Федеральное государственное бюджетное образовательное учреждение
высшего образования «Кемеровский государственный университет»
Новокузнецкий институт (филиал)

Факультет информатики, математики и экономики
Кафедра информатики и вычислительной техники им. В. К. Буторина

О. А. Штейнбрехер

Обработка статистических данных на языке Python

Методические указания по выполнению лабораторных работ по теме «Библиотеки обработки данных языка Python» дисциплине «Основы программирования» для обучающихся по направлению подготовки 39.03.01 Социология Профиль «Социологические и маркетинговые исследования»

Новокузнецк
2021

УДК [378.147.88:004.4](072)
ББК 74.484(2Рос-4Кем)я73+ 32.972я73
Ш88

Ш88 «Обработка статистических данных на языке Python. Методические указания по выполнению лабораторных работ по теме «Библиотеки обработки данных языка Python» : метод. указ (текст. электрон. изд.)/ О.А. Штейнбрехер ; Новокузнец. ин-т (фил.) Кемеров. гос. ун-та – Новокузнецк: НФИ КемГУ, 2021. – 25 с.

Приводятся методические указания по выполнению практических работ по разделу «Библиотеки обработки данных языка Python» дисциплины «Основы программирования».

Методические указания предназначены для студентов всех форм обучения направлений 39.03.01 Социология Профиль «Социологические и маркетинговые исследования».

Рекомендовано
на заседании кафедры
информатики и вычислительной техники им. В. К. Буторина
23 ноября 2020 года.
Заведующий кафедрой
А. В. Маркидонов

Утверждено
методической комиссией факультета информатики, математики и экономики
12 января 2021 года.
Председатель методкомиссии
Г.Н. Бойченко

УДК [378.147.88:004.4](072)
ББК 74.484(2Рос-4Кем)я73+ 32.972я7

ОГЛАВЛЕНИЕ

Введение

Дисциплина «Основы программирования» относится к основной части образовательной программы 39.03.01 Социология. По итогам изучения дисциплины обучающийся должен освоить компетенцию ОПК-1 Способен применять современные информационно-коммуникационные технологии в профессиональной деятельности социолога.

Обучающийся должен показать следующие результаты.

Знает:

- понятия и методы алгоритмизации;
- основы и методы структурного программирования;
- основные понятия объектно-ориентированного программирования;
- синтаксис базовых конструкций языка Python;
- основные библиотеки обработки данных языка Python.

Умеет:

- разрабатывать алгоритмы для решения прикладных практических задач, задач анализа данных;
- разрабатывать программы для реализации прикладных практических задач, задач анализа данных;
- обоснованно выбирать и применять стандартные библиотеки и алгоритмы для решения задач анализа данных;
- разрабатывать алгоритмы обработки данных на основе базовых алгоритмов;
- решать задачи, связанные с обработкой социологической информации, самостоятельно определять структуры данных, необходимых для решения.

Владеет:

- методами структурного и объектно-ориентированного программирования;
- навыками реализации алгоритмов и программ;
- навыками подготовки задания для программирования анализа данных в социологических исследованиях;
- навыками использования базовых технологий объектно-ориентированного программирования для решения социологических задач, прикладных практических задач, задач анализа данных.

Дисциплина предполагает освоение базового синтаксиса языка Python и возможностей использования встроенных и свободных библиотек для статистики. Python — высокоуровневый язык программирования общего назначения, ориентированный на

повышение производительности разработчика и читаемости кода. Синтаксис ядра python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты.

Альтернативой python может выступать язык веб-разметки PHP. Но у него имеется ряд недостатков.

1. PHP не имеет архитектуру, благодаря которой его можно назвать продуманным и надежным языком. Python—это стек, который легче понять и использовать.

2. PHP широко документирован и придерживается классического подхода. С другой стороны, python использует довольно строгие требования к отступам. Можно утверждать, что python более читабелен не только по сравнению с PHP, но и с большинством других языков программирования.

