Python for scientists
Lesson 5
‘for’ loops and functions

def complementary(seq):
    nt_comp = {
        'A': 'T',
        'T': 'A',
        'C': 'G',
        'G': 'C',
    }
    compseq = []
    for nt in seq:
        compseq += nt_comp[nt]
    return ''.join(compseq)

Alvaro Sebastián Yagüé

www.sixthresearcher.com

@SixthResearcher
‘for’ loops
A **for loop** iterates over the items of any list or string, in the order that they appear.

```python
>>> sequences = ['ACGTCCGAT', 'TGCCATTT', 'AGGCTTCAGAT', 'ATT']
>>> for seq in sequences:
    print(seq, len(seq))

ACGTCCGAT 9
TGCCATTT 8
AGGCTTCAGAT 11
ATT 3
```

```python
>>> for seq in sequences:
    if (len(seq)<5):
        print(seq, "is short")

ATT is short
```
‘range’ function

Range is used to iterate over a sequence of numbers.

```python
>>> for x in [0,1,2,3,4,5,6,7,8,9]:
    print (x,end=" ")
0 1 2 3 4 5 6 7 8 9
g>>> for x in range(10):
    print(x, end=" ")
0 1 2 3 4 5 6 7 8 9
g>>> for x in range(1,10):
    print(x, end=" ")
1 2 3 4 5 6 7 8 9
g>>> for x in range(1,10,3):
    print(x, end=" ")
1 4 7
```

end=" " indicates print to finish with a blank space instead of a new line.

```python
animals = ['human', 'monkey', 'cat', 'dog']

>>> for i in range(len(animals)):
    print(i, animals[i])
0 human
1 monkey
2 cat
3 dog
```
‘for’ vs. ‘while’

A ‘for’ statement can be replaced for a ‘while’.

But ‘for’ will be easier and shorter in most of the cases.

<table>
<thead>
<tr>
<th>for</th>
<th>while</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt;&gt; sequences = ['ACGTT', 'TGCCTTT', 'AGGCTT']</td>
<td>&gt;&gt;&gt; sequences = ['ACGTT', 'TGCCTTT', 'AGGCTT']</td>
</tr>
<tr>
<td>&gt;&gt;&gt; for seq in sequences:</td>
<td>&gt;&gt;&gt; while sequences:</td>
</tr>
<tr>
<td>print(seq, len(seq))</td>
<td>seq = sequences.pop()</td>
</tr>
<tr>
<td>ACGTT 5</td>
<td>print(seq, len(seq))</td>
</tr>
<tr>
<td>TGCCTTT 7</td>
<td>ACGTT 5</td>
</tr>
<tr>
<td>AGGCTT 6</td>
<td>TGCCTTT 7</td>
</tr>
<tr>
<td></td>
<td>AGGCTT 6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>for</th>
<th>while</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;&gt;&gt; for x in range(1,11):</td>
<td>&gt;&gt;&gt; count = 1</td>
</tr>
<tr>
<td>print(x, end=&quot; &quot;)</td>
<td>&gt;&gt;&gt; while count &lt;=10:</td>
</tr>
<tr>
<td>1 2 3 4 5 6 7 8 9 10</td>
<td>print(count, end=&quot; &quot;)</td>
</tr>
<tr>
<td></td>
<td>count = count + 1</td>
</tr>
<tr>
<td></td>
<td>1 2 3 4 5 6 7 8 9 10</td>
</tr>
</tbody>
</table>
‘for’ loops and matrices

Let's go through a matrix and exit when it finds a number higher than 100:

<table>
<thead>
<tr>
<th></th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
<th>C4</th>
</tr>
</thead>
<tbody>
<tr>
<td>L0</td>
<td>65</td>
<td>23</td>
<td>12</td>
<td>54</td>
<td>90</td>
</tr>
<tr>
<td>L1</td>
<td>3</td>
<td>42</td>
<td>67</td>
<td>14</td>
<td>32</td>
</tr>
<tr>
<td>L2</td>
<td>39</td>
<td>10</td>
<td>92</td>
<td>78</td>
<td>5</td>
</tr>
<tr>
<td>L3</td>
<td>75</td>
<td>37</td>
<td>107</td>
<td>24</td>
<td>48</td>
</tr>
<tr>
<td>L4</td>
<td>28</td>
<td>84</td>
<td>18</td>
<td>73</td>
<td>6</td>
</tr>
</tbody>
</table>

```python
>>> L0 = [65, 23, 12, 54, 90]
>>> L1 = [ 3, 42, 67, 14, 32]
>>> L2 = [39, 10, 92, 78,  5]
>>> L3 = [75, 37,107, 24, 48]
>>> L4 = [28, 84, 18, 73,  6]
>>> matrix = [L0, L1, L2, L3]
>>> for row in range(len(matrix)):
...     for col in range(len(matrix[0])):
...         if matrix[row][col] > 100:
...             print("High number found in row",row,"and column",col)
...             break

High number found in row 3 and column 2
```
Loop control statements

