Задание 1.
Исходные данные:
Найдем выборочное среднее роста хоккеистов из датасета по формуле.

Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.read_csv('hockey.csv')
>>> df.head()
   year country  no              name  ...                club        age  cohort        bmi
0  2001     RUS  10    tverdovsky oleg  ...  anaheim mighty ducks   24.952772    1976  24.543462
1  2001     RUS   2  vichnevsky vitali  ...  anaheim mighty ducks   21.119781    1980  24.332277
2  2001     RUS  26  petrochinin evgeni  ...  severstal cherepovetal   25.229295    1976  28.680111
3  2001     RUS  28    zhdan alexander  ...          ak bars kazan   29.675565    1971  26.827421
4  2001     RUS  32    orekhovsky oleg  ...          dynamo moscow   23.490760    1977  28.734694

[5 rows x 13 columns]
>>> mean_height = df['height'].sum() / df['height'].shape[0]
>>> mean_height
183.81150667514305
>>> df['height'].mean()
183.81150667514305
>>>
```

Задание 2.

Исходные данные:

Посчитаем выборочную дисперсию роста хоккеистов.

Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.read_csv('hockey.csv')
>>> df.head()
   year country  no              name  ...                club        age  cohort        bmi
0  2001     RUS  10     tverdovsky oleg  ...    anaheim mighty ducks   24.952772    1976  24.543462
1  2001     RUS   2   vichnevsky vitali  ...    anaheim mighty ducks   21.119781    1980  24.332277
2  2001     RUS  26  petrochinin evgeni  ...  severstal cherepovetal   25.229295    1976  28.680111
3  2001     RUS  28     zhdan alexander  ...          ak bars kazan   29.675565    1971  26.827421
4  2001     RUS  32     orekhovsky oleg  ...          dynamo moscow   23.490760    1977  28.734694

[5 rows x 13 columns]
>>>
>>> ((df['height'] - df['height'].mean()) ** 2).sum() / df['height'].shape[0]
28.981317056058433
>>>
>>> (df['height'] - df['height'].mean()) ** 2
0        1.412516
1       17.543476
2        3.281556
3       33.773610
4       77.642650
          ...
6287     0.658543
6288    33.773610
6289    51.674436
6290    17.543476
6291    84.428410
Name: height, Length: 6292, dtype: float64
>>>
>>> df['height'].var(ddof=1)
28.985923846243786
>>>
>>> df['height'].var(ddof=0)
28.981317056058433
>>>
```

```
>>> ((df['height'] - df['height'].mean()) ** 2).sum() / df['height'].shape[0]
28.981317056058433
>>>
>>>
```

Help on function var in module pandas.core.frame:

```
var(self, axis=None, skipna=None, level=None, ddof=1, numeric_only=None, **kwargs)
    Return unbiased variance over requested axis.

    Normalized by N-1 by default. This can be changed using the ddof argument

    Parameters
    ----------
    axis : {index (0), columns (1)}
    skipna : bool, default True
        Exclude NA/null values. If an entire row/column is NA, the result
        will be NA
    level : int or level name, default None
        If the axis is a MultiIndex (hierarchical), count along a
        particular level, collapsing into a Series
    ddof : int, default 1
        Delta Degrees of Freedom. The divisor used in calculations is N - ddof,
        where N represents the number of elements.
    numeric_only : bool, default None
        Include only float, int, boolean columns. If None, will attempt to use
        everything, then use only numeric data. Not implemented for Series.

    Returns
    -------
    Series or DataFrame (if level specified)
~
~
~
~
~
```

~
~
~
~
~
~
~
~
~
(END)

Help on function var in module numpy:

var(a, axis=None, dtype=None, out=None, ddof=0, keepdims=<no value>)
    Compute the variance along the specified axis.

    Returns the variance of the array elements, a measure of the spread of a
    distribution. The variance is computed for the flattened array by
    default, otherwise over the specified axis.

    Parameters
    ----
    a : array_like
        Array containing numbers whose variance is desired. If `a` is not an
        array, a conversion is attempted.
    axis : None or int or tuple of ints, optional
        Axis or axes along which the variance is computed. The default is to
        compute the variance of the flattened array.

        .. versionadded:: 1.7.0

        If this is a tuple of ints, a variance is performed over multiple axes,
        instead of a single axis or all the axes as before.
    dtype : data-type, optional
        Type to use in computing the variance. For arrays of integer type
        the default is `float32`; for arrays of float types it is the same as

the array type.

out : ndarray, optional

Alternate output array in which to place the result. It must have

the same shape as the expected output, but the type is cast if

necessary.

ddof : int, optional

"Delta Degrees of Freedom": the divisor used in the calculation is

``N - ddof``, where ``N`` represents the number of elements. By

default `ddof` is zero.

keepdims : bool, optional

If this is set to True, the axes which are reduced are left

in the result as dimensions with size one. With this option,

the result will broadcast correctly against the input array.

