Python Programming Language Foundation Hometask

You are proposed to implement pure-python command-line calculator using python 3.6.

Minimal requirements

Calculator should be a command-line utility which receives mathematical expression string as an argument and prints evaluated result:

```
$ pycalc '2+2*2'
6
```

It should provide the following interface:

```
$ pycalc --help
usage: pycalc [-h] EXPRESSION

Pure-python command-line calculator.

positional arguments:
  EXPRESSION            expression string to evaluate
```

In case of any mistakes in the expression utility should print human-readable error explanation with "ERROR: " prefix and exit with non-zero exit code:

```
$ pycalc '15*(25+1'
ERROR: brackets are not balanced

$ pycalc 'func'
ERROR: unknown function 'func'
```

Mathematical operations calculator must support

- Arithmetic (+, -, *, /, //, %, ^) (^ is a power).
- Comparison (<, <=, ==, !=, >=, >).
- 2 built-in python functions: `abs` and `round`.
- All functions and constants from standard python module `math` (trigonometry, logarithms, etc.).

Non-functional requirements

- It is mandatory to use `argparse` module.
- Codebase must be covered with unit tests with at least 70% coverage.
- Usage of external modules is prohibited (python standard library only).
- Usage of eval and exec is prohibited.
- Usage of ast and tokenize modules is prohibited.

Distribution

- Utility should be wrapped into distribution package with `setuptools`.
- This package should export CLI utility named `pycalc`.

Codestyle

- Docstrings are mandatory for all methods, classes, functions and modules.
- Code must correspond to pep8 (use `pycodestyle` utility for self-check).
    - You can set line length up to 120 symbols.

Optional requirements

These requirements are not mandatory for implementation, but you can get more points for them.

- Support of functions and constants from the modules provided with `-m` or `--use-modules` command-line option. This option should

accept names of the modules which are accessible via python standard module search paths. Functions and constants from user defined modules have higher priority in case of name conflict then stuff from `math` module or built-in functions.

In this case `pycalc` utility should provide the following interface:

```
$ pycalc --help
usage: pycalc [-h] EXPRESSION [-m MODULE [MODULE ...]]

Pure-python command-line calculator.

positional arguments:
EXPRESSION            expression string to evaluate

optional arguments:
-h, --help            show this help message and exit
-m MODULE [MODULE ...], --use-modules MODULE [MODULE ...]
                      additional modules to use
```

Usage example:

```
# my_module.py
def sin(number):
  return 42
```

```
$ pycalc 'sin(pi/2)'
1.0

$ pycalc 'sin(pi/2)' -m my_module
42

$ pycalc 'time()/3600/24/365' -m time
48.80147332327218
```

Implementations will be checked with the latest cPython interpreter of 3.6 branch.