With control statements we can skip loop iterations or directly go out from a loop.

```python
>>> for x in range(1,10):
    if x != 5:  # Prints the number if is not 5
        print(x, end=" ")
    break
1 2 3 4 6 7 8 9

>>> for x in range(1,10):
    if x == 5:  # If the number is 5
        pass  # does nothing
    else:  # If not prints the number
        print(x, end=" ")
1 2 3 4 6 7 8 9

>>> for x in range(1,10):
    if x == 5:  # If the number is 5
        continue  # goes to next iteration (6)
        print(x, end=" ")
1 2 3 4 6 7 8 9

>>> for x in range(1,10):
    if x == 5:  # If the number is 5
        break  # finishes the loop
        print(x, end=" ")
1 2 3 4
```

- **break**  Finishes the loop execution
- **continue**  Jumps to next iteration
- **pass**  Does nothing

'break' will go out from the loop
Loops and dictionaries

Combining ‘for’ loops and dictionary methods we can go through dictionaries in an easy way.

```python
>>> temperatures = {'Jan': 5, 'Feb': 9, 'Mar': 12, 'Apr': 15, 'May': 20}
>>> for month in temperatures.keys():  # Let's print the registered months
    print(month)
May
Feb
Apr
Jan
Mar
>>> for temperature in temperatures.values():  # Let's print the registered temperatures
    print(temperature)
20
9
15
5
12
>>> for month,temperature in temperatures.items():  # Let's print months and temperatures
    print(month, temperature)
May 20
Feb 9
Apr 15
Jan 5
Mar 12
```
Combining ‘for’ loops and dictionary methods we can go through dictionaries in an easy way.

```python
>>> temperatures_2013 = {'Jan': 5, 'Feb': 9, 'Mar': 11, 'Apr': 14, 'May': 20}
>>> temperatures_2014 = {'Jan': 6, 'Feb': 12, 'Mar': 11, 'Apr': 12, 'May': 19}
>>> for month in temperatures_2013.keys():
    if (temperatures_2014[month] - temperatures_2013[month]) >= 2:
        print("In", month, "2014 temperature was unusually high")
    elif (temperatures_2013[month] - temperatures_2014[month]) >= 2:
        print("In", month, "2014 temperature was unusually low")

In Feb 2014 temperature was unusually high
In Apr 2014 temperature was unusually low

>>> formula1 = {'Hamilton': 'Mercedes', 'Vettel': 'Ferrari', 'Alonso': 'McLaren',
               'Rosberg': 'Mercedes', 'Raikkonen': 'Ferrari', 'Button': 'McLaren'}
>>> teams=dict()
>>> for driver1, team1 in formula1.items():
    for driver2, team2 in formula1.items():
        if driver1==driver2:
            pass
        elif (team1 or team2) in teams.keys():
            pass
        elif team1==team2:
            print(driver1,"and",driver2, "are in team", team1)
            teams[team1]=[driver1, driver2]
```

Rosberg and Hamilton are in team Mercedes
Alonso and Button are in team McLaren
Raikkonen and Vettel are in team Ferrari
Functions
Functions: definition and examples

A function is a block of code that requires some data or variables as arguments and gives as output some data or variables as results.

