Introduction to Python: Basics

Mehmet Tevfik DORAK, MD PhD
Kingston University London
U.K.

Faculty of Science, Engineering and Computing

Kingston University London

Program

Day 2
Introduction to Python as a coding language for bioinformatics
What can you do with programming languages?
What is Python and why Python?
Python basics
Simple Python codes
Demonstrations of more advanced methods
Questions & answers

Kingston University London

An Informal Introduction to Python

3. An Informal Introduction to Python

In the following examples, input and output are distinguished by the presence or absence of prompts (`>>>` and `...`): to repeat the example, you must type everything after the prompt, when the prompt appears; lines that do not begin with a prompt are output from the interpreter. Note that a secondary prompt on a line by itself in an example means you must type a blank line; this is used to end a multi-line command.

Many of the examples in this manual, even those entered at the interactive prompt, include comments. Comments in Python start with the hash character, `#`, and extend to the end of the physical line. A comment may appear at the start of a line or following whitespace or code, but not within a string literal. A hash character within a string literal is just a hash character. Since comments are to clarify code and are not interpreted by Python, they may be omitted when typing in examples.

Some examples:

```
# this is the first comment
SPAM = 1                 # and this is the second comment
                         # ... and now a third!
STRING = "# This is not a comment."
```

3.1. Using Python as a Calculator

Let's try some simple Python commands. Start the interpreter and wait for the primary prompt, `>>>`. (It shouldn't take long.)

