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.

In [26]:
```
string = "In computer programming,  a string is traditionally  a sequence

print(string.index('c'))                      # index of the first 'c'
print(string.rindex('c'))                     # index of the last 'c' (see
print(len(string)-string.count(' '))          # length of the string witho
print(string.startswith('In'))                # whether the string starts
print(string.lower())                         # string as all lower-case
print(string.split(','))                      # list of parts of the sente
print(string.replace('  ',' '))               # all double whitespaces rep
print(string.replace(' traditionally ','')) # without the word "tradition
```

```
3
72
66
True
in computer programming,  a string is traditionally  a sequence of charac
ters.
['In computer programming', '  a string is traditionally  a sequence of c
haracters.  ']
In computer programming, a string is traditionally a sequence of characte
rs.
In computer programming,  a string is a sequence of characters.
```

2. String formatting</h2>

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.

In [1]:
```
client_name = "Obelix"
client_age = 32                                                     # [year
string = "Client %s is %d years old." % (client_name, client_age) # the f
print(string)
```

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

Exercise

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

In [57]:
```
value = 1.73456
print("%.0f    " % value)     # 2         (see "5. Precision", why can't y
print("%.1f    " % value)     # 1.7
print("%.2f    " % value)     # 1.73
print("%7.2f   " % value)     #    1.73   (with a total length of 7, see "
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
```

In [60]:
```python
regexp = "(\d*(\.|,)\d*)"
```

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.
  </ul> *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()).

In [71]:

```
text = "For the movie The Theory of Everything (2014), Jóhann Jóhannsson

#prepare and filter text
text=text.lower()
text=text.replace('ó','o')
text=text.replace('(2014),','')

#add alphabet
alphabet=['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm'

#loop for counting
for i in alphabet:
    if text.count(i)==0:
        continue
    else:
        print(i,'has frequency',text.count(i))
```

```
a has frequency 3
c has frequency 1
d has frequency 2
e has frequency 12
f has frequency 3
g has frequency 2
h has frequency 8
i has frequency 3
j has frequency 2
l has frequency 1
m has frequency 3
n has frequency 8
o has frequency 12
p has frequency 1
r has frequency 4
s has frequency 5
t has frequency 6
u has frequency 1
v has frequency 3
y has frequency 2
```

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 indexing from the hour value.

In [86]:
```
import random

h = random.randint(0, 23) # hour of the day
m = random.randint(0, 59) # minute in the hour

if h<=5:
    part='night'
elif h>=6 or h<=11:
    part='morning'
elif h>=12 or h<=17:
    part='afternoon'
else:
    part='evening'

if h<10:
    hh='0'+str(h)
else:
    hh=str(h)
if m<10:
    mm='0'+str(m)
else:
    mm=str(m)

print('Good '+part+', the time is',hh,':',mm)
```

```
Good morning, the time is 10 : 48
```