КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ

ИНСТИТУТ ЭКОЛОГИИ, БИОТЕХНОЛОГИИ И ПРИРОДОПОЛЬЗОВАНИЯ

Кафедра моделирования экологических систем

ЭЛЕМЕНТАРНОЕ ВВЕДЕНИЕ В PYTHON

Учебно-методическое пособие

Казань – 2025

УДК 51-7

Принято на заседании учебно-методической комиссии
Института экологии, биотехнологии и природопользования
Протокол № 9 от 17 января 2025 года

Авторы-составители:
доктор физико-математических наук, профессор **Зарипов Ш.Х.**;
кандидат физико-математических наук, доцент **Костерина Е.А.**;
кандидат физико-математических наук, доцент **Гильфанов А.К.**;
кандидат физико-математических наук, доцент **Никоненкова Т.В.**;
ассистент **Латыйпова С.Е.**

Рецензент:
доктор биологических наук, профессор **Савельев А.А.**

Элементарное введение в Python: учебно-методическое пособие / Ш.Х. Зарипов, Е.А. Костерина, А.К. Гильфанов, Т.В. Никоненкова, С.Е. Латыйпова. – Казань: Казанский федеральный университет, 2025. – 29 с.

Учебно-методическое пособие представляет собой с элементарное введение в язык Python и содержит описание базовых команд. Даны примеры решения простейших математических задач. Пособие рекомендовано для студентов бакалавриата и магистратуры по направлениям подготовки «Экология и природопользование», «Землеустройство и кадастры», «Биотехнология», «Гидрометеорология», «Почвоведение».

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Высокоуровневый язык программирования Python общего назначения реализован практически во всех операционных системах, и большинство его модулей распространяется бесплатно. Язык программирования Python обладает понятным синтаксисом и хорошо подходит для программирования математических вычислений и решения математических задач, в том числе анализа данных. Стандартные библиотеки включают большой объем математических функций. Кроме того, для Python написано большое количество прикладных библиотек, в том числе для научных расчетов, которые позволяют решать ряд математических задач без необходимости самостоятельной разработки алгоритмов. Программную среду для Python можно установить с сайта разработчиков https://www.python.org/downloads/. Необходимо выбрать желаемую версию Python и скачать инсталляционный пакет, соответствующий вашей операционной системе. После запуска скачанного файла и установки Python, необходимо также установить дополнительные библиотеки. Другой способ установить Python состоит в использовании бесплатного дистрибутива Anaconda (https://www.continuum.io/downloads). Это самый простой способ установить сразу Python и стандартные библиотеки. Кроме всего прочего, вы получите интегрированные оболочки Jupyter Notebook и Spyder, предназначенные для разработки и выполнения программ. С основными принципами работы в среде Python и командами можно ознакомиться, например, в [1-7].

ТИПЫ ДАННЫХ

Числовые данные в Python могут быть представлены двумя способами, которым соответствуют два типа данных. Целые числа (integer) – положительные и отрицательные целые числа, а также 0 (например, 1, 223, -36, 0). Числа с плавающей точкой (float point) – дробные числа (например, 3.32, -5.4321, 0.00111). Разделителем целой и дробной части числа служит точка. Наряду с числовыми данными Python оперирует символьными данными типа string. Строки (string) – набор различных символов, заключенных в кавычки (например, "Kazan", "Yes", 'loading', '656', 'a12345'). Кавычки в Python могут быть одинарными или двойными.

ОРГАНИЗАЦИЯ ВЫЧИСЛЕНИЙ И ВЫПОЛНЕНИЕ КОМАНД

Написание программ в Python может быть выполнено в среде разработки (Spyder) или в режиме интерактивного блокнота (Jupyter). В первом случае сначала пишется программа, а затем получается результат работы программы. Во втором случае программа пишется в режиме «вопрос-ответ», когда разработчик видит результат работы каждого блока. Эти блоки по-английски называются ячейками (Cell). Далее будем предполагать использование среды Jupyter. Запуск на выполнение последовательности команд в активной ячейке в среде Jupyter происходит при нажатии комбинации клавиш "Shift+Enter" или "Ctrl+Enter". Вводимая команда и ответ маркируются символами In[n]: и Out[n]: Эти обозначения представляют начальные буквы слов Input и Output. В квадратных скобках пишется номер команды n. Маркировка команд и их номеров осуществляется автоматически.

Запишем в строке ввода команду 2+2. Получим ответ 4.

In[n]:	`2+2`
Out[n]:	`4`

Вычислим сумму 1/3 + 3/7 и разность дробей 11/3-3/7.

In[n]:	`1/3+3/7`
Out[n]:	`0.7619047619047619`

In[n]:	`11/3-3/7`
Out[n]:	`3.238095238095238`

Для ввода чисел, заданных в экспоненциальной форме, используется буква "e". Найдем произведение чисел $3.4 \cdot 10^{-5}$ и $6.7 \cdot 10^{8}$.

In[n]:	`3.4e-5*6.7e8`
Out[n]:	`22780.0`

Для четырёх основных арифметических операций – сложения, вычитания, умножения и деления – используются символы "+", "-", "*", "/". Для задания приоритета операций используются круглые скобки. Так, для того чтобы вычислить дробь $6(3-4)/(7-3)$, надо использовать следующую команду

In[n]:	`6*(3-4)/(7-3)`
Out[n]:	`-1.5`

Для возведения в степень используется символ ** (две звездочки). Для того чтобы вычислить 2^{10}, 5^{-2}, $27^{1/3}$, следует писать команды

In[n]:	`2**10`
Out[n]:	`1024`

In[n]:	`5**(-2)`
Out[n]:	`0.04`

In[n]:	`27**(1/3)`
Out[n]:	`3.0`

Еще две основные операции – это целочисленное деление // (две наклонные черты) и остаток от деления % (обозначается знаком процента).

In[n]:	`5//2`
Out[n]:	`2`

In[n]:	`5%2`
Out[n]:	`1`

ОПРЕДЕЛЕНИЕ ПЕРЕМЕННЫХ

В Python можно определять переменные. Как известно, переменная – это именованная область памяти. Важно, что в Python имена переменных, имена функций, ключевые слова и другие идентификаторы чувствительны к регистру. Зададим x=5 и y=10 и найдем сумму и произведение x и y:

In[n]:	`x=5`
	`y=10`
	`z1=x+y`
	`z2=x*y`
	`print(z1, z2)`

Out[n]:	`15 50`

Задания.