The importance of functions is that we can write once and use many times.

```python
>>> def addition(value1, value2):
    result = value1 + value2
    return result

>>> addition(1,2)
3

>>> def welcome(name):
    sentence = "Welcome "+name
    return sentence

>>> print(welcome("Tom"))
Welcome Tom
```

```python
def function(<data>):
    <code>
    return <data>
```
Exercise: calculating the factorial of a number
Calculating the factorial of a number

In previous lesson we were asked to write a ‘while’ loop to calculate the factorial of a number, now let’s write a ‘for’ loop to do the same task:

Factorial

In mathematics, the factorial of a non-negative integer \( n \), denoted by \( n! \), is the product of all positive integers less than or equal to \( n \). For example,

\[
5! = 5 \times 4 \times 3 \times 2 \times 1 = 120.
\]

```python
>>> number = 5
>>> factorial = 1
>>> for x in range(number, 1, -1):
    factorial = factorial * x
    # To print x and factorial variables in each iteration:
    print(x, factorial)

5 5
4 20
3 60
2 120
```

```text
5 5
4 20
3 60
2 120
```

```python
>>> factorial
120
```
Calculating the factorial of a number

Finally, let's use the previous code to create a function that takes as input a number and gives as result its factorial.

```python
>>> def factorial(number):
    result = 1
    for x in range(number, 1, -1):
        result *= x
    # Uncomment to print result in each iteration:
    # print(x, result)
    return result

>>> factorial(5)
120
```
Exercise: counting nucleotides in a DNA sequence
Counting nucleotides in a DNA sequence

Let’s write a ‘for’ loop that reads letter by letter a DNA sequence and counts the number of nucleotides of each type: adenine (A), cytosine (C), guanine (G) and thymine (T).

```python
>>> dna = "AGCCCTCCAGGACAGGCTGCATCAGAAGAGGCCATCAAGCGATCACGTCTTTCTTGGCATGACCTGTCGGGCTCACCTGTCGGGCTCACTGCCAAGAGGAGAGACCTGCGATTGGAACATGTGCTACAGCCTTCCTCTCCATGGGACTACTGCACTAGACAGCAGCCGCAGCCAGGGGTGTGCATTTGCTGTAAGCGCACTATGGGAAAGCCTTTGAGAATGAA"
>>> nucleotides = { 'A':0, 'C':0, 'G':0, 'T':0 }
>>> for i in range(len(dna)):
    if dna[i] == 'A':
        nucleotides['A']+=1 # Increments 1
    elif dna[i] == 'C':
        nucleotides['C']+=1
    elif dna[i] == 'G':
        nucleotides['G']+=1
    elif dna[i] == 'T':
        nucleotides['T']+=1

>>> nucleotides
{'C': 156, 'T': 77, 'A': 95, 'G': 141}
```

Counting nucleotides in a DNA sequence

We can simplify the previous example:

```python
>>> dna = "AGCCCTCCAGGACAGGCTGCACTCAAGAGAGGCCATCAGCAGATCACGTCTCTCTGCCCATTGGCCCTGTGGATCGGCCTCTGCCTCCCTGCACACCTGGTGGAGCTCCGAGGTGGCCAGGTTGGGCATTGTGGACAATGCTGTACCGAGCATCCTGCTCTCCCTCTCTACCAGCTGGAGAACCTACTGCAACTGAGACGCGCCCCGGCAGCCAGCCCACACCCGGGCCCTCCTGCCACCGAGAGAGATGGAATAAAGCCCTTGAACCAGCAAAAA"
>>> nucleotides = { 'A':0, 'C':0, 'G':0, 'T':0 }
>>> for i in range(len(dna)):
...     nucleotides[dna[i]]+=1

>>> nucleotides
{'C': 156, 'T': 77, 'A': 95, 'G': 141}
```

You can copy the insulin cDNA sequence from here:
Functions: counting nucleotides in a DNA sequence

Let's use the previous code to create a function that reads letter by letter a DNA sequence and counts the number of nucleotides of each type: adenine (A), cytosine (C), guanine (G) and thymine (T).

```python
>>> def count_nrs(dna):
    nucleotides = { 'A': 0, 'C': 0, 'G': 0, 'T': 0 }
    for i in range(len(dna)):
        nucleotides[dna[i]] += 1
    return nucleotides

>>> dna_insulin = "AGCCTCCAGGACAGGCTGCATCAAGAGCCCATCAAGCATCAGTGCTCCTTCTGC" +
                "CATGCGCCCTCTGCTGGGCAGGACCTGAAAAACCCAGCCGACCCCTTTGTGA" +
                "CCACACTTGCCGCTC" +
                "ACACACCCAGGACCGCTGGCAAGCTCCTTCTGGGCCAGGACCCCGTGAGACCTG" +
                "ACTGCAACTAGACGCGAGGCAGGCAGGCAGGCCCCACACCCGCGGCTCCTGCA" +
                "CCGAGAGAGATGGAATAAAGCCTTGTGAACAGCAAA"

>>> count_nrs(dna_insulin)
{'T': 77, 'G': 141, 'A': 95, 'C': 156}
```