3. python требуется меньше средств отладки, чем PHP.

Так же альтернативой python может выступать высокоуровневый язык программирования ruby. Но он так же имеет недостатки.

В сфере применения очевидно преимущество занимает python. Он применяется в:

- web-разработка;
- мобильных и десктопных приложениях;
- играх;
- больших данных;
- искусственном интеллекте;
- сетевом администрировании.

Краткие теоретические сведения

Python (питон) — это высокоуровневый язык программирования общего назначения, который стал одним из ведущих и популярнейших в сообществе программистов. По своим возможностям он классифицируется от разработки упрощенных приложений до проведения сложных математических вычислений с одинаковым уровнем сложности.

Синтаксис Python имеет следующие особенности.

- Конец строки является концом инструкции (точка с запятой не требуется).
- Вложенные инструкции объединяются в блоки по величине отступов. Отступ может быть любым, главное, чтобы в пределах одного вложенного блока отступ был одинаков. И про читаемость кода не забывайте. Отступ в 1 пробел, к примеру, не лучшее решение. Используйте 4 пробела (или знак табуляции, на худой конец).
- Вложенные инструкции в Python записываются в соответствии с одним и тем же шаблоном, когда основная инструкция завершается двоеточием, вслед за которым располагается вложенный блок кода, обычно с отступом под строкой основной инструкции.

Синтаксис

Идентификаторы в Python — это имена, используемые для определения («идентификации«) переменных, функций, классов, модулей и других объектов. Идентификатор начинается с букв A-Z или a-z, либо знака подчеркивания (_), после чего следуют ноль или больше букв (*совет — никогда не создавайте свою собственную переменную с именем «_», т.к. это имя зарезервировано самими интерпретатором), знаков подчеркивания или цифр от 0 до 9.

В идентификаторах Python не используются знаки @, $ и %. Так же — Python чувствителен к регистру символов, т.е. Manpower и manpower являются двумя различными именами (идентификаторами).

Хеш-тег (#), который не находится внутри строки задаёт начало комментария. Все символы после # и до конца строки являются частью комментария, и Python игнорирует их.

Python относится к языкам с неявной сильной динамической типизацией. Неявная типизация означает, что при объявлении переменной вам не нужно указывать её тип, при явной – это делать необходимо.

В Python типы данных можно разделить на встроенные в интерпретатор *(built-in*) и не встроенные, которые можно использовать при импортировании соответствующих модулей.

К основным встроенным типам относятся:

1. *None* (неопределенное значение переменной)
2. Логические переменные (*Boolean Type*)
3. Числа (*Numeric Type*)
    a. *int* – целое число
    b. *float* – число с плавающей точкой
    c. *complex* – комплексное число
4. Списки (*Sequence Type*)
    a. *list* – список
    b. *tuple* – кортеж
    c. *range* – диапазон
5. Строки (*Text Sequence Type*)
    a. *str*
6. Бинарные списки (*Binary Sequence Types*)
    a. *bytes* – байты
    b. *bytearray* – массивы байт
    c. *memoryview* – специальные объекты для доступа к внутренним данным объекта через protocol buffer
7. Множества (*Set Types*)
    a. *set* – множество
    b. *frozenset* – неизменяемое множество
8. Словари (*Mapping Types*)
    a. *dict* – словарь

В Python существуют изменяемые и неизменяемые типы.

К неизменяемым (immutable) типам относятся: целые числа (int), числа с плавающей точкой (float), комплексные числа (complex), логические переменные (bool), кортежи (tuple), строки (str) и неизменяемые множества (frozen set).

К изменяемым (mutable) типам относятся: списки (list), множества (set), словари (dict).

Как уже было сказано ранее, при создании переменной, вначале создается объект, который имеет уникальный идентификатор, тип и значение, после этого переменная может ссылаться на созданный объект.