1. Вычислить $\frac{\sqrt{25}+1}{8^{\frac{2}{3}}-1}$. Ответ 2.

2. Задать $a=2$, $b=a+1/a$, $c=b^a$. Найти сумму $a+b+c$. Ответ 43/4.

ЭЛЕМЕНТЫ ПРОГРАММИРОВАНИЯ

Логические выражения

Наряду с числовыми и символьными переменными в Python вводится также логический тип данных. Данные этого типа могут быть представлены двумя значениями: True (истина, константа 1) и False (ложь, константа 0). С помощью логического типа данных можно выразить значение логических выражений и/или результат логических операций.

Логические операторы

Логические операции – это операторы сравнения:

>	больше
<	меньше
>=	больше или равно
<=	меньше или равно
==	равно
!=	не равно

и логические операторы and, or, not.

Примеры логических условий

`x == 8`	x равен 8
`x != 2`	x не равен 2
`x > 15`	x больше 15
`x < 52`	x меньше 52
`x >= 16`	x больше или равен 16
`x <= 43`	x меньше или равен 43

Замечание. Один знак равенства соответствует операции присваивания, например, x = 103+3, а двойной знак равенства – операции сравнения, например, x==4.

Два и более простых логических выражения могут быть объединены в единое логическое выражение с помощью логических операций "**and**" (и) и "**or**" (или). В этом случае при использовании оператора **and** значение логического выражения **True (истина)** достигается, если истинны результаты обоих простых выражений, которые связывает данный оператор. Если хотя бы одно из этих простых выражений будет иметь результат **False** (ложь), то и все сложное выражение будет ложным. При использовании оператора **or** значение **True** достигается, если будет истинным результат хотя бы одного простого выражения, входящего в состав сложного. Сложное выражение с оператором **or** становится ложным лишь тогда, когда ложны все его составляющие.

Примеры сложных логических условий

`x == 2 and y < 3`	x равен 2 и y меньше 3
`x > 4 and y < 5`	x больше 4 и y меньше 5
`x != -1 or y < 1`	x не равен -1 или y меньше 1
`x < 1 or y > 0`	x меньше 1 или y больше 0

При использовании логических операций в одном выражении следует иметь в виду, что по приоритету сначала вычисляется not, затем and и в последнюю очередь – or. Управлять приоритетом можно с помощью скобок.

`x == 1 or y < 2 and z > 3`	сначала будет вычислена операция and, а затем or
`(x == 1 or y < 2) and z > 3`	сначала будет вычислена операция or, а затем and

Задания.

1. Присвойте переменным *x* и *y* произвольные числовые значения. Составьте сложные логических выражения с помощью оператора **and**, два из которых должны давать истину, а два других – ложь.

2. Выполните задание 1 с логическим выражением с помощью оператора **or**.

Условный оператор if

Условный оператор if позволяет выполнять различные действия в зависимости от выполнения тех или иных условий. Синтаксис оператора имеет следующий вид:

```
if условие:
    действия, выполняемые в случае результата True в условии
else:
    действия, выполняемые в случае результата False в условии
```

Конструкцию **else** можно пропустить в написании. В условии используются операторы сравнения и логические операторы.

Важную роль в синтаксисе языка Python играют отступы. Отступы делаются с помощью четырех пробелов или по нажатию клавиши Tab. Отступами выделяются блоки операторов – последовательности операторов, которые должны выполняться друг за другом. В Python каждый блок операторов должен быть записан по одной вертикальной линии отступов. Такую же роль в языке программирования Pascal играют ключевые слова begin ... end, а в языке C фигурные скобки {}.

Пример.

С использованием оператора **if**, присвоим переменной *c* наибольшее из значений переменных *a* = 10 и *b* = 2:

In[n]:	a=10
	b=2
	if a>b:
	    c=a
	else:
	    c=b
	print(c)

Out[n]:	10

Операторы цикла for и while

Для формирования списков и массивов используются операторы цикла for и while.

Синтаксис оператора **while** имеет вид:

```
while условие:
    операторы, выполняемые в цикле
```

Пример.

Вывести на экран числа, квадраты которых меньше 50:

In[n]:	i = 1
	while i**2 < 50:
	    print(i)
	    i = i + 1

Out[n]:	1 2 3 4 5 6 7

Синтаксис оператора **for** имеет вид:

```
for имя индекса in range(начальное значение индекса, конечное значение индекса):
    операторы, выполняемые в цикле
```

Функция **range()** создает последовательность чисел внутри определенного диапазона и не используется как самостоятельная функция, а обычно применяется только для работы с циклом **for**.

- range(start, stop, step) создает последовательность из чисел от start до (stop - 1) с шагом step;
- range(start, stop) создает последовательность от start до (stop - 1) с шагом 1;
- range(stop) создает последовательность от 0 до (stop - 1) с шагом 1.

Примеры.

1. Вывести на экран сумму квадратов чисел от 1 до 10.

In[n]:	`sum = 0`
	`for i in range(1,10):`
	`    sum = sum+ i**2`
	`    print(sum)`

Out[n]:	`1`
	`5`
	`14`
	`30`
	`55`
	`91`
	`140`
	`204`
	`285`

2. Найти сумму чисел от 1 до 10.

In[n]:	`sum = 0`
	`n = 10`
	`for i in range(1, n):`
	`    sum = sum + i`
	`print(sum)`

Out[n]:	`45`

3. Вывести на экран квадраты чисел от 1 до 5.

In[n]:	`for i in range(6):` `    print(i**2)`

Out[n]:	`0`
	`1`
	`4`
	`9`
	`16`
	`25`

Задания.

1. Напишите программу, которая определяет, состоит ли заданное двузначное число из одинаковых цифр. Если состоит, то вывести «Да», в противном случае вывести «Нет».

2. Напишите программу, которая выводит на экран квадраты нечетных чисел от 1 до 10.

3. Вывести в столбик таблицу умножения на 10.