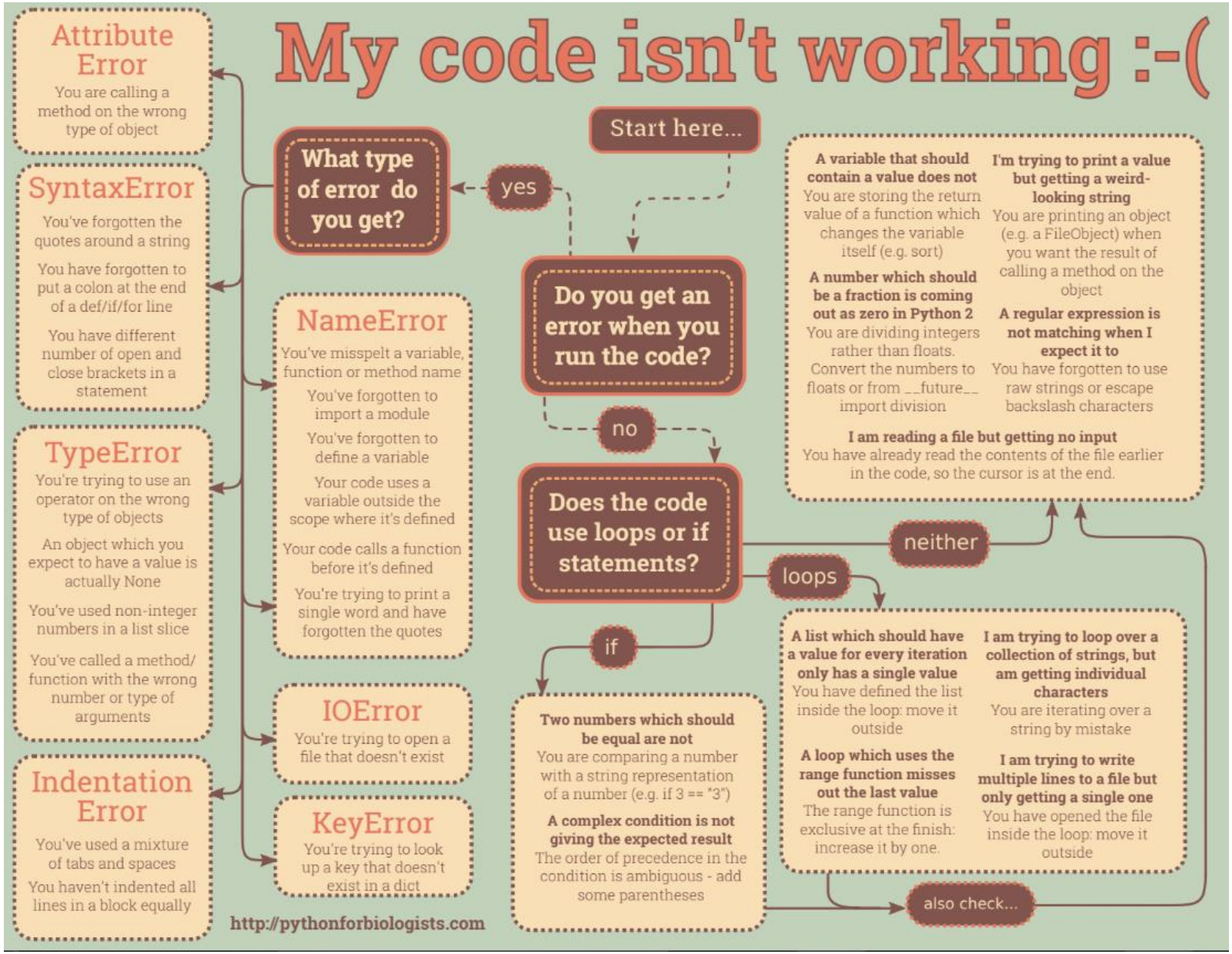
3.1.1. Numbers

The interpreter acts as a simple calculator: you can type an expression at it and it will write the value. Expression syntax is straightforward: the operators `+`, `-`, `*` and `/` work just like in most other languages (for example, Pascal or C); parentheses can be used for grouping. For example:

```
>>> 2+2
4
>>> # This is a comment
... 2+2
4
>>> 2+2  # and a comment on the same line as code
4
```

Kingston University London

29 Common Beginner Python Errors



Kingston University London

Python Data Types

NUMBERS
int, float, complex

SEQUENCES
string, list, tuple

SETS

MAPS
dictionaries

Sequence Data Types

Box 1. Most-Used Sequence Data Types

String: Usually enclosed by quotes (') or double quotes ("). Triple quotes ("") are used to delimit multiline strings. Strings are immutable. Once created they can't be modified. String methods are available at http://www.python.org/doc/2.5/lib/string-methods.html.
For example:
>>> s0='A regular string'

List: Defined as an ordered collection of objects; a versatile and useful data type. C programmers will find lists similar to vectors. Lists are created by enclosing their comma-separated items in square brackets, and can contain different objects.
For example:
>>> MyList=[3,99,12,"one","five"]
This statement creates a list with five elements (three numbers and two strings) and binds it to the name "MyList". Each element of the list can be referred to by an integer index enclosed between square brackets. The index starts from 0, therefore MyList[3] returns "one". All list operations are available at http://www.python.org/doc/2.5/lib/typesseq-mutable.html.

Tuple: Also an ordered collection of objects, but tuples, unlike lists, are immutable. They share most methods with lists, but only those that don't change the elements inside the tuple. Attempting to change a tuple raises an exception. Tuples are created by enclosing their comma-separated items between parentheses. Tuples are similar to Pascal records or C structs; they are small collections of related data that are operated on as a group. They are used mostly for encapsulating function arguments, or any data that are tightly coupled.
For example:
>>> MyTuple=(2,3,10)
Tuple operations are available at http://www.python.org/doc/2.5/lib/typesseq.html.

Kingston University London

OPEN ACCESS Freely available online
PLoS COMPUTATIONAL BIOLOGY
Education

A Primer on Python for Life Science Researchers

Sebastian Bassi

Sequence Data Types

Box 2. Unordered Types

Set: An unordered collection of immutable values. It is mostly used for membership testing and removing duplicates from a sequence. Sets are created by passing any sequential object to the set constructor, such as: set([1,2,3])
For more information on sets, please refer to http://www.python.org/doc/2.5/lib/types-set.html.
For example:

```
>>> ResEzSet1=set(['BamH1', 'HindIII', 'EcoR1', 'SalI'])
>>> ResEzSet2=set(['PlaA', 'EcoR1', 'Eco143'])
>>> ResEzSet1&ResEzSet2
set(['EcoR1'])
```