Неизменяемость типа данных означает, что созданный объект больше не изменяется. Например, если мы объявим переменную k = 15, то будет создан объект со значением 15, типа int и идентификатором, который можно узнать с помощью функции id().

Список основных операций для чисел:

1. A+B — сумма;
2. A-B — разность;
3. A*B — произведение;
4. A/B — частное;
5. A**B — возведение в степень.

Полезно помнить, что квадратный корень из числа x — это x**0.5, а корень степени n — это x**(1/n). Есть также унарный вариант операции -, то есть это операция с одним аргументом. Она возвращает число, противоположное данному. Например: -A.

Ветвление (или условная инструкция) в Python имеет следующий синтаксис:

```
if Условие:
    Блок_инструкций_1
else:
    Блок_инструкций_2
```

Блок_инструкций_1 будет выполнен, если Условие истинно. Если Условие ложно, будет выполнен Блок_инструкций_2.

В условной инструкции может отсутствовать слово else и последующий блок. Такая инструкция называется неполным ветвлением. Например, если дано число x и мы хотим заменить его на абсолютную величину x, то это можно сделать следующим образом:

```
if x < 0:
    x = -x
print(x)
```

В этом примере переменной x будет присвоено значение -x, но только в том случае, когда x<0. А вот инструкция print(x) будет выполнена всегда, независимо от проверяемого условия.

Цикл while («пока») позволяет выполнить одну и ту же последовательность действий, пока проверяемое условие истинно. Условие записывается до тела цикла и проверяется до выполнения тела цикла. Как правило, цикл while используется, когда невозможно определить точное значение количества проходов исполнения цикла.

Синтаксис цикла while в простейшем случае выглядит так:

```
while Условие:
    Блок_инструкций
```

Цикл for может быть использован как более краткая альтернатива циклу while.

Для последовательного перебора целых чисел из диапазона [0; n) можно использовать цикл for:

```
for i in range(10):
   print(i)
```

Этот код по выполняемым действиям полностью соответствуют циклу while:

```
i = 0
while i < 10:
  print(i)
```

```
  i += 1
```

Можно задавать начальные и конечные значения для переменной цикла, а также шаг:

```
for i in range(20, 10, -2):
  print(i)
```

Аналогичный цикл while

```
i = 20
while i > 10:
  print(i)
  i -= 2
```

Являясь одним из ведущих языков программирования, он имеет много фреймворков (платформ для построения приложений) и библиотек, которыми можно воспользоваться. Библиотека языка программирования — это набор модулей и функций, которые облегчают некоторые специфические операции с использованием этого языка программирования.

Основные математические библиотеки и команды для проведения статистических расчетов на языке Python

Для работы с анализом данных и статической обработки Python имеет следующие сторонние библиотеки.

1. Pandas - это пакет Python с открытым исходным кодом, который предоставляет высокоэффективные, простые в использовании структуры данных и инструменты анализа для помеченных данных на языке программирования Python. Он предназначен для быстрой и простой обработки данных, чтения, агрегирования и визуализации.
2. NumPy - универсальный пакет для обработки массивов. Он предоставляет высокопроизводительные объекты многомерных массивов и инструменты для работы с массивами. NumPy - это эффективный контейнер универсальных многомерных данных. Основной объект NumPy - это однородный многомерный массив. Это таблица элементов или чисел одного и того же типа данных, проиндексированная набором натуральных чисел. В NumPy размеры называются осями, а число осей называется рангом. Класс массива NumPy называется ndarray, он же array. NumPy используется для обработки массивов, в которых хранятся значения одного и того же типа данных.
3. Библиотека SciPy содержит модули для эффективных математических процедур, таких как линейная алгебра, интерполяция, оптимизация, интеграция и статистика. Основной функционал библиотеки SciPy построен на NumPy и его массивах. Он имеет различные модули для выполнения общих задач научного программирования, таких как линейная алгебра, интеграция, матанализ, обыкновенные дифференциальные уравнения и обработка сигналов.