СПИСКИ

Списки являются последовательностями числовых значений и/или символьных данных, заключенных в квадратные скобки [] и отделенных друг от друга с помощью запятой:

```
a1 = [15, 55, -10, 33, -4]  # список целых чисел
a2 = [11.1, 51.31, 21.68, 14.61, 4.1, 1.85] # список дробных чисел
a3 = ["Moscow","Samara", "Ufa", "Kazan"]  # список из слов
a4 = ["Moscow ", "Kazan", 1222, 10,-1]  # смешанный список
a5 = [[0,1, 2], [3, 2, 1], [-1, -1, -1]] # список, состоящий из списков
a6 = ['l', 'i', ['st', 2]]  # список из значений и списка
```

Создание списков

1. Присваивание значений в списке

```
q = [0, 1, 2, 3, 10]  # список целых чисел
s = []                # пустой список
```

2. Список можно получить при помощи функции **list()**

In[n]:	`list('Kazan')`
Out[n]:	`['K', 'a', 'z', 'a', 'n']`

In[n]:	`list('12345')`
Out[n]:	`['1', '2', '3', '4', '5']`

3. Создание списка при помощи функции split() (расщепить)

In[n]:	`stroka ="Hi,Kazan"`
	`list1=stroka.split(",")`
	`list1`

Out[n]:	`['Hi', 'Kazan']`

4. Создание списка с помощью генератора элементов

Список формируется умножением одного элемента на число:

In[n]:	`s=[1]*10` `print(s)`
Out[n]:	`[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]`

Список формируется с помощью цикла

In[n]:	`n=10`
	`s = []`
	`for i in range(n):`
	`  s.append(i)`
	`print(s)`

Out[n]:	`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

Аналогичный список можно получить с помощью следующей конструкции

In[n]:	`s = [i for i in range(10)]` `print(s)`
Out[n]:	`[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

In[n]:	`s = [i**2 for i in range(10)]` `print(s)`
Out[n]:	`[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`

In[n]:	`s = [10-i for i in range(10)]` `print(s)`
Out[n]:	`[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]`

5. Списки можно складывать (конкатенировать) с помощью знака «+»:

In[n]:	`s=[1,2,3] + [4,5,6] + [10]` `print(s)`
Out[n]:	`[1, 2, 3, 4, 5, 6, 10]`

6. Формирование двумерного списка

In[n]:	m=2 n=5 s2d=[[i+j for j in range(m)] for i in range(n)] print(s2d)
Out[n]:	[[0, 1], [1, 2], [2, 3], [3, 4], [4, 5]]

Списки можно рассматривать как массивы с индексацией от 0. Элементы одномерного и двумерного списков могут быть получены как s[i] и s[i][j] соответственно, как показано ниже:

In[n]:	s=[1, 2, 3, 4, 5, 6, 10] print(s[0],s[4])
Out[n]:	1 5

In[n]:	s = [[0, 1], [1, 2], [2, 3], [3, 4], [4, 5]] print(s[0][1],s[3][0])
Out[n]:	1 3

Задания.

Создать список, используя генератор:

1. Заполнить список квадратами чисел от 1 до 10.

2. Заполнить список числами от 10 до 1.

3. Создайте список целых чисел от -5 до 25.

4. Создайте список целых чисел от -10 до 10 с шагом 2.

5. Создайте список из 20 пятерок.

6. Создайте список из сумм чисел от 0 до 10:

[0, 0+1, 0+1+2, …, 0+1+2+…+10].

МАССИВЫ NumPy

Списки удобны для хранения данных различного типа. Но их использование в качестве числовых массивов не всегда возможно. Например, умножение на 2 приводит к удвоению списка, а не удвоению его элементов:

In[n]:	mylist=[1,2,3,4] print(mylist) mylist2=[1,2,3,4]*2 print(mylist2) type(mylist)
Out[n]:	[1, 2, 3, 4] [1, 2, 3, 4, 1, 2, 3, 4] list

Для создания массивов используется библиотека NumPy, которая позволяет применять обширный набор высокоуровневых математических функций для операций с большими массивами чисел и вызывается командой

import numpy

Используя функции из библиотеки, нужно каждый раз указывать имя библиотеки, например, numpy.array(). Для сокращения записи можно в рамках своего кода дать библиотеке сокращенное имя и далее ссылаться на него. Будем давать библиотеке NumPy имя np, используя команду

import numpy as np

Список может быть преобразован в массив командой array()

In[n]:	import numpy as np mylist=[1,2,3,4] print(mylist) myarray=np.array(mylist) print(type(myarray)) myarray2=myarray*2 myarray2
Out[n]:	[1, 2, 3, 4] <class 'numpy.ndarray'> array([2, 4, 6, 8])

ВСТРОЕННЫЕ ФУНКЦИИ

Таблица стандартных встроенных функций

`abs()`	Абсолютное значение
`sum()`	Сумма элементов списка
`type()`	Определение типа переменной
`max()`	Максимальное значение в списке
`min()`	Минимальное значение в списке
`str()`	Перевод значения переменной в текстовое
`int()`	Целая часть переменной
`float()`	Представление переменой в виде числа с плавающей точкой
`len()`	Количество элементов в объекте (например, длина списка или строки)

Примеры.

In[n]:	`abs(-5)`
Out[n]:	`5`

In[n]:	`L = [1,5,30, 40, 55, 111]` `sum(L)`
Out[n]:	`242`

In[n]:	`type('Kazan')`
Out[n]:	`str`

In[n]:	`max([1,-2,10,112,2,5,70])`
Out[n]:	`112`

In[n]:	`S=[1,13,12,1,3,5,7]` `len(S)`
Out[n]:	`7`

Модуль (иначе говоря, библиотека) math включает в себя набор математических функций и вызывается командой

`import math`

Модуль math содержит математические константы – числа π и e.

In[n]:	`import math` `math.pi`
Out[n]:	`3.141592653589793`

In[n]:	`import math` `math.e`
Out[n]:	`2.718281828459045`

Таблица математических функций библиотеки math

Функция		Описание
`math.sqrt(x)`	$\sqrt{x}$	квадратный корень из x
`math.sin(x)`	$\sin x$	синус x (x указывается в радианах)
`math.cos(x)`	$\cos x$	косинус x (x указывается в радианах)
`math.tan(x)`	$\mathrm{tg}\, x$	тангенс x (x указывается в радианах)
`math.atan(x)`	arctg x	арктангенс x, возвращает значение от –π/2 до π/2
`math.atan2(y, x)`	arctg (y/x)	арктангенс отношения y и x, возвращает значение от –π до π
`math.fabs(x)`	$\lvert x \rvert$	модуль x
`math.factorial(x)`	$x!$	факториал числа x
`math.fmod(x, y)`		остаток от деления x на y
`math.exp(x)`	e^x	exp(x)
`math.log(x, [a])`	$\log_a x$	логарифм x по основанию a. Если a не указано, вычисляется натуральный логарифм
`math.degrees(x)`		переводит радианы в градусы
`math.radians(x)`		переводит градусы в радианы
`math.erf(x)`	$erf\ x = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$	функция ошибок

Примеры.

