Overview

day one

0. introduction
1. text output and manipulation
2. reading and writing files

day two

3. lists and loops
4. writing functions

today

5. conditional statements
6. dictionaries

day four

7. files, programs and user input
8. biopython

day five

hands on training

feedback and discussion

This course (apart from chapter 8) is based on the book "**Python for Biologists**":

http://pythonforbiologists.com/

Today

- Pad for today: http://python-from-scratch.pad.spline.de/3?
- Longer lunch break: 11:30 – 13:00

python™ from scratch

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

CHAPTER 5

Conditional tests

Why conditions?

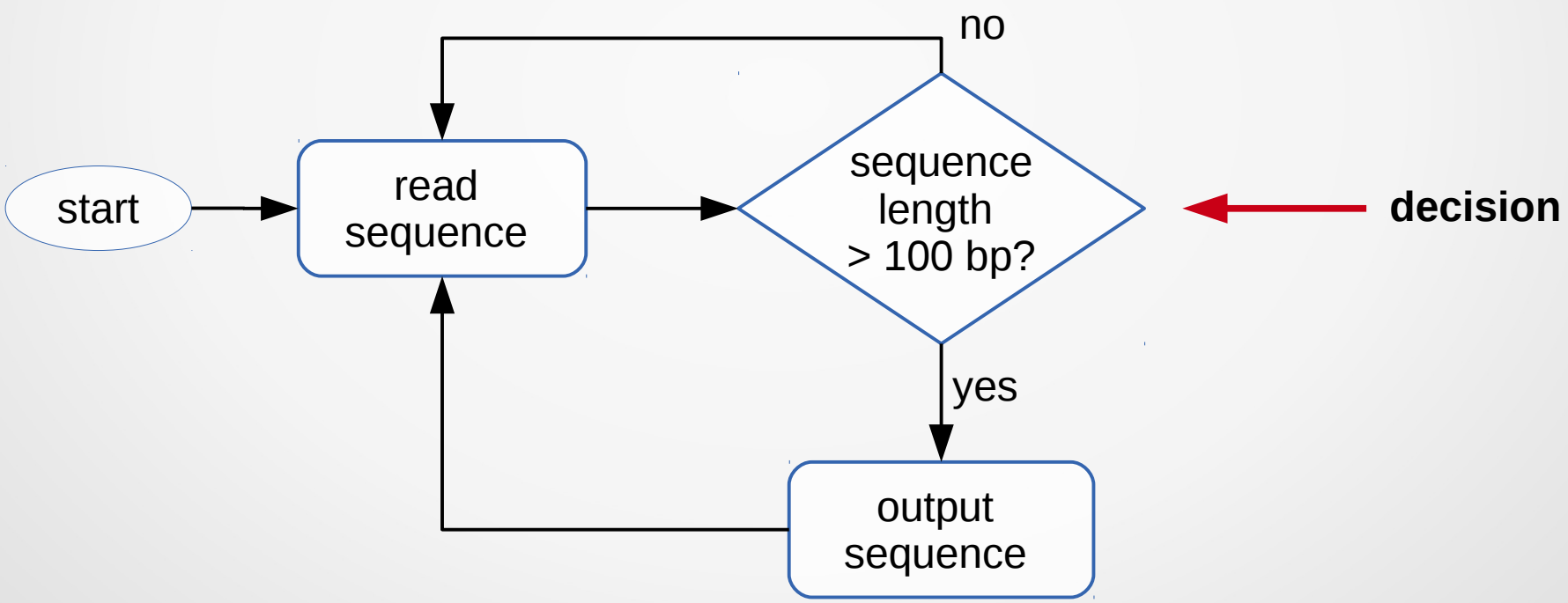
Decisions need to be made based on the given situation.

The more different cases your code is able to handle, the more flexible and useful it will be.

Checking for conditions increases the fault tolerance of your programs.

Example: Filtering DNA sequences

"Filter a number of DNA sequences such that only sequences ***longer than 100 bp*** *are retained."*



Conditions as logical statements

making decisions requires evaluating the truth of (logical) statements

logical statements can only have two values:

- True
- False

examples:

```
len(sequence) > 100
len(seq1) == len(seq2)
get_gc_content(seq) <= 0.5
5 < 8
```

Operators

The following operators can be used in simple comparisons:

operator	explanation	example
`<`	less than	`3 < 5`
`<=`	less than or equal	`3 <= 5`
`>`	greater than	`8 > 5`
`>=`	greater than or equal	`8 >= 5`
`==`	equal	`len('AAA') == 3`
`!=`	not equal	`"AAA" != "aaa"`
`in`	test if element is in a list	`3 in [1,2,3]`
`not`	inverts the value of the following test	`not 5 in [1,2,3]`

More conditional testing

There are quite a few more ways to test for properties of values (especially strings).

