# Python (BSU FAMCS Fall'19)

## Seminar 2

Advisor: Dzmitryi Kasitsyn

**Task 1. (0.5 points).** Implement a function that takes a natural number $n$ as an argument and calculates $1^2 * 2 + 2^2 * 3 + ... + (n-1)^2 * n$ sum. Use only one comprehension expression in your solution. Asymptotical complexity of the algorithm should be $O(n)$.
***Note.*** Name your function *calculate_special_sum* and save it into a *special_sum.py* file.

**Example**

```
assert calculate_special_sum(3) == 14
```

**Task 2. (0.5 points).** Implement a function that takes a sequence (a `list` or a `tuple`) and returns an *iterable* of pairs containing a unique item and number of times the item is presented in the given sequence. A pair is tuple of 2 elements. Pairs in the *iterable* returned may be ordered arbitrary.
***Note.*** Name your function *compress* and save it into a *unique.py* file.

**Example**

```
expected_sorted = [(1, 2), (2, 1), (3, 1)]
actual_sorted = sorted(compress([1, 2, 1, 3]))
assert expected_sorted == actual_sorted
```

**Task 3. (1.5 points).** Implement a function that produces all primes that aren't greater than a given narutal number $n$. Use only one comprehension to provide a result. Nested comprehensions are permitted.
***Note.*** Name your function *get_primes* and save it into a *primes.py* file.

**Example**

```
assert [2, 3, 5, 7, 11] == sorted(get_primes(11))
```

**Task 4. (1.5 points).** You are asked to calculate a histogram containing $k$ cells from a sample given as a `list`.

In other words you have to find minimum and maximum items in a sample, split the interval between them into $k$ equal subintervals ($1 \leqslant k \leqslant d$, $d$ – number of distinct items in a given sample) and count the number of items that belong to each subinterval (lie between the bounds). Left bound of each interval is supposed to be included in it, the right bound should be excluded for all subintervals besides the last one.

Your solution function should take two parameters : a sample and $k$ – the number of cells – and provide a `list` of $k$ items length containing the number of items in each cell.

Asymptotical complexity of your solution is expected to be $O(n)$, where $n$ is the length of a given sample.
***Note.*** Name your function *distribute* and save it into a *hist.py* file.

**Example**

```
assert distribute([1.25, 1, 2, 1.75], 2) == [2, 2]
```