If the default value is passed, then `keepdims` will not be

:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> a = np.array( [ [1, 2], [3, 4] ])
>>> np.var(a, axis=0)
array([1., 1.])
>>>
>>> np.var(a, axis=1)
array([0.25, 0.25])
>>>

vik@vik-Z580:~$ command python3
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> a = np.zeros(([2, 512*512]), dtype=np.float32)
>>> np.var(a)
```

```
0.0
>>>
>>> b = np.zeros(([2, 512*512]), dtype=np.float64)
>>> np.var(b)
0.0
>>>
```

Задание 3.

Исходные данные:

Среднее квадратическое отклонение роста хоккеистов:

Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.read_csv('hockey.csv')
>>> df.head()
   year country  no              name  ...                  club        age  cohort        bmi
0  2001     RUS  10    tverdovsky oleg  ...    anaheim mighty ducks   24.952772    1976  24.543462
1  2001     RUS   2  vichnevsky vitali  ...    anaheim mighty ducks   21.119781    1980  24.332277
2  2001     RUS  26  petrochinin evgeni  ...  severstal cherepovetal   25.229295    1976  28.680111
3  2001     RUS  28    zhdan alexander  ...           ak bars kazan   29.675565    1971  26.827421
4  2001     RUS  32    orekhovsky oleg  ...          dynamo moscow   23.490760    1977  28.734694

[5 rows x 13 columns]
>>>
>>> np.sqrt(((df['height'] - df['height'].mean()) ** 2).sum() / (df['height'].shape[0] - 1))
5.3838577104381
>>>
>>> df['height'].std(ddof=1)
5.3838577104381
>>>
>>>
```

Задание4.

Исходные данные:

Посчитаем медиану роста хоккеистов. Для начала воспользуемся определением. Нужно отсортировать значения из выборки и взять середину этого массива. Посчитаем размер выборки:

Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> import pandas as pd
>>> df = pd.read_csv('hockey.csv')
>>> height = sorted(df['height'])
>>> length = len(height)
>>> length
6292
>>>
>>> height[length // 2 - 1], height[length // 2]
(183, 183)
>>> median = (height[length // 2 - 1] + height[length // 2]) / 2
>>> median
183.0
>>> (df['height'] <= median).sum() / length, (df['height'] >= median).sum() / length
(0.5028607755880483, 0.5902733630006357)
>>>
>>>
>>> df['height'].median()
183.0
>>> df['height'].mean()
183.81150667514305
>>>
>>>
```

Help on function quantile in module numpy:

quantile(a, q, axis=None, out=None, overwrite_input=False, interpolation='linear', keepdims=False)

Compute the q-th quantile of the data along the specified axis.

.. versionadded:: 1.15.0

Parameters
---
a : array_like
    Input array or object that can be converted to an array.
q : array_like of float
    Quantile or sequence of quantiles to compute, which must be between
    0 and 1 inclusive.
axis : {int, tuple of int, None}, optional
    Axis or axes along which the quantiles are computed. The
    default is to compute the quantile(s) along a flattened
    version of the array.
out : ndarray, optional
    Alternative output array in which to place the result. It must
    have the same shape and buffer length as the expected output,
    but the type (of the output) will be cast if necessary.
overwrite_input : bool, optional
    If True, then allow the input array `a` to be modified by intermediate
    calculations, to save memory. In this case, the contents of the input
    `a` after this function completes is undefined.
interpolation : {'linear', 'lower', 'higher', 'midpoint', 'nearest'}
    This optional parameter specifies the interpolation method to
    use when the desired quantile lies between two data points
    ``i < j``:

        * linear: ``i + (j - i) * fraction``, where ``fraction``
          is the fractional part of the index surrounded by ``i``
          and ``j``.
        * lower: ``i``.
        * higher: ``j``.
        * nearest: ``i`` or ``j``, whichever is nearest.
        * midpoint: ``(i + j) / 2``.
keepdims : bool, optional

If this is set to True, the axes which are reduced are left in

:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)
[GCC 9.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import numpy as np
>>> a = np.array([ [10, 7, 4], [3, 2, 1] ])
>>> a
array([[10,  7,  4],
       [ 3,  2,  1]])
>>>
>>> np.quantile(a, 0.5)
3.5
>>> np.quantile(a, 0.5, axis=0)
array([6.5, 4.5, 2.5])
>>> np.quantile(a, 0.5, axis=1)
array([7., 2.])
>>>
>>> m = np.quantile(a, 0.5, axis=0)
>>> out = np.zeros_like(m)
>>> mp.quantile(a, 0.5, axis=0, out=out)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'mp' is not defined
>>> np.quantile(a, 0.5)
3.5
>>> np.quantile(a, 0.5, axis=0, out=out)
array([6.5, 4.5, 2.5])
>>> m
array([6.5, 4.5, 2.5])
>>> b = a.copy()
>>> overwrite_input=True
>>> np.quantile(b, 0.5, axis=1, overwrite_input=True)
array([7., 2.])
>>>
```