4. Matplotlib - это библиотека Python, предоставляющая API для встраивания графиков в приложения.
5. Statsmodels - это универсальный пакет Python, который обеспечивает простые вычисления для описательной статистики и оценки и формирования статистических моделей.

Кроме того, имеется библиотека Python statistics — встроенная библиотека Python для описательной статистики. Рекомендуется использовать её, если наборы данных не слишком велики или если нельзя полагаться на импорт других библиотек.

Графические библиотеки

Графические команды - это функции, которые, принимая некоторые параметры, возвращают какой-то графический результат. Примеры графических методов высокоуровневого языка программирования python представлены в таблице 1.

Таблица 1 – Графические команды языка python

	Библиотека	Метод
Гистограмма	Matplotlib	plt.hist()
Столбчатая диаграмма	Matplotlib	plt.bar()
Полигон	Matplotlib	plt.scatter()
Круговая диаграмма	Matplotlib	plt.pie()
Графические функции	Matplotlib	plt.plot()

После установки библиотеки matplotlib вызывается в консоли или в скрипте как модуль. В Matplotlib заложены как простые графические команды, так и достаточно сложные. Доступ к ним через pyplot означает использование синтаксиса вида "plt.название_команды()".

Главной единицей при работе с matplotlib является рисунок (Figure). Любой рисунок в matplotlib имеет вложенную структуру:

Figure(Рисунок) -> Axes(Область рисования) -> Axis(Координатная ось).

Рисунок (Figure). Рисунок является объектом самого верхнего уровня, на котором располагаются одна или несколько областей рисования (Axes), элементы рисунка (заголовки, легенда и т.д.) и основа-холст (Canvas). На рисунке может быть несколько областей рисования Axes, но данная область рисования Axes может принадлежать только одному рисунку Figure.

Область рисования (Axes). Область рисования является объектом среднего уровня, который является главным объектом работы с графикой matplotlib в объектно-ориентированом стиле. Каждая область рисования Axes содержит две (или три в случае трёхмерных данных) координатных оси (Axis объектов), которые упорядочивают отображение данных.

Координатная ось (Axis). Координатная ось являются объектом среднего уровня, которые определяют область изменения данных, на них наносятся деления ticks и подписи к делениям ticklabels. Расположение делений определяется объектом Locator, а подписи делений обрабатывает объект Formatter. Конфигурация координатных осей заключается в комбинировании различных свойств объектов Locator и Formatter.

Элементы рисунка (Artists). Практически всё, что отображается на рисунке является элементом рисунка (Artist), объекты Figure, Axes и Axis. Элементы рисунка Artists включают в себя такие простые объекты как текст (Text), плоская линия (Line2D), фигура (Patch) и другие.

Контейнеры - это объекты-хранилища, на которые можно наносить графические примитивы. Всего существует 4 вида Artists контейнеров:

- Контейнеры рисунка (Figure containers)

Figure - это контейнер самого высокого уровня. На нём располагаются все другие контейнеры и графические примитивы.

- Контейнеры областей рисования (Axes containers)

Экземпляры Axes - это области, располагающиеся в контейнере Figure, для которых можно задавать координатную систему (декартовая или полярная). На нём располагаются все другие контейнеры, кроме Figure, и графические примитивы. Это области на рисунке, на которых располагаются графики и диаграммы, в которые вставляются изображения.

- Контейнеры осей (Axis containers)

Этот контейнер обслуживает экземпляры Axes. Он отвечает за создание координатных осей, на которые будут наноситься деления осей, подписи делений и линий вспомогательной сетки.

- Контейнеры делений (Tick containers)

Контейнер низшего уровня. Его специализация - задавать характеристики (цвет, толщина линий) линий сетки, делений и их подписей (размеры и типы шрифтов).

