Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информатика»

Т. А. Трохова

ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON

ПРАКТИКУМ
по выполнению лабораторных работ
для студентов специальности 1-40 04 01 «Информатика
и технологии программирования»
дневной формы обучения

Гомель 2025

УДК 681.3.06(075.8)
ББК 32.81я73
Т76

Рекомендовано научно-методическим советом
факультета автоматизированных и информационных систем
ГГТУ им. П. О. Сухого
(*протокол № 10 от 14.06.2023 г.*)

Рецензент: доц. каф. «Информационные технологии» ГГТУ им. П. О. Сухого
канд. техн. наук, доц. *В. В. Комраков*

Трохова, Т. А.

Т76 Язык программирования Python : практикум по выполнению лаборатор. работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» днев. формы обучения / Т. А. Трохова. – Гомель : ГГТУ им. П. О. Сухого, 2025. – 72 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

Содержит краткие теоретические сведения, включающие описание инструкций программирования, примеры использования и набор вариантов заданий для изучения языка программирования Python.

Для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения.

УДК 681.3.06(075.8)
ББК 32.81я73

Лабораторная работа № 1
Введение в программирование Python

Цель работы: получить навыки составления программ на Python с использованием базовых инструкций языка.

Краткие теоретические сведения
Структура программы

Программа, написанная на языке программирования Python, должна соответствовать определенным правилам:

– программа должна состоять из инструкций;

– конец строки является концом инструкции (точка с запятой не требуется);

– вложенные инструкции объединяются в блоки по величине отступов (рис. 1.1);

– отступ может быть любым, но в рамках блока – отступ должен быть одинаков;

– основная инструкция вложенного блока завершается двоеточием, вслед за которым располагается вложенный блок кода, с отступом под строкой основной инструкции;

– можно записать несколько инструкций в одной строке, разделяя их точкой с запятой;

– существует возможность разбить инструкцию, на несколько строк с использованием символа переноса строки **(\)**.

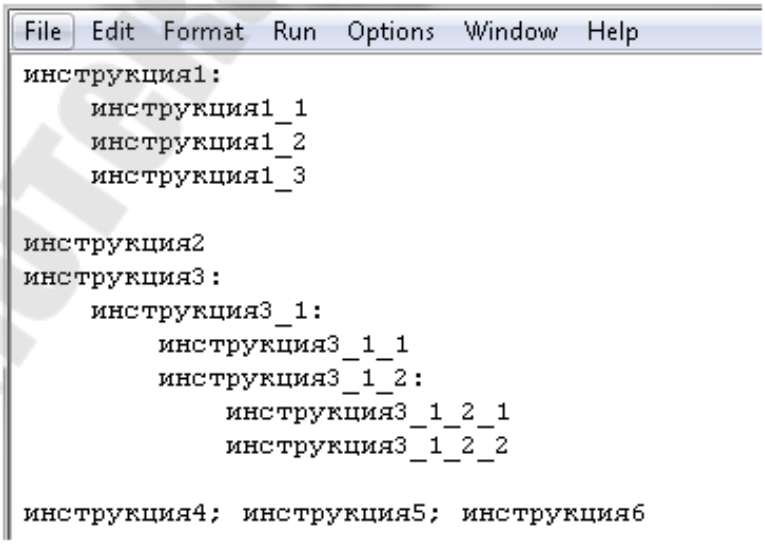


Рис. 1.1. Структура программы Python

Символ # используется для ввода комментариев в текст программы. Например:

L=2*pi*r # Вычисление длины окружности

#Программа табулирования функции

При выполнении программы транслятор комментарии игнорирует.

Данные, выражения

Алфавит языка программирования Python включает:

– буквы латинского алфавита и символ подчеркивания (_);

– арабские цифры;

– специальные символы (**. ,+ - * / = : < > [ ] { } ( ) ^ @ $ #<> <= >= := (* *)**);

– служебные слова;

– комментарии (#).

Для реализации алгоритма при разработке программы на языке программирования Python используются:

– данные;

– выражения;

– операторы (инструкции).

Данные представлены константами и переменными. Они делятся на пользовательские и системные.

Данные в программах на Python имеют тип. Типы данных можно классифицировать на простые и составные (сложные). Характеристика простых типов данных приведена в табл. 1.1.

Таблица 1.1

Стандартные типы данных в Python

Тип	**Обозначение**	**Характеристика типа**	**Примеры**
Целый	**int**	целые числа (положительные, отрицательные, ноль)	**14,** **-48,** **0,** **1286**
Вещественный	**float**	вещественные числа	**5.8,** **3.08,** **-6.5829,** **5.77e-4**

Окончание табл. 1.1

Тип	Обозначение	Характеристика типа	Примеры
Логический	**bool**	величины, которые могут принимать значения истина (true) или ложь (false)	**true (истина/ да/ 1)** **false (ложь/ нет / 0)**
Комплексный	**complex**	Комплексные числа	**4+7j,** **2.8 – 5j**

Переменные – это имеющие имена объекты, способные хранить некоторые данные. В языке Python в переменной хранится адрес объекта, расположенного в памяти, а не само значение объекта, в этом отличие Python от других языков программирования.