In[n]:	`import math` `math.sqrt(9)`
Out[n]:	`3`

In[n]:	`import math` `math.degrees(math.pi)`
Out[n]:	`180.0`

ОПРЕДЕЛЕНИЕ ФУНКЦИЙ

Наряду с встроенными функциями пользователь может задавать свои функции. **Функция** – это блок кода, который начинается с ключевого слова **def**, названия функции и двоеточия. **Функция** может принимать входные данные (аргументы или параметры), выполнять действия с ними и возвращать данные. Вызвать функцию – значит передать ей входные данные, необходимые для выполнения действий внутри тела функции и для возвращения результата выполнения функции. Функции в Python схожи с математическими функциями. Например, функция, выражающая квадрат числа $f(x) = x^2$, в Python имеет вид

```
def f(x):
    return x ** 2
```

Ключевое слово def определяет функцию. После def следует имя функции, которое должно отвечать тем же правилам, что и имена переменных: в имени функции нельзя использовать заглавные буквы, а слова должны быть разделены подчеркиванием. Для вызова функции после ее имени надо указать круглые скобки и поместить внутрь параметры, отделив каждый из них запятой. После скобок ставится двоеточие. Тело функции является блоком операторов и должно быть на одной вертикальной линии отступа. Ключевое слово return используется для определения значения, которое функция возвращает как результат своей работы. Для вывода результата на экран применим функцию print.

Примеры.

In[n]:	`def function():`
	`  print("Задаем функцию")`
	`function()`

Out[n]:	`Задаем функцию`

In[n]:	`def f(x):`
	`  return x * 2`
	`y = f(5)`
	`print(y)`

Out[n]:	10

In[n]:	def g(x, y, z): return x**2 + y-2*z a = g(1, 2, 3) print(a)
Out[n]:	-3

Задания.

1. Найти значение выражения $\frac{\pi^2}{1+\sqrt{e-1}}$ в десятичной форме. Ответ 4.2710.

2. Присвоить функции $\frac{e^x-e^{-x}}{e^x+e^{-x}}$ имя th и вычислить значение этой функции при а) *x=1*, б) *x=ln(2)*, с) *x=–4*. Ответы *а)* 0.7616*, б)*0.6*, в)* –0.9993.

ПОСТРОЕНИЕ ГРАФИКОВ

Для построения графиков может быть использована библиотека Matplotlib. Библиотека имеет широкие возможности для визуализации данных двумерной и трехмерной графики, построения различных диаграмм. Наилучшим вариантом использования библиотеки является выбор подходящего графика из галереи с официального сайта и его адаптация под свои задачи.

Библиотека Matplotlib вызывается командой

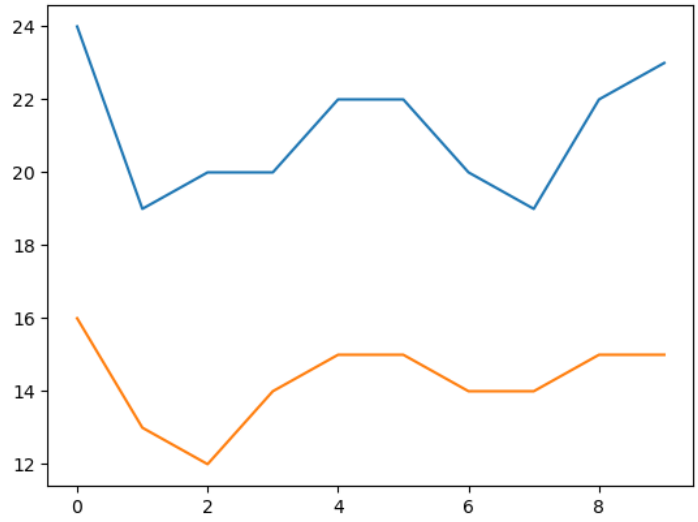
```
import matplotlib.pyplot as plt
```

График функции, заданной набором точек

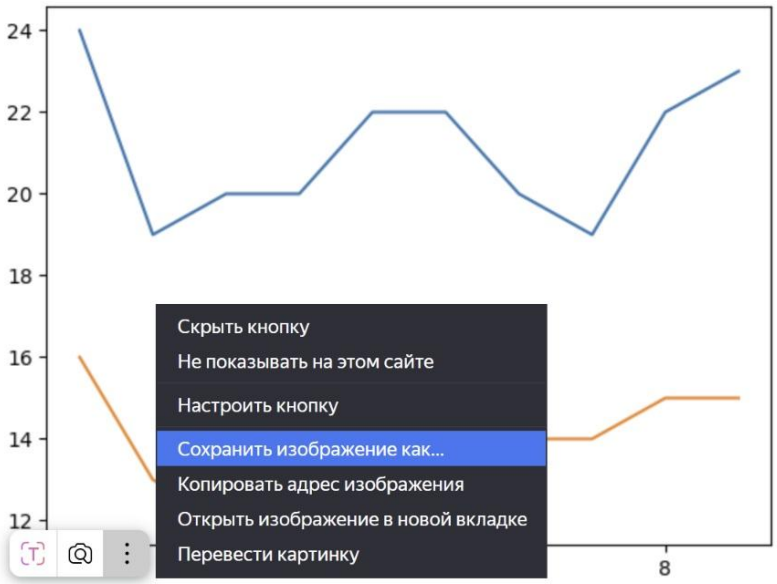
Для визуализации наборов точек data_d=[24,19,20,20,22,22,20,19,22,23] и data_n=[16,13,12,14,15,15,14,14,12,12], соответствующих прогнозу дневной и ночной температуры на 10 дней в июле 2024 в г. Казани, используем команду plt.plot(data)

In[n]:
```
import matplotlib.pyplot as plt
data_d=[24,19,20,20,22,22,20,19,22,23]
data_n=[16,13, 12, 14, 15, 15,14,14,15,15]
plt.plot(data_d)
plt.plot(data_n)
plt.show()
```

Out[n]:



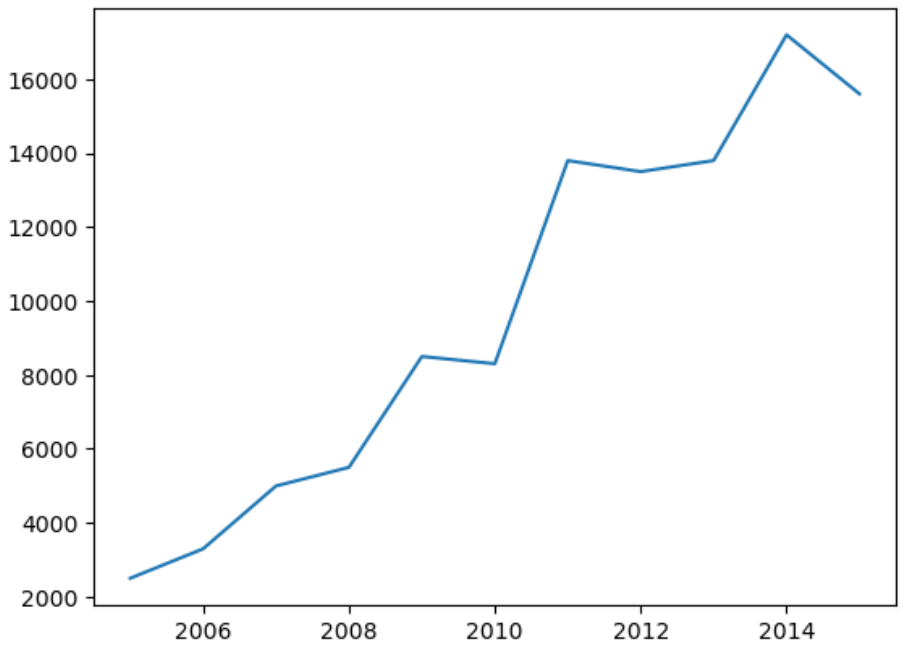
Функция plot() строит график линии, а функция show() показывает его на экране. На графике появится ломаная линия, в которой номерам элементов данных data_d и data_n [0,1,2,3,4,5,6,7,8,9] будут соответствовать их значения. Рисунок можно сохранить в одном из графических форматов, например *.png, с помощью крайней правой иконки (три точки) на панели в левом нижнем углу графика:



Для визуализации численности бобров в Республике Татарстан с 2005 по 2015 год используем списки:

`data_y=[2502,3298,4998,5501,8500,8303,13797,13498,13797,17197, 15599],data_t=[2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015]` и программу:

In[n]:	`import matplotlib.pyplot as plt`
	`data_y=[2502,3298,4998,5501,8500,8303,`
	`13797,13498,13797,17197,15599]`
	`data_t=[2005,2006,2007,2008,2009,2010,`
	`2011,2012,2013,2014,2015]`
	`plt.plot(data_t,data_y)`
	`plt.show()`

Out[n]:	

Набор точек численности бобров может быть представлен в виде отдельных точек с помощью команды plt.scatter(data_t,data_y)

In[n]:	`import matplotlib.pyplot as plt`
	`data_y=[2502,3298,4998,5501,8500,8303,`
	`3797,13498,13797,17197,15599]`
	`data_t=[2005,2006,2007,2008,2009,2010,`
	`2011,2012,2013,2014,2015]`
	`plt.scatter(data_t,data_y)`
	`plt.show()`

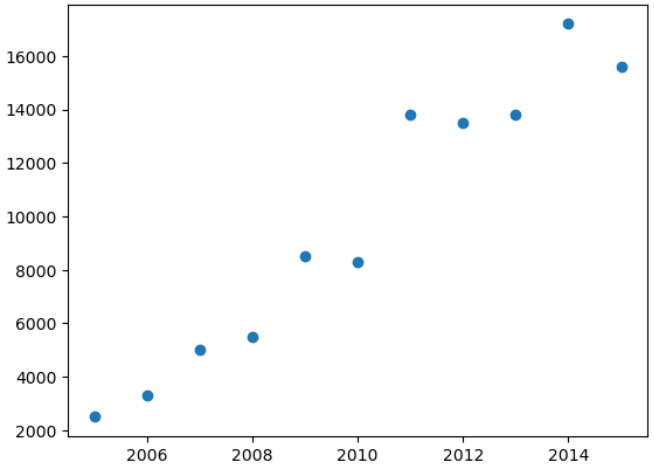
Out[n]:	

График функции, заданной формулой

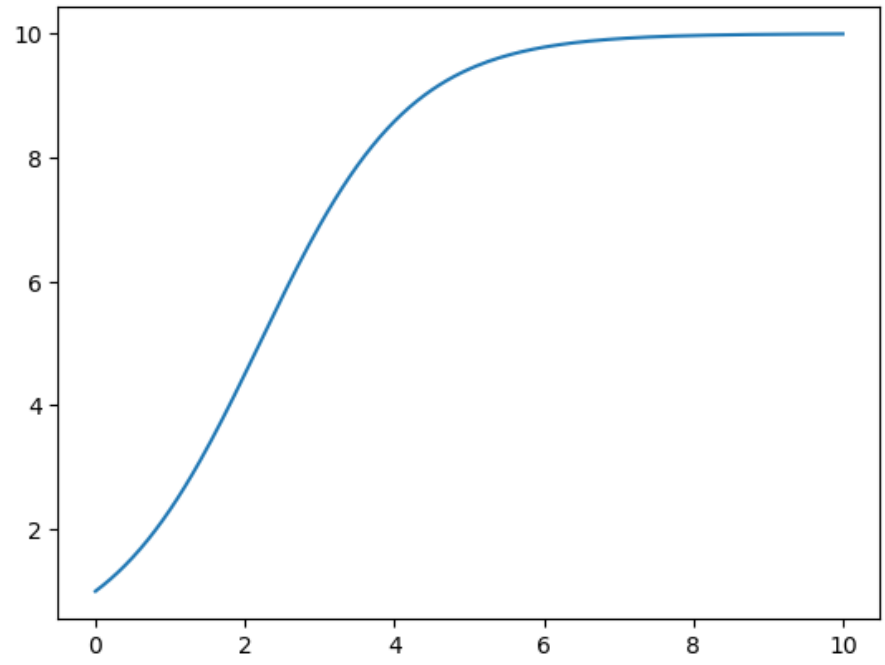
Для построения функции одной переменной надо задать эту функцию, с помощью команды linspace библиотеки NumPy создать список точек, в которых функция будет показана, и построить график функции с помощью команды plt.plot(t,y).