Exercise: RNA translator
Exercise for biologists: RNA translator

Let’s write a function to translate RNA sequences into proteins:

- Example input:

  ```
  rna_insulin="AUGGCCCUGUGGAAUGCGCCUCCUGCCCUCUGCCUCUGGCGCUGCUUGGCCCUCUGUGCACUGACGGGCAGCCGC
  AGCCUUGUGAAACCAAACACCCUGUGCGGUCACACCUUGUGGAGACUCUCUACCUGUCUGGCGGGAACGAGGCUUC
  UUCUACACCCAAAGACCGCCGCGAGGACAGAGGACCUCAGAGGUCGGGACCGUGGCAGGCGCAGGCGGCAGGCCUUGGUG
  CAGGAGAGCCUGCAGCCUGCCUCUGGAGGUGCCUUGAGCAAGCGAGUGGAACAAUGCUUGAACCAGCAUCCUGUCCACCAT
  UGCUCUCUCACCAGCUGAGACACACUCUAGACUCAGCCCGAGCAGGCCGCCACCCACCCACCCCGCCCGCCUCUCUGCAGCAG
  AGAGAGAUGGAAUAAAGGCCCUUGAACCAGC"
  ```

- Example output:

  ```
  prot_insulin="MALWMRLLPLLALLALVGPDPAAAFVNQHLCGSHLVEALYLVCEGFGTYPKRREAEDLQVGQVELGGGPG
  AGSLQPLALEGSLQKRGIEQCCCTISCQVLQNYCN*TQPAGSSPPAASCTERDGIKPLNQ"
  ```
Exercise for biologists: RNA translator

Little help:

genetic_code = {
    'AAA': 'K', 'AAG': 'K', # Lysine
    'AAC': 'N', 'AAU': 'N', # Asparagine
    'ACA': 'U', 'ACC': 'U', 'ACG': 'U', 'ACU': 'U', # Threonine
    'AGA': 'R', 'AGG': 'R', # Arginine
    'AGC': 'S', 'AGU': 'S', # Serine
    'AUA': 'I', 'AUC': 'I', 'AUU': 'I', # Isoleucine
    'AUG': 'M', # Methionine
    'CAA': 'Q', 'CAG': 'Q', # Glutamine
    'CAC': 'H', 'CAU': 'H', # Histidine
    'CCA': 'P', 'CCC': 'P', 'CCG': 'P', 'CCU': 'P', # Proline
    'CGA': 'R', 'CGC': 'R', 'CGG': 'R', 'CGU': 'R', # Arginine
    'CUA': 'L', 'CUC': 'L', 'CUG': 'L', 'CUU': 'L', # Leucine
    'GAA': 'E', 'GAG': 'E', # Glutamic Acid
    'GAC': 'D', 'GAU': 'D', # Aspartic Acid
    'GCA': 'A', 'GCC': 'A', 'CCG': 'A', 'GCU': 'A', # Alanine
    'GGA': 'G', 'GGC': 'G', 'GGG': 'G', 'GGU': 'G', # Glycine
    'GUA': 'V', 'GUC': 'V', 'GUG': 'V', 'GUU': 'V', # Valine
    'UAA': '*', 'UAG': '*', # STOP codon
    'UAC': 'Y', 'UAU': 'Y', # Tyrosine
    'UCA': 'S', 'UCC': 'S', 'UCG': 'S', 'UCU': 'S', # Serine
    'UGA': '*', # STOP codon
    'UGC': 'C', 'UGU': 'C', # Cysteine
    'UGG': 'W', # Tryptophan
    'UUA': 'L', 'UUG': 'L', # Leucine
    'UUC': 'F', 'UUU': 'F', # Phenylalanine
    'UAA': '*', 'UAG': '*', # STOP codon
}
Python for scientists

Next lesson...
Built-in functions.
Reading and writing files

def complementary(seq):
    nt_comp = {
        'A': 'T',
        'C': 'G',
        'G': 'C',
        'T': 'A',
    }
    compseq = ''
    for nt in seq:
        compseq += nt_comp[nt]
    return compseq