Задание5.

Исходные данные:

Посчитаем первый и третий квартили значений роста хоккеистов. Вычисления аналогичны случаю вычисления медианы, однако нам нужно делить массив не в пропорции $1 : 1$, а в пропорции $\alpha : (1 - \alpha)$.

Посчитаем первый квартиль, т.е. квартиль порядка 0.25. Процесс похож на вычисление медианы, просто делить будем не на 2, а на 4. Правило вычисления квартиля:

- Если размер массива делится на 4 нацело, то первый квартиль будет находиться между крайним правым элементом из левой четверти (в отсортированном массиве) и элементом, следующим за ним.
- Если размер массива на 4 не делится, то первым квартилем будет элемент, стоящий на позиции $[n / 4] + 1$, где $[x]$ — целая часть числа x.

Итак, остаток от деления длины массива на 4:

Решение:

```
Python 3.8.5 (default, May 27 2021, 13:30:53)

[GCC 9.3.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import numpy as np

>>> import pandas as pd

>>> df = pd.read_csv('hockey.csv')

>>> df.head()

   year country  no              name  ...                club       age  cohort       bmi

0  2001     RUS  10     tverdovsky oleg  ...  anaheim mighty ducks  24.952772    1976  24.543462
```

```
1  2001     RUS   2   vichnevsky vitali   ...    anaheim mighty ducks   21.119781    1980  24.332277

2  2001     RUS  26  petrochinin evgeni   ...  severstal cherepovetal   25.229295    1976  28.680111

3  2001     RUS  28     zhdan alexander   ...          ak bars kazan   29.675565    1971  26.827421

4  2001     RUS  32     orekhovsky oleg   ...          dynamo moscow   23.490760    1977  28.734694



[5 rows x 13 columns]

>>>

>>> height = sorted(df['height'])

>>> length = len(height)

>>> length

6292

>>>

>>> height[length //4 -1], height[length // 4]

(180, 180)

>>> q1 = height[length // 4]

>>> q1
```

```
180

>>>

>>> (df['height'] <= q1).sum() / length, (df['height'] >= q1).sum() / length

(0.28877940241576605, 0.8034011443102352)

>>>

>>> df['height'].quantile(0.25)

180.0

>>> q3 = height[3 * length // 4]

>>> q3

188

>>>

>>> (df['height'] <= q3).sum() / length, (df['height'] >= q3).sum() / length

(0.8161157024793388, 0.2598537825810553)

>>>

>>> df['height'].quantile(0.75)

188.0
```

```
>>>

>>> df['height'].quantile([0.25, 0.5, 0.75])

0.25    180.0

0.50    183.0

0.75    188.0

Name: height, dtype: float64

>>>

>>> df['height'].describe()

count    6292.000000

mean      183.811507

std         5.383858

min       165.000000

25%       180.000000

50%       183.000000

75%       188.000000

max       205.000000
```

```
Name: height, dtype: float64

>>>

>>>

>>> a = np.linspace(0, 1000, 102)

>>>

>>> q_our = sorted(a)[a.shape[0] // 4]

>>> q_numpy = np.quantile(a, 0.25)

>>> q_our, q_numpy

(247.5247524752475, 250.0)

>>> (a <= q_our).sum() / a.shape[0], (a >= q_our).sum() / a.shape[0]

(0.2549019607843137, 0.7549019607843137)

>>>

>>> (a <= q_numpy).sum() / a.shape[0], (a >= q_numpy).sum() / a.shape[0]

(0.2549019607843137, 0.7450980392156863)

>>>

>>>
```

```
Python 3.8.5 (default, May 27 2021, 13:30:53)

[GCC 9.3.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import numpy as np

>>> ar = np.ones(1000)

>>> ar[0] = 10000

>>> ar.mean(), ar.std()

(10.999, 316.03800562432366)

>>>

>>>

>>> np.quantile(ar, [0.25, 0.75])

array([1., 1.])

>>>

>>>
```