These tests are mostly performed by string methods:

operator	explanation	example
`startswith`	check prefix of a string	`"ACGT".startswith("A")`
`endswith`	check suffix of a string	`"ACGT".endswith("T")`
`islower`	Is given string in lower case?	`"acgt".islower()`
`isupper`	Is given string in upper case?	`"ACGT".isupper()`

Conditional statements: if

condition

```
if len(dna) >= 100:
  print(dna)
```

body: this code block is executed only if the condition is met (i.e. equals True)

If-statement example

- Here is a more useful example:

```
accs = ['ab56', 'bh84', 'hv76', 'ay93', 'ap97', 'bd72']
for accession in accs:
   if accession.startswith('a'):
      print(accession)
```

- Note: Indentations are **nested** whenever a new code block is introduced!

If-else-statement

- What if we need to do something when the condition is not met?

```
file1 = open("one.txt", "w")
file2 = open("two.txt", "w")
accs = ['ab56', 'bh84', 'hv76', 'ay93', 'ap97', 'bd72']
for accession in accs:
   if accession.startswith('a'):
      file1.write(accession + "\n")
   else:
      file2.write(accession + "\n")
```

- implements an "either or" situation

elif statements

- Handling more than two cases can results in bulky if-statements:

```
if exp_lvl <= 100:
  print("gene is lowly expressed")
else:
   if exp_lvl > 100 and exp_lvl <= 150:
      print("gene is normally expressed")
   else:
      print("gene is highly expressed")
```

- The elif clause makes your code more readable:

```
if exp_lvl <= 100:
   print("gene is lowly expressed")
elif exp_lvl > 100 and exp_lvl <= 150:
   print("gene is normally expressed")
else:
   print("gene is highly expressed")
```

Conditional loops: while

remember the `for` loop:

```
for element in some_list:
   # do something
```

conditions can be used in `while` loops:

```
samfile = open("mapping.sam")
line = samfile.readline()
while line.startswith('@'):
   line = samfile.readline()
```

The while loop iterates (i.e. runs) as long as the **condition is true**.

Complex conditions

- Simple conditional tests can be combined using "**and**", "**or**" and "**not**":

```
accs = ['ab56', 'bh84', 'hv76', 'ay93', 'ap97', 'bd72']

for accession in accs:
    if accession.startswith('a') and accession.endswith('3'):
        print(accession)

for accession in accs:
    if accession.startswith('a') or accession.startswith('b'):
        print(accession)

for accession in accs:
    if accession.startswith('a') and not accession.endswith('6'):
        print(accession)
```

true/false functions

- boolean functions are no different than other functions:

```
def is_at_rich(dna):
   length = len(dna)
   a_count = dna.upper().count('A')
   t_count = dna.upper().count('T')
   at_content = (a_count + t_count) / length
   if at_content > 0.65:
      return True
   else:
      return False
```

Note: This function will not work correctly with *Python 2* unless you include the following import at the top.

```
from __future__ import division
```

true/false functions

- Now you can use the function in conditional statements...

```
if is_at_rich(my_dna):
   # do something with the sequence
```

- ...or for testing purposes

```
# test function "is_at_rich()"
assert is_at_rich("ATTATCTACTA")
assert is_at_rich("attatctacta")
assert not is_at_rich("CGGCAGCGCT")
```

Recap

In this unit you learned about:

- conditions
- **using** conditions in conditional statements
- **combining** conditions into more complex statements using boolean operators
- handling and returning boolean values
- writing, testing and using boolean functions

Exercise 5-1: filtering data

- In file `data.csv` you find the following input:

```
D. melanogaster,atatata[...],kdy647,264
D. melanogaster,actgtga[...],jdg766,185
D. simulans,atcgat[...],kdy533,485
[...]
```

- The data is structured in 4 fields (separated by **commas**):

1. species name
2. DNA sequence
3. gene name
4. expression level

Exercise 5-1: filtering data

- Filter the input data by the following criteria:

  a) **Species name**: Print out the gene names for all genes belonging to *D. melanogaster* or *D. simulans*

  b) **Length range**: Print out the gene names for all genes between 90 and 110 bp long

  c) **GC content**: Print out the gene names for all genes whose **GC** content is less than 0.5 and whose expression level is greater than 200.

  d) **Gene name**: Print out the gene names for all genes whose name begins with "k" or "h" except those belonging to *D. melanogaster*.

Exercise 5-2: AT content

- Using the same input data as in ex. 5-1, print for each gene, whether its AT content is
  - **high**: greater than **0.65**
  - **low**: less than **0.45**
  - **medium**: between **0.45** and **0.65**