каждый рисунок можно структурно представить следующим образом:

1. Область рисования Axes
    - Заголовок области рисования -> plt.title();
2. Ось абсцисс Xaxis

- Подпись оси абсцисс OX -> plt.xlabel();

3. Ось абсцисс Yaxis
    - Подпись оси абсцисс OY -> plt.ylabel();
4. Легенда -> plt.legend();
5. Цветовая шкала -> plt.colorbar()
    - Подпись горизонтальной оси абсцисс OY -> cbar.ax.set_xlabel();
    - Подпись вертикальной оси абсцисс OY -> cbar.ax.set_ylabel();
6. Деления на оси абсцисс OX -> plt.xticks();
7. Деления на оси ординат OY -> plt.yticks().

Команды встроенной библиотеки Statictics языка Python

Функция mean(data) предназначена для вычисления среднего значения некоторых данных. Оно рассчитывается путем деления суммы всех точек данных на количество точек данных. Если данные пустые, будет генерироваться StatisticsError Пример:

```
import statistics
from fractions import Fraction as F
from decimal import Decimal as D

statistics.mean([11, 2, 13, 14, 44])
# returns 16.8

statistics.mean([F(8, 10), F(11, 20), F(2, 5), F(28, 5)])
# returns Fraction(147, 80)

statistics.mean([D("1.5"), D("5.75"), D("10.625"), D("2.375")])
# returns Decimal('5.0625')
```

Функция mode() возвращает наиболее распространенную точку данных из дискретных числовых и нецифровых данных. Это единственная статистическая функция, которая может использоваться с нечисловыми данными.

```
import random
import statistics

data_points = [ random.randint(1, 100) for x in range(1,1001) ]
statistics.mode(data_points)
# returns 94

data_points = [ random.randint(1, 100) for x in range(1,1001) ]
statistics.mode(data_points)
# returns 49

data_points = [ random.randint(1, 100) for x in range(1,1001) ]
statistics.mode(data_points)
# returns 32

mode(["cat", "dog", "dog", "cat", "monkey", "monkey", "dog"])
# returns 'dog'
```

Функция median() возвращает медианное значение заданных числовых данных путем вычисления среднего из двух средних точек, если это необходимо. Если количество точек данных нечетное, функция возвращает среднюю точку. Если число точек данных четное, оно возвращает среднее значение двух медианных значений.

В случае четного количества точек можно использовать median_low() или median_high() для вычисления медианы. При четном числе точек данных эти функции будут возвращать меньшее и большее значение двух средних точек соответственно.

```
import random
import statistics

data_points = [ random.randint(1, 100) for x in range(1,50) ]
statistics.median(data_points)
# returns 53

data_points = [ random.randint(1, 100) for x in range(1,51) ]
statistics.median(data_points)
# returns 51.0

data_points = [ random.randint(1, 100) for x in range(1,51) ]
statistics.median(data_points)
# returns 49.0

data_points = [ random.randint(1, 100) for x in range(1,51) ]
statistics.median_low(data_points)
# returns 50

statistics.median_high(data_points)
# returns 52

statistics.median(data_points)
# returns 51.0
```

Библиотека Pandas

Программная библиотека на языке Python для обработки и анализа данных. Работа pandas с данными строится поверх библиотеки NumPy, являющейся инструментом более низкого уровня.

Основная область применения — обеспечение работы в рамках среды Python не только для сбора и очистки данных, но для задач анализа и моделирования данных, без переключения на более специфичные для статобработки языки (такие, как R и Octave). В основе Pandas лежит DataFrame. DataFrame – двумерное (табличное) представление данных.

Ниже представлен пример подключения библиотеки и считывания файла .csv.

```
import pandas as pd
data = pd.read_csv('athlete_events.csv')
```

Обращение к библиотеке будет происходить через созданный объект pd. Используемый метод read_csv считывает указанный файл и возвращает DataFrame, который в дальнейшем будет обрабатываться. Например, data.head() позволит просмотреть первые пять записей.

