Example of using the PyRadiomics toolbox in Python

First, import some built-in Python modules needed to get our testing data. Second, import the toolbox, only the `featureextractor` is needed, this module handles the interaction with other parts of the toolbox.

In [ ]:
```
from __future__ import print_function
import six
import os  # needed navigate the system to get the input data

import radiomics
from radiomics import featureextractor  # This module is used for in
teraction with pyradiomics
```

Setting up data

Test cases can be downloaded to temporary files. This is handled by the `radiomics.getTestCase()` function, which checks if the requested test case is available and if not, downloads it. It returns a tuple with the location of the image and mask of the requested test case, or (None, None) if it fails.

Alternatively, if the data is available somewhere locally, this directory can be passed as a second argument to `radiomics.getTestCase()`. If that directory does not exist or does not contain the testcase, functionality reverts to default and tries to download the test data.

If getting the test case fails, PyRadiomics will log an error explaining the cause.

In [ ]:
```
# Get the testCase
imagePath, maskPath = radiomics.getTestCase('brain1')

if imagePath is None or maskPath is None:  # Something went wrong, i
n this case PyRadiomics will also log an error
    raise Exception('Error getting testcase!')  # Raise exception to
prevent cells below from running in case of "run all"

# Additonally, store the location of the example parameter file, sto
red in \pyradiomics\examples/exampleSettings
paramPath = os.path.join('..', 'examples', 'exampleSettings', 'Param
s.yaml')
print('Parameter file, absolute path:', os.path.abspath(paramPath))
```

Instantiating the extractor

Now that we have our input, we need to define the parameters and instantiate the extractor. For this there are three possibilities:

1. Use defaults, don't define custom settings
2. Define parameters in a dictionary, control filters and features after initialisation
3. Use a parameter file

Method 1, use defaults

In [ ]:
```
# Instantiate the extractor
extractor = featureextractor.RadiomicsFeatureExtractor()

print('Extraction parameters:\n\t', extractor.settings)
print('Enabled filters:\n\t', extractor.enabledImagetypes)
print('Enabled features:\n\t', extractor.enabledFeatures)
```

Method 2, hard-coded settings:

In [ ]:
```
# First define the settings
settings = {}
settings['binWidth'] = 20
settings['sigma'] = [1, 2, 3]

# Instantiate the extractor
extractor = featureextractor.RadiomicsFeatureExtractor(**settings)
# ** 'unpacks' the dictionary in the function call

print('Extraction parameters:\n\t', extractor.settings)
print('Enabled filters:\n\t', extractor.enabledImagetypes)  # Still
the default parameters
print('Enabled features:\n\t', extractor.enabledFeatures)  # Still t
he default parameters
```

In [ ]:
```
# This cell is equivalent to the previous cell
extractor = featureextractor.RadiomicsFeatureExtractor(binWidth=20,
sigma=[1, 2, 3])  # Equivalent of code above

print('Extraction parameters:\n\t', extractor.settings)
print('Enabled filters:\n\t', extractor.enabledImagetypes)  # Still
the default parameters
print('Enabled features:\n\t', extractor.enabledFeatures)  # Still t
he default parameters
```

In [ ]:
```
# Enable a filter (in addition to the 'Original' filter already enab
led)
extractor.enableImageTypeByName('LoG')
print('')
print('Enabled filters:\n\t', extractor.enabledImagetypes)

# Disable all feature classes, save firstorder
extractor.disableAllFeatures()
extractor.enableFeatureClassByName('firstorder')
print('')
print('Enabled features:\n\t', extractor.enabledFeatures)

# Specify some additional features in the GLCM feature class
extractor.enableFeaturesByName(glcm=['Autocorrelation', 'Homogeneity
1', 'SumSquares'])
print('')
print('Enabled features:\n\t', extractor.enabledFeatures)
```

Method 3, using a parameter file

In [ ]:
```
# Instantiate the extractor
extractor = featureextractor.RadiomicsFeatureExtractor(paramPath)

print('Extraction parameters:\n\t', extractor.settings)
print('Enabled filters:\n\t', extractor.enabledImagetypes)
print('Enabled features:\n\t', extractor.enabledFeatures)
```

Extract features

Now that we have our extractor set up with the correct parameters, we can start extracting features:

In [ ]:
```
result = extractor.execute(imagePath, maskPath)
```

In [ ]:
```
print('Result type:', type(result))  # result is returned in a Pytho
n ordered dictionary)
print('')
print('Calculated features')
for key, value in six.iteritems(result):
    print('\t', key, ':', value)
```