Пример.

Пример программы построения графика зависимости численности популяции от времени (r – коэффициент прироста, $к$ – емкость среды, x_0 – начальная численность)

$$x(t) = \frac{k}{1 + \frac{e^{-rt}(k - x_0)}{x_0}}.$$

In[n]:	`import numpy as np`
	`import matplotlib.pyplot as plt`
	`k=10`
	`r=1`
	`x0=1`
	`def x(t):  # x(t)=k/(1+e-rt (k-x0)/ x0)`
	`  return k/(1+np.exp(-r*t)*(k-x0)/x0)`
	`t = np.linspace(0, 10, 201)  # 101 точка между 0 и 5`
	`y = x(t)`
	`plt.plot(t, y)`
	`plt.show()`

Out[n]:	

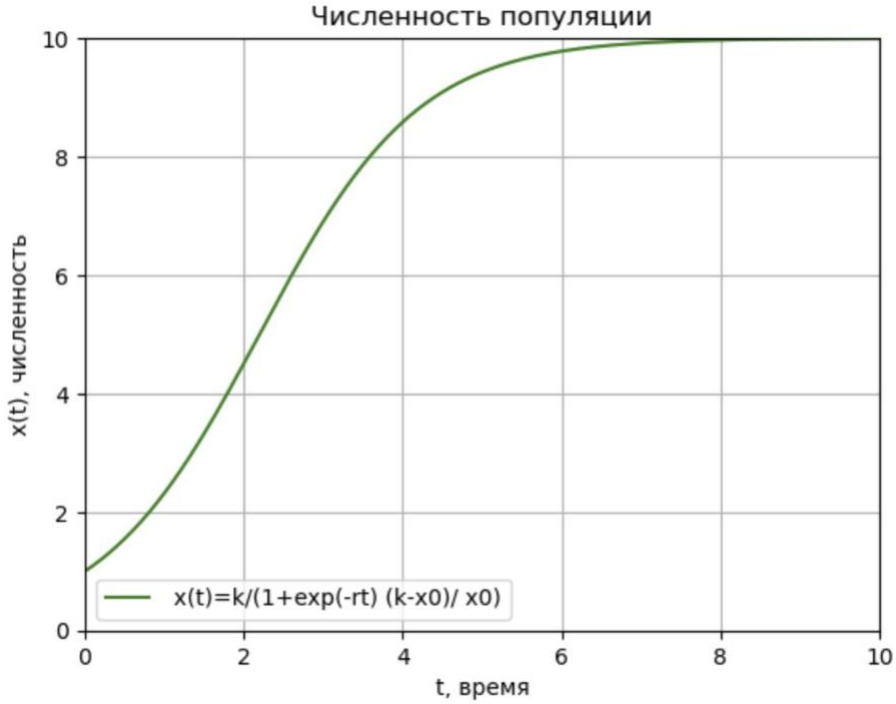
Оформление графика и сохранение рисунка в файл

Добавим к построению кривой название графика, обозначение осей, легенду. Изменим вид кривой и пределы ее просмотра. Можно также сохранять файлы с рисунком, включив в код команду для автоматического сохранения результатов в файле.

In[n]:	

```
import numpy as np
import matplotlib.pyplot as plt
import os

#os.chdir("C:\\Python") # куда сохранить файл с графи-
ком
k=10
r=1
x0=1
def x(t): #  x(t)=k/(1+exp(-rt)*(k-x0)/ x0)
   return k/(1+np.exp(-r*t)*(k-x0)/x0)
t = np.linspace(0, 10, 201)  # 201 точка от 0 до 10
n=len(t)
y = []
for i in range(n):
    y.append(x(t[i]))
plt.plot(t, y,'g', label=' x(t)=k/(1+exp(-rt) (k-x0)/
x0)')
plt.axis([0,10,0,10])  # задание [xmin, xmax, ymin,
ymax]
plt.xlabel('t, время')    # обозначение оси абсцисс
plt.ylabel('x(t), численность') # обозначение оси орди-
нат
plt.title('Численность популяции')  # название графика
plt.legend()       # вставка легенды (текста в label)
plt.grid()
plt.savefig('Рис.1.png', dpi=200)
plt.show()
```

Out[n]:	

Пример 1.

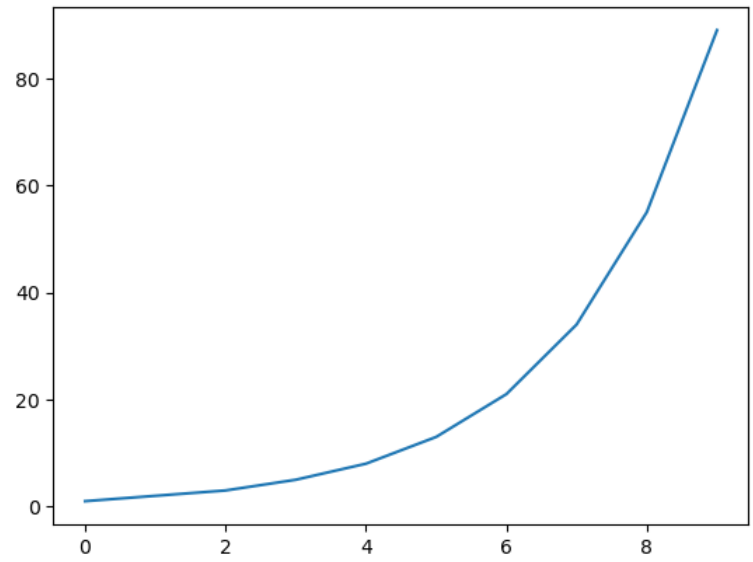
Вычислить последовательность чисел Фибоначчи, представляющую динамику численности биологической популяции по формуле

$$x_{n+2} = x_{n+1} + x_n$$

и представить в графической форме.

In[n]:	`# вычисление ряда чисел Фибоначчи`
	`import matplotlib.pyplot as plt`
	`a=1`
	`b=1`
	`n=10`
	`x =[1 for i in range(10)]`
	`for i in range(1, n):`
	`  c=a+b`
	`  a=b`
	`  b=c`
	`  x[i]=c`
	`print(x)`
	`plt.xlabel('i')  # обозначение оси абсцисс`
	`plt.ylabel('xi, численность популяции')  # обозначение оси ординат`
	`plt.plot(x)`
	`plt.show()`

Out[n]:	`[1, 2, 3, 5, 8, 13, 21, 34, 55, 89]`

Пример 2.

