Back channels

Output results to a file

The built-in function `print` can be used to write the string representation of a result to a file. This can be done by making use of the optional parameter `file` of the function `print`.

By default, the function `print` will write the result to the special file `sys.stdout` (the default value of the parameter `file`) that for example might be attached to the Console window of PyCharm. The same effect can be obtained by passing the value `None` to the parameter `file`.

By passing a file object that was opened for writing to the parameter `file` of the function `print`, the string representation of the result is written to this file.

```
>>> line1 = 'This is the first line.'
>>> line2 = 'This is the second line.'
>>> print(line1)
This is the first line.
>>> print(line2, file=None)
This is the second line.

>>> outfile = open('output.txt', 'w')
>>> print(line1, file=outfile)
>>> print(line2, file=outfile)
>>> outfile.close()
>>> infile = open('input.txt', 'r')
>>> infile.readline()
'This is the first line.\n'
>>> infile.readline()
'This is the second line.\n'
>>> infile.readline()
''
```

Plutokiller

String representation of a grid

The following *list comprehension* constructs a string representation of a grid, with each the rows of the grid on a separate line. In other words, the lines are separated from each other by a newline (the string on which the outer `join` method is called). The elements of the rows are separated from each other by a singel space (the string on which the inner `join` method is called).

```
>>> grid = [['A', 'B', 'C'], ['D', 'E', 'F'], ['G', 'H', 'I']]
>>> print('\n'.join([' '.join(rij) for row in grid]))
A B C
D E F
G H I
```

Poem codes

The `random` module

The random module from the The Python Standard Library can be used to add randomness to your Python code. Here's a selection of the functions implemented by this module.

function	short description
`random()`	returns a random floating point number from the range $[0, 1[$
`randint(a, b)`	returns a random integer from the range $[a, b]$
`choice(s)`	returns a random element from the non-empty sequence `s`
`sample(s, k)`	returns `k` distinct elements from the sequence or set `s`
`shuffle(l)`	randomly shuffles the sequence s in place

Here are some examples.

```
>>> import random

>>> random.random()
0.954131645221452
>>> random.random()
0.3548429482674793

>>> random.randint(3, 10)
5
>>> random.randint(3, 10)
8

>>> aList = ['a', 'b', 'c']
>>> random.choice(aList)
'b'
>>> random.choice(aList)
'a'
>>> aList
['a', 'b', 'c']

>>> random.sample(aList, 2)
['a', 'c']
>>> random.sample(aList, 2)
['b', 'a']
>>> aList
['a', 'b', 'c']

>>> random.shuffle(aList)
>>> aList
['c', 'a', 'b']
```

General

Copy text file to PyCharm

If you want to locally test your solution for an assignment using text files, you must also make sure to have a local copy of the text files. Otherwise the test cases of the doctest will not be able to access these text files. The text files that are used in a given doctest are always linked in the description on top of the doctest. You can inspect the content of these text files in your browser by clicking this link.

The most general procedure to obtain a local copy of these text files in PyCharm goes as follows:

- open the text file in your browser
- copy the file content to the clipboard (`CTRL-A` + `CTRL-C`)
- create a new text file in Pycharm

  - right click the directory that needs to contain the text file (you must make sure that the text file is in the same directory as your Python script)
  - chose the menu item `New` and then the menu item `File`
  - enter the correct name of the file; make sure that the file extension must also be given (usually `.txt`)
- paste the content of the clipboard into the file (`CTRL-V`)

The following screenshot shows you the way.

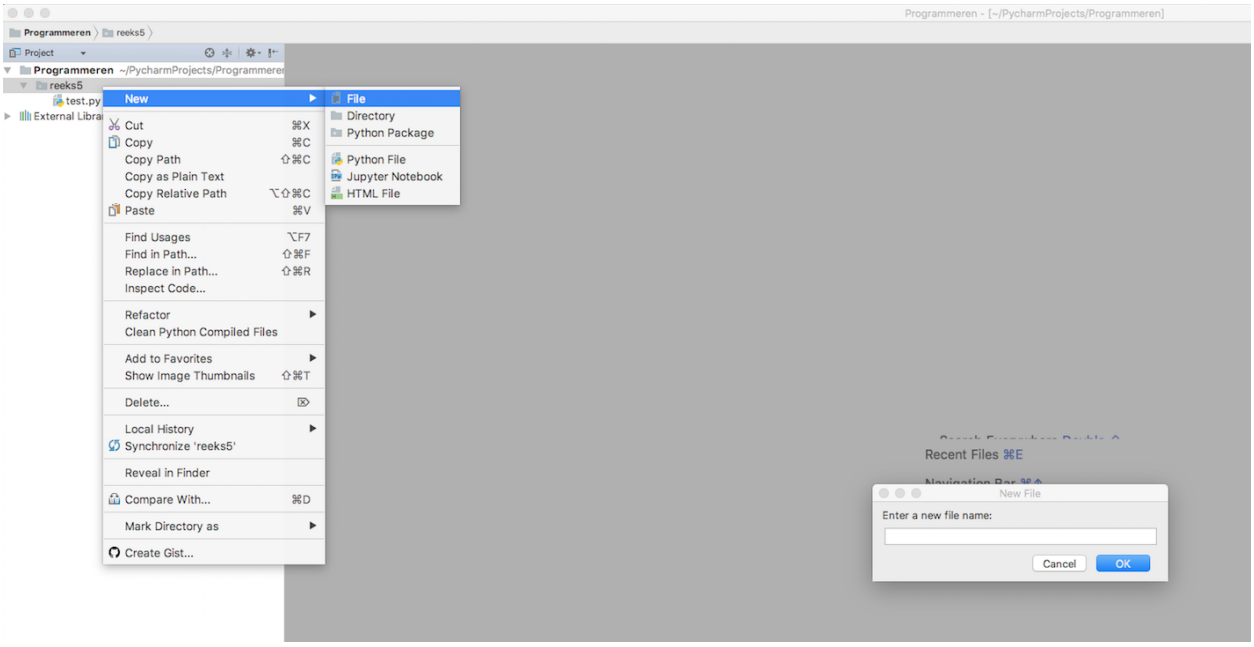

Figure 1: menu new file

If you submit a solution to Dodona, the platform will make sure that the necessary text files are in the same directory as the Python script.