Dictionary:
A data type that stores unordered one-to-one relationships between keys and values. Unordered in this context means that each key–value pair is stored without any particular order in the dictionary. It is analogous to a hash in Perl or a Hashtable class in Java. Dictionaries are created by placing a comma-separated list of key–value pairs within braces.
For example:
Set Translate as a dictionary with codon triplets as keys and the corresponding amino acids as values:

```
>>> Translate={"cca":"P","cag":"Q","agg":"R"}
```

Creating a new entry:

```
>>> Translate["gat"]="D"
```

To see what is inside the dictionary:

```
>>> Translate
{'agg': 'R', 'cag': 'Q', 'gat': 'D', 'cca': 'P'}
```

Dictionaries share some methods with lists. A complete list of methods on can be seen at: http://www.python.org/doc/2.5/lib/typesmapping.html.

Kingston University London

OPEN ACCESS Freely available online
PLoS COMPUTATIONAL BIOLOGY
Education

A Primer on Python for Life Science Researchers

Sebastian Bassi

Python Arithmetic Operators

Addition: +

Subtraction: -

Multiplication: *

Exponent: **

Division: /

Floor division: //

Modulus: %
(returns remainder)

Numeric Data Types

Table 1. Numeric Data Types

Numeric Data Type	Description	Example
Integer	Holds integer numbers without any limit (besides your hardware). Some texts still refers to a "long integer" type since there were differences in usage for numbers higher than $2^{32} - 1$. These differences are now reserved for language internal use.	42, 0, −77
Float	Handles the floating point arithmetic as defined in ANSI/IEE Standard 754 [43], known as "double precision." Internal representation of float numbers is not exact, but accurate enough for most applications. The "decimal" module [44] has more precision at the expense of computing time.	424.323334, 0.00000009, 3.4e-49
Complex	Complex numbers are the sum of a real and an imaginary number and is represented with a "j" next to the real part. This real part can be an integer or a float number. j is the notation for imaginary number, equal to the square root of −1.	3 + 12j, j 9j
Boolean	True and False are defined as values of the Boolean type. Empty objects, None, and numeric zero are considered False. Objects are considered True.	False, True

doi:1011371/journal.pcbi.0030199.t001

Kingston University London

OPEN ACCESS Freely available online

PLoS COMPUTATIONAL BIOLOGY

Education

A Primer on Python for Life Science Researchers

Sebastian Bassi

Control Flow

Box 3. Control Structures

If statement: Tests for a condition and acts upon the result of that condition. If the condition is true, the block of code after the "if condition" will be executed. If it is false, the program will skip that block and will test for the next condition (if any). Several conditions can be tested using **elif**. If all conditions are false, the block under **else** will be executed. **Elif** can be used to emulate a C "switch-case" statement.
Scheme of an if statement:

```
if condition1:
   block1
elif condition2:
   block2
else:
   block3
```

For loop: Iterates over all the members of a sequence of values (as in Perl's "foreach"). It is different from C and VB because there isn't a variable that increments or decrements on each cycle. This sequence could be any type of iterable object like a list, string, tuple, or dictionary. The code inside a for loop will be executed once for each item in the sequence, and at the same time the variable will take the value of each item in the sequence. There could be an optional **else** clause. If it is present, the block under the else clause is executed when the loop terminates through exhaustion of the list, but not when the loop is terminated by a **break** statement.
The structure of a for loop is:

```
for variable in sequence:
   block1
else:
   block2
```

while loop: Executes a block of code as long as a condition is true. As the for loop, there could be an optional **else** clause. When present, the block under the else clause is executed when the condition becomes false but not when the loop is terminated by a **break** statement.
The general form is:

```
while condition:
   block1
else:
   block2
```