Найти площадь S полигона, вершины которого задаются координатами x=(3,5,12,9,5,3) и y=(4,11,8,5,6,4), по формуле землемера:

$$S=S=\frac{1}{2}\left|\sum_{i=1}^{i=n-1} x_i y_{i+1}+x_i y_1-\sum_{i=1}^{i=n-1} x_{i+1} y_i-x_i y_n\right|=$$

$$\frac{1}{2}\left|x_1 y_2+x_2 y_3+\cdots+x_{n-1} y_n+x_n y_1-x_2 y_1-x_3 y_2-\cdots-x_n y_{n-1}-x_1 y_n\right|$$

In[n]:	`# расчет площади полигона по формуле землемера`
	`import matplotlib.pyplot as plt`
	
	`# Координаты полигона`
	`x=[1,2,3,4,5,1]`
	`y=[4,11,10,9,6,4]`
	`xy1=[x[i]*y[i+1] for i in range(5)]`
	`xy2=[x[i+1]*y[i] for i in range(5)]`
	`s=0.5*abs(sum(xy1)-sum(xy2))`
	`print ('Площадь полигона=',s)`
	
	`plt.figure(figsize=(8,5))`
	`plt.title("Полигон",fontsize=20)`
	`plt.plot(x,y,'-',linewidth=2)`
	`plt.xlabel("x",fontsize=20)`
	`plt.ylabel("y",fontsize=20)`
	`plt.tick_params(axis='both', which='major',`
	`labesize=15)`
	`plt.grid()`
	`plt.show()`

Out[n]:	`Площадь полигона= 15.0`

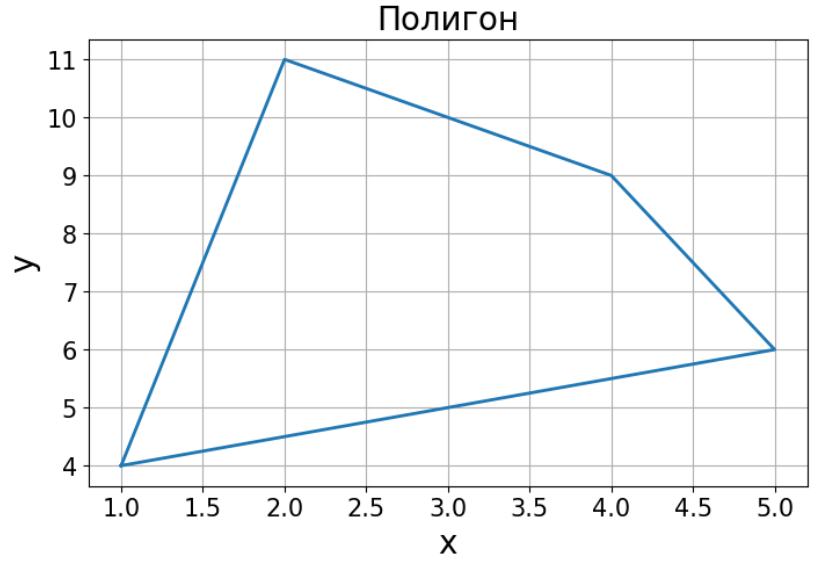


График функции двух переменных

В экологии одной из важных задач является исследование функции двух переменных, к которым относятся участок земной поверхности в виде модели рельефа, двумерное распределение природных и экологических факторов по пространству. При этом наиболее информативным является представление функции двух переменных в виде изолинии (изотермы, изобары, изохоры и т.п.).

Пример 1.

Построить поверхность – изображение функции Экли и изолинии этой поверхности в диапазоне x= [-3,3], y=[-3,3]:

$$f(x,y) = -20\exp\left(-0.2\sqrt{0.5(x^2+y^2)}\right) - \exp\left[0.5(\cos(2\pi x) + \cos(2\pi y)\right] + e + 20$$

In[n]:

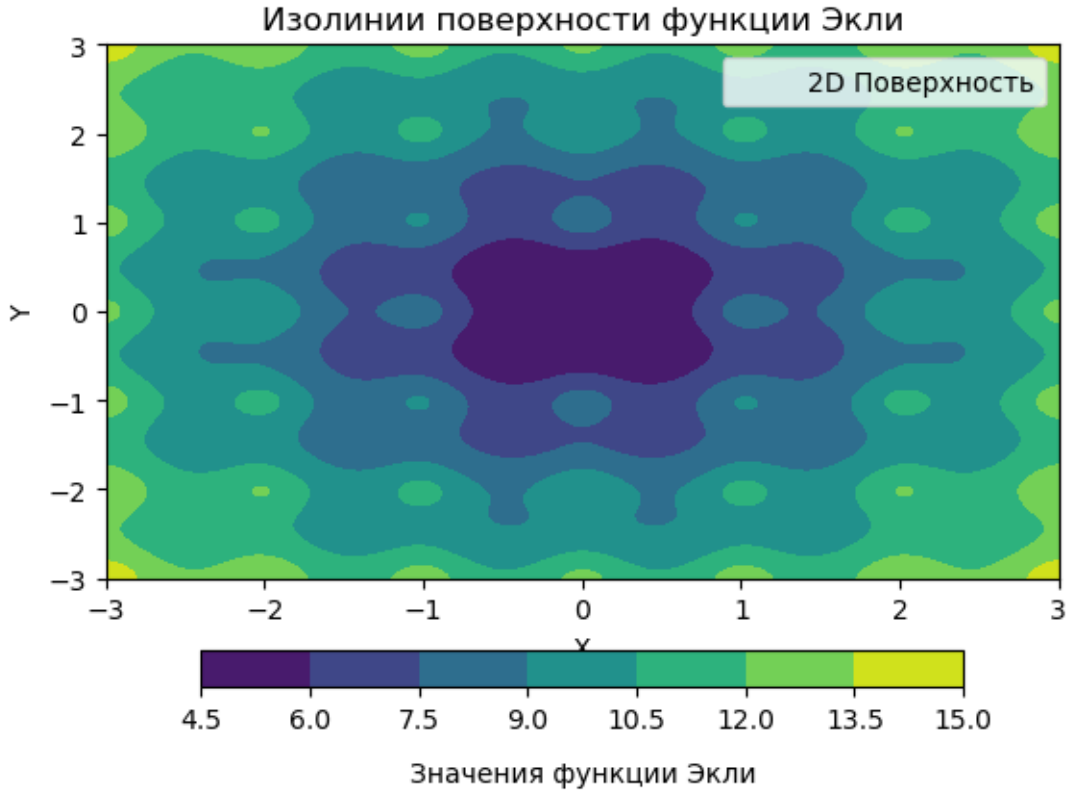
```
# Изолинии и поверхность функции Экли
import math
import numpy as np
import matplotlib.pyplot as plt
mpi=math.pi
me=math.e
# задаем функцию Экли
def f(x,y):
    return -20*np.exp(-
0.2*np.sqrt(0.5*(x**2+y**2)))+np.exp(0.5*(np.cos(2*mpi*x)+
np.cos(2*mpi*y)))+me+20
```

