1. String methods

Like any programming language, Python allows many operations on strings. Finding sub-strings, splitting, joining, etc. You can find a list of the available methods here.

Exercise
Use the appropriate methods to make the following lines of code work.

```
string = "In computer programming,  a string is traditionally  a sequence of
characters.  "

print(string.index('c')         )                    # index of the first 'c'
print(string.rindex('c')          )                   # index of the last 'c'
(see r{name} methods for right-hand functions)
print(len(string.rstrip()       ))                 # length of the string without
trailing whitespaces
print(string.startswith('In')                )              # whether the string
starts with "In"
print(string.lower()       )                       # string as all lower-case
print(string.split(',')         )                    # list of parts of the
sentence, split by ","
print(string.replace('  ',' ')                   )            # all double
whitespaces replaced by a single whitespace
print(string.replace('traditionally','')                              ) # without
the word "traditionally" (beware of whitespaces)
```

```
3
72
78
True
in computer programming,  a string is traditionally  a sequence of characters.
['In computer programming', '  a string is traditionally  a sequence of
characters.  ']
In computer programming, a string is traditionally a sequence of characters.
In computer programming,  a string is   a sequence of characters.
```

2. String formatting

Formatting a string allows you to export or print data. For example, printing the string `Client name: %s` where `%s` is formatted to be the name of a client given as a string. Besides substituting strings at `%s`, other data types can also be formatted in to the string. See here for a list of all formatting conversions. This includes formatting/rounding numbers.

A general way to format a string is given below. Note the `%d` for an integer. In case of a single argument, the `()` are not nessecary.

```
client_name = "Obelix"
client_age = 32                                                    # [years]
string = "Client %s is %d years old." % (client_name, client_age) # the format is:
string % (arguments)
print(string)
```

```
Client Obelix is 32 years old.
```

Exercise

Use the appropriate format to make the following lines of code work.

```
value = 1.73456
print("%.0f  " % value)       # 2        (see "5. Precision", why can't you use
%d?)
print("%.1f   " % value)      # 1.7
print("%.2f   " % value)      # 1.73
print("%7.2f   " % value)     #    1.73  (with a total length of 7, see "4.
Minimum field width")
print("%07.2f     " % value)  # 0001.73  (see Flag '0')
print("%+.2f    " % value)    # +1.73    (see Flag '+')
print("%+07.2f      " % value)# +001.73
print("%.2e    " % value)     # 1.73e+00 (exponential format)
```

```
2
1.7
1.73
   1.73
0001.73
+1.73
+001.73
1.73e+00
```

3. Regular expressions

Regular expressions are used to find patterns in text, without exactly specifying each character. For example to find words, to find numbers that were formatted in a particular way, etc.

A single digit can for example be matched with `\d`. That would match at 4 locations in the string `The width of the car is 2m, and the height is 1.65m.`.

Another example is that we can match a set of characters. This can be matched using `[xyz]`. That would

match at 4 locations in the string `If x = 2y, than y = 6z.`.
At Python Regular Expressions more information can be found on matching string patterns in Python. Using this information, make the following assignment.

Exercise
Open regex101.com.
On the left-hand side, select the "Python" flavor.
Copy the text below in the "TEST STRING" box.
In the "REGULAR EXPRESSION" text box, write a pattern that:

- Matches the first 10 lines with a decimal number.
- Does not match the integer in the 11th line.
- Does not match the text in the 12th line.

Tip: Start with simple cases. For example, first make it work for either "." or ",", and without leading zeros. Then add these one by one.

```
0001,2345
1,2345
1,23
,2345
1,
001.2345
1.2345
1.23
.2345
1.
1
thisisnotanumber
```

```
regexp = .*[,.].*
```

4. Counting characters

Exercise
Print all non-zero frequencies of each character from the alphabet in the text given in the code box.

- Treat accented characters as normal characters.
- Combine uppercase and lowercase characters in a single count.
- Print in alphabetical order.

Hint: Have one step where you prepare and filter some data, and a second step with a loop.
Hint: sets have unique values, and lists are indexed and can thus be sorted (sort()).

```
text = "For the movie The Theory of Everything (2014), Jóhann Jóhannsson composed
the song A Model of the Universe"
import unicodedata
import collections
text = unicodedata.normalize('NFD',text).encode('ascii','ignore').decode("utf-8")
text = text.lower()
text = filter(str.isalpha,text)
dict = collections.defaultdict(int)
for c in text:
    dict[c] += 1

for c in sorted(dict):
    if dict[c]>=0:
        print('%s%d' % (c,dict[c]))
```

```
a3
c1
d2
e12
f3
g2
h8
i3
j2
l1
m3
n8
o12
p1
r4
s5
t6
u1
v3
y2
```

5. Good... afternoon?

The code below generates a random time in the day. Suppose we want to present a user a welcoming message when the user opens a program at that time.

Exercise

- Print a message with the (pseudo) format: Good {part of day}, the time is hh:mm
- Parts of the day are night [0-5], morning [6-11], afternoon [12-17] or evening [18-23].
- Hour or minute values below 10 should have a leading 0.

Hint: you can use if-elif-else for the part of the day, but you can also have a fixed list of parts of the day and use clever .

```
from msilib.schema import EventMapping
import random
from winreg import ExpandEnvironmentStrings

h = random.randint(0, 23) # hour of the day
m = random.randint(0, 59) # minute in the hour
if h < 6:
    partofday = 'night'
elif h<12:
    partofday = 'morning'
elif h<18:
    partofday = 'afternoon'
else:
    partofday = 'evening'
h = "%02d" % h
m = "%02d" % m
print('Good ',partofday,', the time is',h,':',m)
```

```
Good  morning , the time is 11 : 33
```