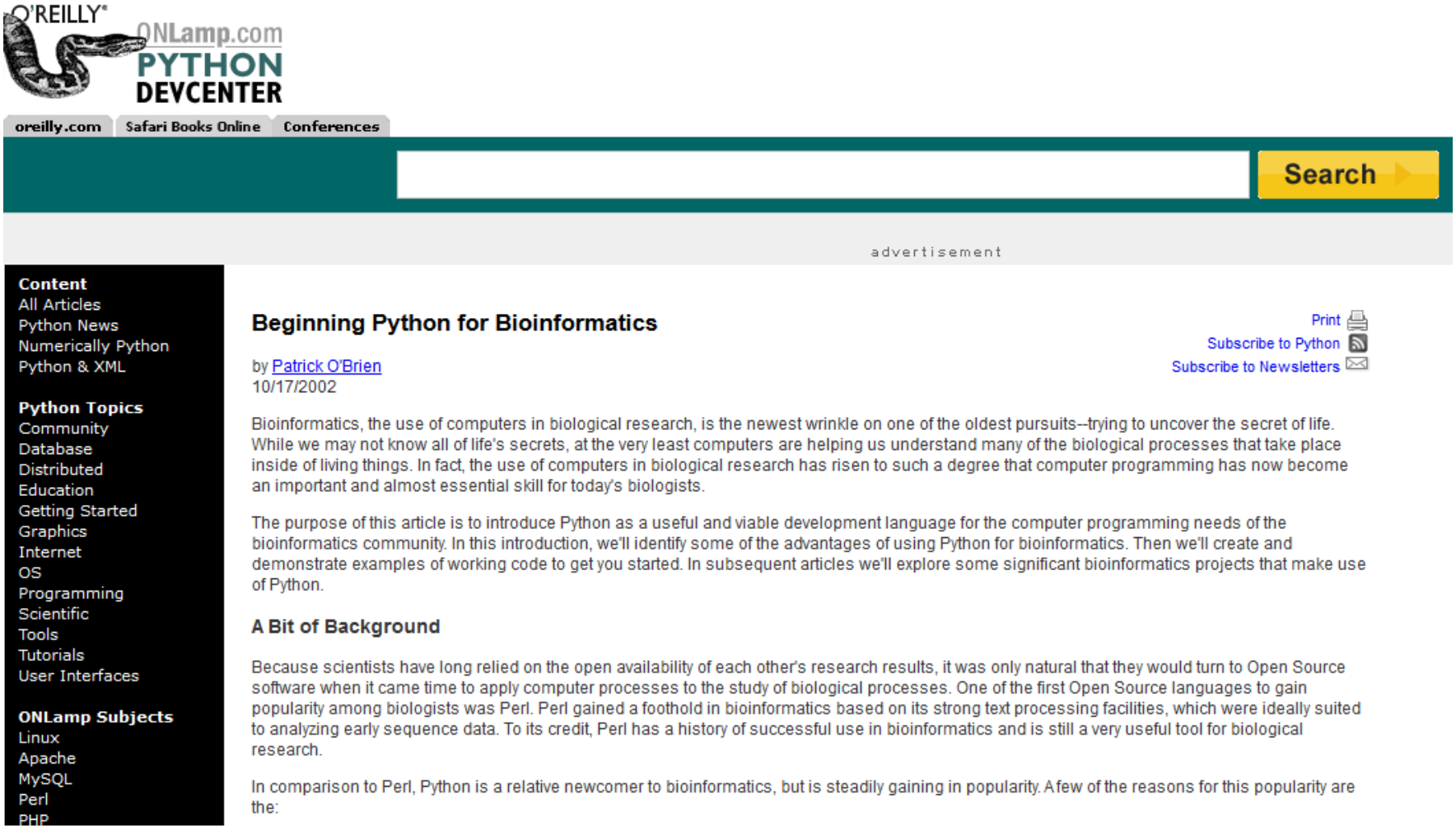
Kingston University London

OPEN ACCESS Freely available online
PLoS COMPUTATIONAL BIOLOGY
Education

A Primer on Python for Life Science Researchers

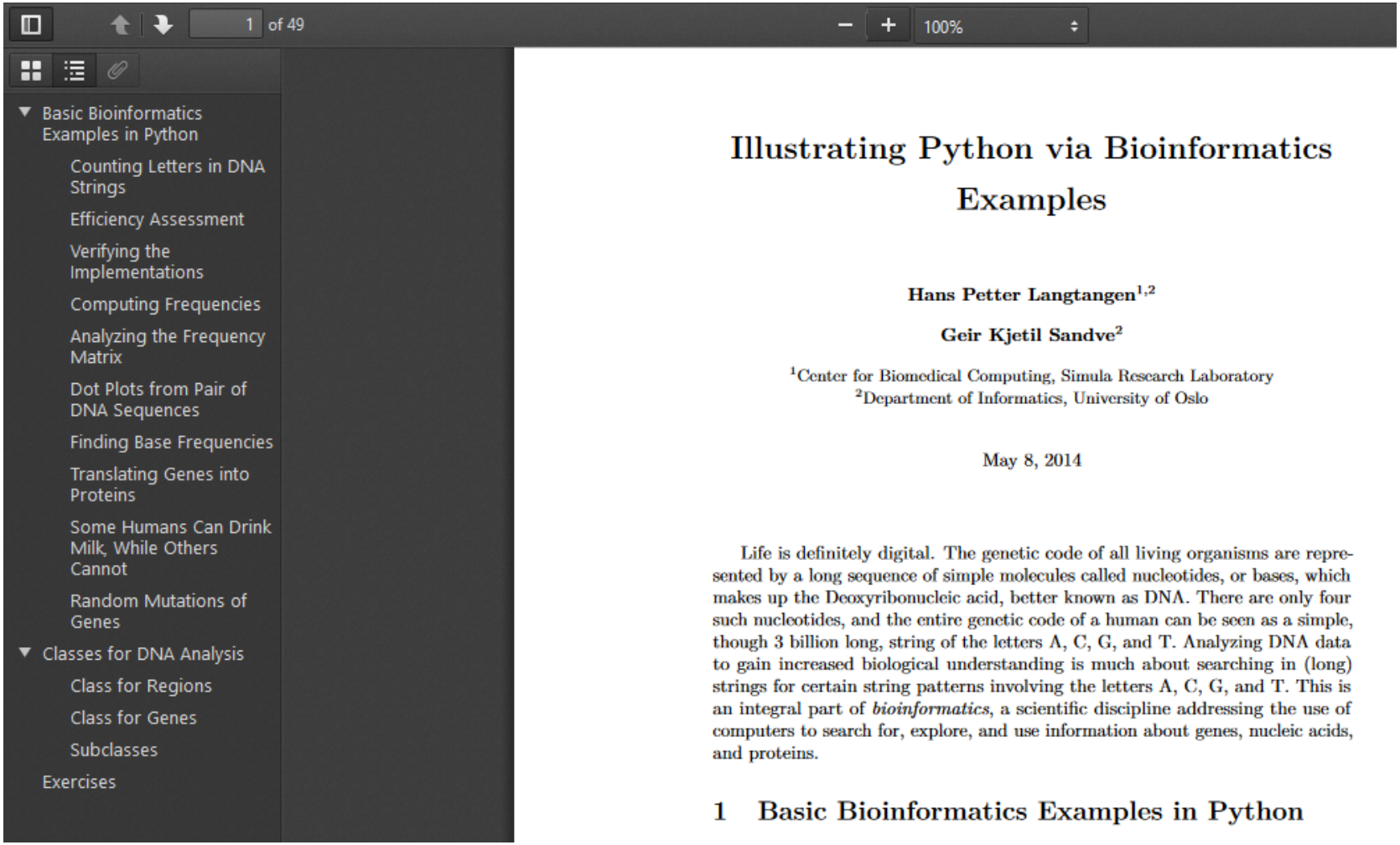