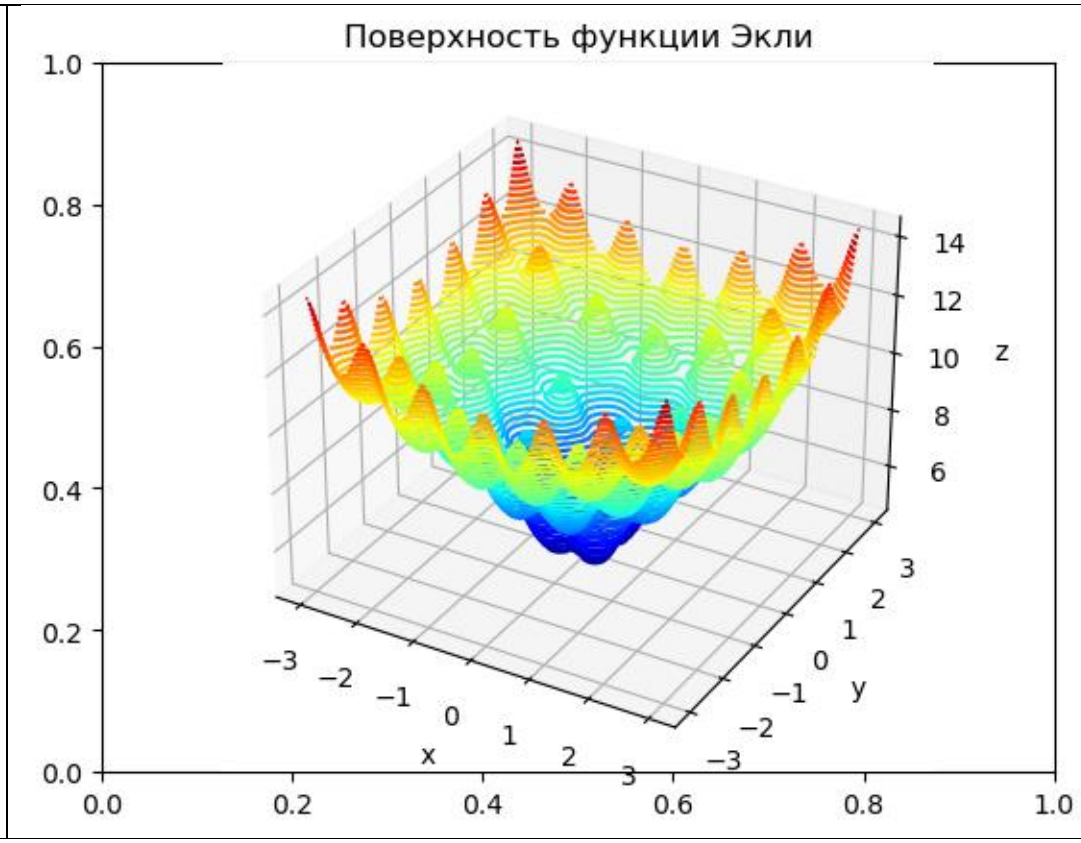
```
# создаем массив дискретных значений аргументов в диапа-
зоне x= # [-3,3], y=[-3,3].
x = np.linspace(-3, 3, 1000)
y = np.linspace(-3, 3, 1000)

# создаем двумерный массив дискретных значений аргументов
X,Y=np.meshgrid(x,y)
# вычисляем функцию Экли на введенном двумерном массиве
Z=f(X,Y)
# рисуем изолинии функции Экли
CF=plt.contourf(X,Y,Z)
# добавляем легенду, названия осей и графика
plt.legend(['2D Поверхность'], fontsize=10)
cbar = plt.colorbar(CF, orientation='horizontal', pad=0.1,
shrink=0.8)
cbar.set_label('Значения функции Экли', rotation=0, la-
belpad=10)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Изолинии поверхности функции Экли')
plt.show()

# рисуем поверхность, задаваемую функцией Экли
fig=plt.figure()
plt.title('Поверхность функции Экли')
ax=plt.axes(projection='3d')
ax.contour3D(X,Y,Z,50,cmap='jet')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
```

Out[n]:





Задания.

1. Нарисовать график функции *y(x)*: $y = -x^2 + 4x - 1$.

2. Нарисовать график функции *y(x)*

$$y = \begin{cases} x^3, & \text{если } x > 0 \\ |x|, & \text{если } x \leq 0 \end{cases}.$$

3. Нарисовать изолинии и поверхность функции Химмельблау $f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$ в диапазоне x= [-5,5], y=[-5,5].

ЛИТЕРАТУРА

1. Доля П.Г. Введение в научный Python – Харьков: Харьковский национальный ун-т, 2016. – 333 с.
2. Доусон М. Программируем на Python. – СПб.: Питер, 2014. – 416 с.
3. Лутц М. Изучаем Python. – СПб.: Символ-Плюс, 2011. – 1280 с.
4. Соболев А.Н., Воронцов А.Г. Компьютерная физика: учебное пособие. – Челябинск: Издательский центр ЮУрГУ, 2016. – 119 с.
5. Шапошникова С.В. Основы программирования на Python. – Лаборатория юного линуксоида, 2011. – 44 с.
6. Программирование и научные вычисления на языке Python [Электронный ресурс] – Режим доступа: https://ru.wikiversity.org/wiki/Программирование и научные вычисления на языке Python, свободный. – Дата обращения: 16.01.2025.
7. Зарипов Ш.Х., Никоненкова Т.В., Ложкин Г.И., Гильфанов А.К., Костерина Е.А. Математические модели динамики популяций: реализация на языке Python: учебное пособие. – Казань: Изд-во Казанского федерального ун-та, 2023. – 44 с.