Метод .shape позволяет получить количество строк и столбцов в DataFrame. Для получения отдельных данных по столбцам возможно в виде объекта Series. Кроме этого используя список (list) необходимых столбцов, можно получить новый датафрейм: `data[['Name', 'City', 'Sport']]`.

Извлечение строк также возможно с помощью метод iloc по индексу строки. На рисунке 1 представлен пример полученных данных. Для получения нескольких строк используется диапазон ([100:500]). Методы извлечения строк и столбцов могут быть использованы совместно.

```
>>> data.iloc[100]
ID                                      36
Name                    Stefan Remco Aartsen
Sex                                      M
Age                                     21
Height                                 194
Weight                                  78
Team                           Netherlands
NOC                                    NED
Games                          1996 Summer
Year                                  1996
Season                              Summer
City                               Atlanta
Sport                             Swimming
Event     Swimming Men's 100 metres Butterfly
Medal                                  NaN
Name: 100, dtype: object
```

Рисунок 1 – Получение данных по индексу строки

Метод .discribe() предназначен для получения описательной статистики:

- count – количество записей,
- mean – среднее арифметическое,
- std – стандартное отклонение,
- min – минимальное значение,
- n-ый (25, 50, 75) % – n-ый квартиль,
- max – максимальное значение.

Библиотека SciPy

SciPy — библиотека для языка программирования Python с открытым исходным кодом, предназначенная для выполнения научных и инженерных расчётов. Основные возможности:

- поиск минимумов и максимумов функций;
- вычисление интегралов функций;
- поддержка специальных функций;
- обработка сигналов;
- обработка изображений;
- работа с генетическими алгоритмами;
- решение обыкновенных дифференциальных уравнений;
- и др.

Основной структурой данных в SciPy является многомерный массив, реализованный модулем NumPy.

Рассмотрим пример применения данной библиотеки. Нормальный закон распределения является простым и удобным для дальнейшего исследования. Чтобы проверить имеет ли тот или иной атрибут нормальное распределение, можно воспользоваться двумя критериями Python-библиотеки scipy с модулем stats. Модуль scipy.stats поддерживает большой диапазон статистических функций, полный перечень которых представлен в официальной документации.

В основе проверки на “нормальность” лежит проверка гипотез. Нулевая гипотеза — данные распределены нормально, альтернативная гипотеза — данные не имеют нормального распределения.

```
import scipy
stat, p = scipy.stats.shapiro(data[‘Rings’]) # тест Шапиро-Уилк
print(‘Statistics=%.3f, p-value=%.3f’ % (stat, p))
alpha = 0.05
if p > alpha:
print(‘Принять гипотезу о нормальности’)
else:
print(‘Отклонить гипотезу о нормальности’)
```

Второй тест по критерию согласия Пирсона, который тоже возвращает соответствующее значение статистики и p-значение:

```
stat, p = scipy.stats.normaltest(data['Length'])
# Критерий согласия Пирсона
print('Statistics=%.3f, p-value=%.3f' % (stat, p))
alpha = 0.05
if p > alpha:
print('Принять гипотезу о нормальности')
else:
print('Отклонить гипотезу о нормальности')
```

T-тест (или тест Стьюдента) решает задачу доказательства наличия различий средних значений количественной переменной в случае, когда имеются лишь две сравниваемые группы. Модуль stats Python-библиотеки scipy также предоставляет t-тест.

