double quotes."

'''Even triple quotes can be used.
These can even contain multiple lines!'''
```

- You will work with strings A LOT! (e.g. DNA/Protein sequences, input/output of programs, ...)

Creating your first script

Typing commands into the command line interface will get annoying quickly, so let's create a script file:

1. open your text editor
2. type this line:

```
print("Hello, python world!")
```

3. save the file as `hello.py`

Running your first script

1. open up a terminal
2. navigate to the location of the file you just created
3. type

```
$ python3 hello.py
```

4. admire the output
5. go play! (modify your script, see what works and what doesn't)

Comments

- Comments are a very important means of documenting your work.
- See them as guides to “future you” or a colleague to understand your code.
- Everything after a `#` is a comment

```
print("Hello, python world!") # print message
# this line is a comment
# this: print(“Hello”) will not print
```

Printing special characters

- useful for formatting:
  - `'\n'`: newline character
  - `'\t'`: tabulator (TAB) character
- examples:

```
>>> print("Hello\npython world!")
Hello
python world!
>>> print("Hello\n\tpython world!")
Hello
   python world!
```

Variables

- a variable contains a value of a certain type (e.g. a string)
- variable names are arbitrary, everything is allowed except:
  - names that start with a number
  - reserved keywords (e.g. "`print`")
  - special characters (e.g. "€", "§", ...)
- variable names are case-sensitive ("dna" ≠ "Dna")
- **choose clear, self-documenting names!**
- examples:

```
x = "ATTagg"        # bad name
my_dna = "ATTagg" # better
seq_count = 3       # numeric value
```

String manipulation

- concatenation:

```
print("hello" + " " + "world")
```

- length:

```
len("hello world")
```

→ returns an `int` (integer number)

- changing case

```
my_dna = "ATTagg"
```

  - all upper case:

```
my_dna.upper()
```

  - all lower case:

```
my_dna.lower()
```

- replacement:

```
my_rna = my_dna.replace("T", "U")
```

Getting parts of a string

- strings support **indexing**:

```
"AUUGC"[1:3]
```

- syntax: string[<start>:<end>]
- positions are counted up from **0**, not 1!
- <start> is **in**clusive
- <end> is **ex**clusive

```
>>> "AUUGC"[1:3]
"UU"
```

- a single index gives one character

```
>>> "AUUGC"[0]
"A"
```

- This notation is very important!
  We will make extensive use of it.

Counting and searching

- Count occurrences of a substring:

```
>>> "TATATCGC".count("T")
3
>>> "TATATCGC".count("AT")
2
```

- You can also find the location of the first occurrence:

```
>>> "TATATCGC".find("TA")
0
>>> "TATATCGC".find("X")
-1
```

- For more useful string functions see: http://docs.python.org/3/library/stdtypes.html#string-methods

Data types

variables can store values of certain types:

type	explanation	example
`str`	strings are sequences of characters, textual data	`"ACGT"`
`int`	integral numbers	`42`
`float`	floating point number	`3.14`, `42.0`
`bool`	a logical value	`True`, `False`

You can convert a value/variable to a different type by using the type name as a function:

```
>>> str(42)
'42'
```

Recap (one)

So far we have seen

- the difference between functions, statements and arguments
- the importance of comments and how to use them
- how to store values in variables
- the way that data types work, and the importance of understanding them
- the difference between functions and methods, and how to use them

Recap (two)

We've encountered some tools that are specifically for working with strings:

- different types of **quotes** and how to use them
- special characters
- **concatenation**
- changing the **case** of a string
- **finding** and **counting** substrings
- **replacing** bits of a string
- **extracting** bits of a string to make a new string

Getting help

- python **interactive help** is right there at your fingertips:

```
>>> help()

Welcome to Python 3.4!  This is the interactive help utility.
[…]
```

```
>>> help(str)
```

- the **online tutorial** is also very helpful:

http://docs.python.org/3.4/tutorial/

Exercise 1-1: GC-content

You are given the following **DNA sequence**:

```
ACTGATCGATTACGTATAGTATTTGCTATCATACATATATATCGATGCGTTCAT
```

Write a program that will **print out the GC content** of this DNA sequence.

Hints:

- Use the `len` function and the `count` method
- you can use normal mathematical symbols like add (+), subtract (-), multiply (*), divide (/) and parentheses to carry out calculations on numbers in Python.

When using python 2 include this line at the top of your script!

```
from __future__ import division
```

Exercise 1-2: complementing DNA

You are given the following **DNA sequence**:

```
ACTGATCGATTACGTATAGTATTTGCTATCATACATATATATCGATGCGTTCAT
```

Write a program that will **print out the complement** of this DNA sequence.

Hint:

- the `replace` method is your friend here

Exercise 1-3: restriction fragment lengths

Here's a short DNA sequence:

```
ACTGATCGATTACGTATAGTAGAATTCTATCATACATATATATCGATGCGTTCAT
```

The sequence contains a recognition site for the *EcoRI* restriction enzyme, which cuts at the motif `G*AATTC` (the position of the cut is indicated by an asterisk).

Write a program which will calculate the size of the two fragments that will be produced when the DNA sequence is digested with *EcoRI*.

Exercise 1-4: Splicing out introns

Here's a short section of **genomic DNA**:

```
ATCGATCGATCGATCGACTGACTAGTCATAGCTATGCATGTAGCTACTCGATCG
ATCGATCGATCGATCGATCGATCGATCGATCATGCTATCATCGATCGATATCGA
TGCATCGACTACTAT
```

It comprises **two exons and an intron**. The first exon runs from the **start** of the sequence to character **63**, and the second exon runs from character **91** to the **end** of the sequence.

Write a program that will **print just the coding regions** of the DNA sequence.

Excercise 1-4: Splicing out introns (pt. 2)

(a) Using the data from part one, write a program that will calculate **what percentage of the DNA sequence is coding**.

(b) Using the data from part one, write a program that will **print out the original genomic DNA** sequence with **coding bases in uppercase** and **non-coding bases in lowercase**.