Sebastian Bassi

Python for Bioinformatics



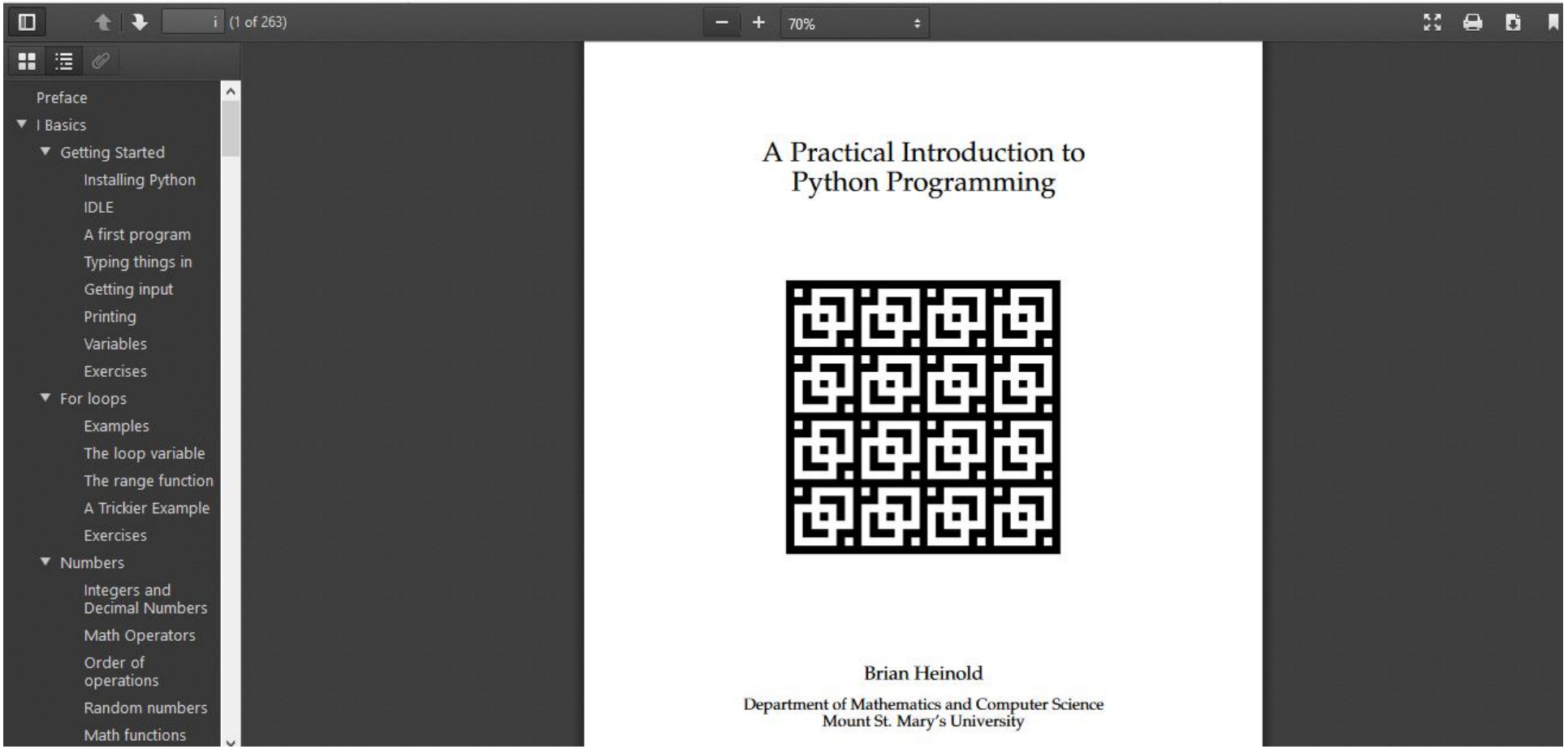
Kingston University London

Python for Bioinformatics



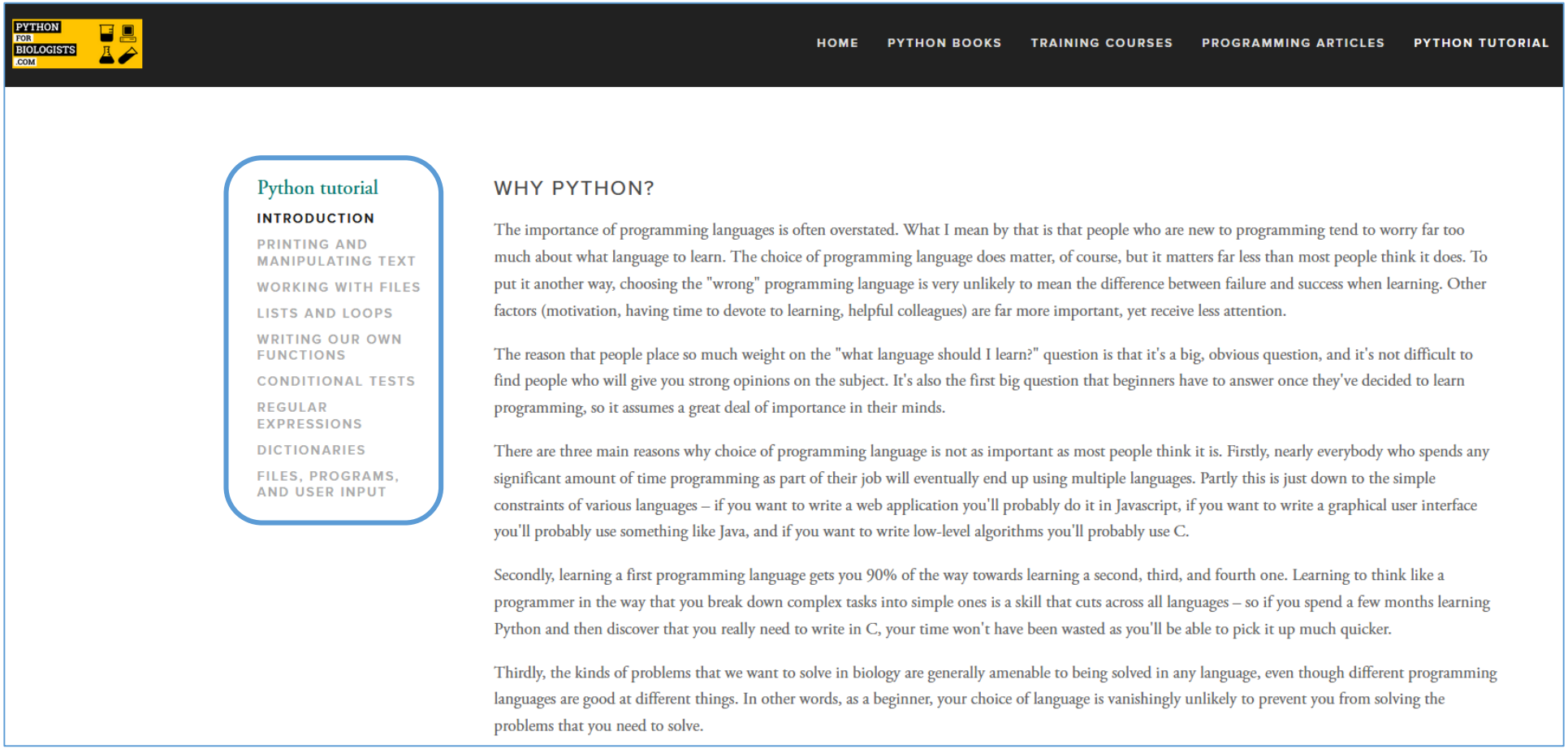
Kingston University London

Python for Bioinformatics



Kingston University London

Python for Biologists: Tutorial



Kingston University London

https://pythonforbiologists.com

… Looking forward

Day 2
Introduction to Python as a coding language for bioinformatics
What can you do with programming languages?
What is Python and why Python?
Python basics
Simple Python codes
Demonstrations of more advanced methods
Questions & answers

Kingston University London

Kingston University London