ML for Finance

Fall 2019

Homework

- Provide your solutions inside special code cells.
  - You can check your functions within magic cells that begin with `assert` function.
  - Points for each task are provided in parentheses.
  - Any kind of plagiarism is assessed as 0 points for such tasks.
  - Below you can find criteria for transfer such Points to your final grade:

$\sum$ of pts	Final Grade
$0-3$	0
$4-6$	1
$7-10$	2
$11-15$	3
$16-20$	4
$21-25$	5
$26-30$	6
$31-35$	7
$36-40$	8
$41-45$	9
$46-50$	10

*some tasks are adapted from math-info.hse.ru and made</small>

#1 (1)

Write function `is_prime(n)` , which checks `n` is prime number (returns `True`) of not (returns `False`).

In [ ]:
```
# CODE HERE
```

In [ ]:
```
assert(is_prime(2) and
       is_prime(3) and
       is_prime(139) and
       is_prime(617) and
       is_prime(293) and
       not is_prime(12) and
       not is_prime(13*7))
```

#2 (4)

Let the ariphmetic expression contains brackets of two types: `( )`, `[ ]`. Each open bracket should be correctly closed within close brackets of the same type.

Examples:

- `5 * [1 + 2 + (3 + 4) + 5]` — correct
- `4 + (2 +` - incorrect
- `[1 + (2 + 3] + 4)` - incorrect

Write function `check_brackets(expr)`, which take the string input with expression. Returns `True` if `expr` is correct and `False` otherwise.

In [ ]:
```
# CODE HERE
```

In [ ]:
```
assert check_brackets('5 * [1 + 2 + (3 + 4) + 5]')
assert not check_brackets('4 + (2 + ')
assert not check_brackets('[1 + (2 + 3] + 4)')
assert check_brackets('[]()[][[([([])])])]]')
assert not check_brackets(']')
assert not check_brackets(')')
assert not check_brackets('[)]')
assert not check_brackets('[)()]')
assert not check_brackets('[)](')
assert not check_brackets("[[([()])]]")
assert not check_brackets('[')
assert not check_brackets('(')
assert not check_brackets('(' * 100 + ')' * 99)
assert check_brackets('(' * 1000 + ')' * 1000)
assert not check_brackets('([' + '(' * 998 + ')' * 909 + ']')
```

#3 (6)

Imagine, you help the producer to choose proper shooting days. The producer asks you to choose two days in order to ensure that the temperature in such days will be not more than 5 degree different between each other (by the absolute value). You know the weather forecast for n few days (the sequence of integer numbers: $t_1, t_2, \ldots, t_n$, where i - the number of the days in the increasing order, t_i - the temperature in the particular day). For two choosen days should be true the following: $|t_i - t_j| \leq 5$. You know also that the producer do not want to have long pauses between these days, so, you need also to minimize the distance between two particular days.

Input: you have the list of numbers (temperature in days, ordered by the number of the day). **Output**: write the function `mindays(list)`, that returns the minimum days of leisure (pause) in between for the producer; if no such days you can choose - return -1.

In [ ]:
```
# CODE HERE
```

In [ ]:
```
assert mindays([4, 12, -5, 8, 14]) == 1
assert mindays([10, 10, 10]) == 0
assert mindays([-20, -14, -8, -2]) == -1
assert mindays([-3, 1, -5, 8, 14]) == 0
assert mindays([-4, 2, 8, 0]) == 1
assert mindays([-4, 2, -5, 0]) == 0
```