Формирование имени переменной необходимо осуществляется по четко оговоренным правилам с учетом ограничений. Перечень правил и ограничений при формировании имен переменных приведен ниже:

– имя может состоять только из букв латинского алфавита (ASCII, Unicode), знака подчеркивания (_), цифр (0-9);

– имя не может начинаться с цифры (4A – недопустимо, A4 – допустимое имя переменной);

– имя регистрозависимо (**A4** и **a4** – разные имена переменных); имя должно быть уникальным в программе и не может совпадать с зарезервированными словами языка Python.

Примеры корректно сформированных имен переменных:

V1, sp, prim4, sum3, rez1.

Язык программирования Python поддерживает динамическую типизацию. Это означает, что тип переменной определяется только во время исполнения программы.

Иногда существует необходимость преобразовать данные одного типа данных в другой. Для этого используем следующую конструкцию:

имя_типа(имя_переменной)

Например:

- **int(a)** – преобразует значение переменной ‘a’ в целый тип;
- **float(123)** – преобразует целое число 123 в вещественное число 123.0;
- **str(123)** – преобразует число 123 в строку ‘123’.

Выражения в Python делятся на арифметические и логические. Арифметическим выражением называется совокупность констант, переменных, стандартных функций, связанных знаками арифметических операций. Арифметические выражения могут содержать круглые скобки.

Арифметические операции можно выполнять с переменными целого и вещественного типа. Тип результата вычисления выражения зависит от типа данных. Если в выражении участвует переменная целого типа и переменная вещественного типа, то результат преобразуется к вещественному типу.

В табл. 1.2 приведен перечень основных арифметических операций.

Таблица 1.2

Основные арифметические операции

Название	Знак операции	Пример
Сложение	+	x+y
Вычитание	-	x-y
Умножение	*	x*y
Деление	/	x/y
Возведение в степень	**	x**5
Целочисленное деление	//	x//y
Остаток от деления	%	x%y

Последовательность выполнения операций при вычислении арифметического выражения в Python подчинена следующим правилам приоритета:

- приоритет операции возведения в степень выше приоритета операций умножения и деления;
- приоритет операций умножения и деления выше приоритета операций сложения и вычитания;
- одинаковые по приоритету операции выполняются в порядке следования в выражении слева направо.

Для изменения приоритета операций в арифметических выражениях используются круглые скобки. Степень вложения скобок не ограничивается.

Стандартные функции, которые можно использовать в арифметических выражениях, делятся на функции, встроенные в систему программирования и доступные без всяких условий и функции, со-

держащиеся в стандартных модулях. Количество встроенных функций для формирования арифметических выражений невелико, большинство стандартных функций содержится в модуле **math**.

В языке Python модуль – это особым образом оформленный файл с Python-кодом. Модуль может содержать переменные, функции, другие модули, классы и исполняемый код. Для того чтобы воспользоваться функциями, которые находятся в модуле, его необходимо подключить к программе с помощью инструкции **import**.

Возможности подключения функций модуля следующие:

- подключение модуля целиком;
- подключение модуля с сокращенной префиксной записью;
- подключение отдельных функций модуля;
- подключение всех функций модуля без префиксной записи.

Модуль может быть подключен целиком, тогда для использования функций этого модуля в программе нужно использовать точечную (префиксную) запись, например,

math.sqrt(9), numpy.sum(a).

Общий вид инструкции подключения модуля целиком следующий:

import Имя модуля

Например, подключив модуль **math** целиком, можно воспользоваться функциями этого модуля так:

```
import math
A=math.sqrt(9)
B=math.sin(4.7)
```

Общий вид инструкции подключения модуля целиком следующий с сокращенной префиксной записью:

import Имя модуля as Сокращение

Например, подключив модуль **math** целиком с сокращением **m**, можно воспользоваться функциями этого модуля так:

```
import math as m
A=m.sqrt(9)
```

Чаще всего такой способ подключения применяется, когда имя модуля достаточно велико или оно содержит еще и имя включенного в него модуля, например:

import matplotlib.pyplot as plt

Общий вид инструкции подключения конкретных функций модуля следующий:

from Имя модуля import Имя_функции

Например, подключив из модуля **math** только функцию **sqrt**, ею можно воспользоваться без префикса:

from math import sqrt
A=sqrt(9)

Инструкция, позволяющая получить доступ ко всем функции модуля без префиксной записи, имеет следующий общий вид:

from Имя модуля import*

Подключив таким образом модуль **math,** можно пользоваться его функциями без префикса:

from math import*
A=sqrt(9)

Перечень наиболее часто используемых в инженерных расчетах стандартных функций модуля **math** приведен в табл. 1.3.