Здесь имеется функция ttest_ind, вычисляющую t-тест двух независимых выборок. Для зависимых выборок используется функция ttest_rel.

```
half = len(data['Length']) / 2
sam1 = data.loc[:half, 'Length']
sam2 = data.loc[half:, 'Length']
scipy.stats.ttest_ind(sam2, sam1)
```

В модуле stats можно посмотреть табличное значение благодаря функции t.ppf. Она принимает в качестве аргументов соответствующие квартили (с доверительным интервалом 95% они будет равняться 0.975 или 0.025, так как это двусторонний тест) и суммарную степень свободы – сумма степеней свободы выборок.

```
dfs = (half - 1) + (half - 1)
scipy.stats.t.ppf(0.975, dfs).
```

Линейная регрессия в STATSMODEL

Python имеет библиотеку statsmodel, предоставляющую классы и функции для оценки статистических моделей.

Линейную регрессию можно построить с помощью метода наименьших квадратов. В statsmodel есть API, который дает возможность писать в R-стиле.

Метод ols принимает в качестве аргументов формулу для вычислений и DataFrame. Метод summary вернет результат вычисленной модели (рисунок 2).

```
import statsmodels.formula.api as smf
model = smf.ols('Rings ~ Diameter', data=data)
res = model.fit()
res.summary()
```

OLS Regression Results

Dep. Variable:	Rings	**R-squared:**	0.330
Model:	OLS	**Adj. R-squared:**	0.330
Method:	Least Squares	**F-statistic:**	2059.
Date:	Fri, 26 Jun 2020	**Prob (F-statistic):**	0.00
Time:	16:34:53	**Log-Likelihood:**	-9979.2
No. Observations:	4177	**AIC:**	1.996e+04
Df Residuals:	4175	**BIC:**	1.998e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	2.3186	0.173	13.423	0.000	1.980	2.657
Diameter	18.6699	0.411	45.371	0.000	17.863	19.477

Omnibus:	1414.851	**Durbin-Watson:**	0.967
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4737.552
Skew:	1.714	**Prob(JB):**	0.00
Kurtosis:	6.933	**Cond. No.**	11.8

Рисунок 2 – Результаты линейной регресии

Лабораторная работа 1. Алгоритмы обработки данных

Цель: получить навыки использования стандартной библиотеки для получения статистических показателей

Задание:

1. Реализовать приложение, вычисляющее средние значения (среднее, моду, медиану) для набора числовых данных. Числовые данные могут вводиться пользователем с клавиатуры или считываться из внешнего файла. Вычисление реализовать с использованием арифметических операций.
2. Провести анализ функций вычисления медианы стандартной библиотеки Statictica. В качестве входных данных использовать массивы с различным количеством входных данных. Сравнить поведение функций для массивов с четным и нечетным количеством точек.
3. Определить средние значение для массива нечисловых данных с использованием стандартной библиотеки.

4. Входные данные представляют собой информацию о доходах домохозяйств за последние два года. Используя функции стандартной библиотеки языка Python рассчитайте средние показатели по различным группировкам данных – за год, за месяц, в зависимости от размера домохозяйства и т.д.
5. Проанализировать временной ряд. Найти все показатели описательной статистики.

Лабораторная работа 2. Библиотека Pandas языка Python

Цель: получить навыки подключения и использования внешних библиотек для вычисления описательной статистики

Задание:

1. Задайте результаты статистического исследования в виде двумерного представления данных.
2. Определите несколько выборок данных по различным критериям.
3. Определите стандартную описательную статистику, используя методы библиотеки Pandas.
4. Создайте двумерное представление, имеющее не заполненные данные. Проведите обработку массива для получения новой выборки без пропущенных значений.

Лабораторная работа 3. Библиотека SciPy

Цель: получить навыки подключения и использования внешних библиотек для проверки гипотез

Задание:

1. Подключите библиотеку SciPy
2. Проверьте соответствие данных нормальному закону распределения.
3. Выделите два набора данных. Проверьте их зависимость по статистике Стьюдента.

Лабораторная работа 4. Графическое представление статистики

Цель: получить навыки использования графических объектов и библиотек

Задание:

1. Подключите библиотеку Mathplotlib

2. Постройте графические объекты – графики и гистограммы – для иллюстрации статистических расчетов, проведенных в предыдущих практических работах.