Таблица 1.3

Функции модуля math

Описание	Имя	Описание	Имя
Экспонента	exp(x)	Синус	sin(x)
Натуральный логарифм	log(x)	Косинус	cos(x)
Десятичный логарифм	log10(x)	Тангенс	tan(x)
Корень квадратный	sqrt(x)	Арксинус	asin(x)
Основание натурального логарифма	e	Арккосинус	acos(x)
Число π	pi	Арктангенс	atan(x)
Перевод из радиан в градусы	degrees(x)	Перевод из градусов в радианы	radians(x)

Функция абсолютной величины abs(x) – это функция встроенная в систему. Ниже приведены примеры корректной записи арифметических выражений в Python, если модуль подключен инструкцией:

from math import*

Запись в математике:	Запись в Python:
$\frac{\cos x^2}{x+\sin^3 x}+e^{-2.1}$	**cos(x**2)/(x+sin(x)**3)+exp(-2.1)**
$\frac{1.2-2x}{\lg(x+3.2)}-\sqrt{\lvert x-5.8\rvert}$	**(1.2-2*x)/log10(x+3.2) - sqrt(abs(x-5.8))**

Оператор присваивания

Одним из основных операторов в Python является оператор присваивания. В программе этот оператор выполняется следующим образом – присваивает переменной, стоящей слева от знака «=» значение выражения, стоящего справа.

Общий вид оператора присваивания:

Имя_переменной = Выражение

В качестве параметра «**Имя_переменной»** может выступать имя одной переменной или несколько имен переменных, разделенных запятыми. В качестве параметра «**Выражение»** применяется арифметическое, логическое или строковое выражение.

Ниже приведены примеры правильной записи оператора присваивания (модуль подключен инструкцией **from math import*,** переменным x, c, d, p, a присвоены числовые значения).

A = cos(x)+c-(d2)*(p**2)+4.92**

B= atan(a*c)+ sqrt(p3)**

В некоторых случаях вместо оператора простого присваивания используются оператор составного присваивания. Описание операторов составного присваивания приведено в табл. 1.4.

Таблица 1.4

Операции составного присваивания в Python

Знак операции	Название операции	Значение переменных		Составное присваивание	Результат	
		a	b		a	b
+=	сложение с присваиванием	10	3	**a+=b**	13	3
-=	вычитание с присваиванием	10	3	**a -=b**	7	3

Окончание таблицы 1.4

Знак операции	Название операции	Значение переменных		Составное присваивание	Результат	
		a	b		a	b
*=	умножение с присваиванием	10	3	**a*=b**	30	3
/=	деление с присваиванием	10	3	**a/=b**	3.33	3
//=	целочисленное деление с присваиванием	10	3	**a//=b**	3	3
**=	возведение в степень с присваиванием	10	3	**a**=b**	1000	3

Вывод данных в языке Python

Для вывода информации в Python используется специальная функция **print()**, которую, в силу ее значимости при программировании, принято называть оператором. Она имеет ряд параметров, основным из которых является список выводимых данных и/или выражений.

Общий вид функции без учета опциональных параметров следующий:

print(Список вывода)

Здесь **Список вывода** – перечень имен переменных, констант, выражений, разделенных запятыми. Ниже приведены примеры корректной записи функции **print** и результатов ее работы.

Программа	Интерактивное окно
print("Изучаем Python!!!")	**Изучаем Python!!!**
print(10)	**10**
a=5	
b=3	
print(a)	**5**
print(a+b)	**8**
print(a,b)	**5 3**

При форматном выводе с использованием метода **format** список вывода оператора **print** имеет следующий общий вид:

СтрокаФормата.format(Список вывода)

Параметр **СтрокаФормата** – это символьная константа или символьная переменная, содержащая трафарет для вывода. Примеры вывода с использованием метода **format** приведены ниже.

Фрагмент программы	Интерактивное окно
x=31.546 **y1=x**2** **y2=y1/345** **fviv= 'y1={0:.3f}, y2={1:.2f}'** **print(fviv.format(y1,y2))**	**y1= 995.150, y2=2.88**

Ввод данных в языке Python

В языке Python в качестве оператора ввода используется функция **input().**

Она имеет следующий общий вид:

ИМЯ = input(Символьная константа)

Здесь **ИМЯ** – это имя простой переменной, **Символьная константа** – любой набор символов, заключенный в двойные кавычки. Символьная константа, как правило, разъясняет смысловое назначение вводимой переменной. Например:

Фрагмент программы	Интерактивное окно
S=input("Задайте площадь S=")	**Задайте площадь S=**
A=input("Задайте значение A=")	**Задайте значение A=**

Оператор выполняется следующим образом: в интерактивном окне выводится набор символов, стоящий в скобках после **input** (символьная константа), выполнение программы приостанавливается, и компьютер переходит в режим ожидания; пользователь вводит константу, и ссылка на введенную константу помещается в оперативной памяти в переменную, стоящую слева от функции **input**.

При запуске на выполнение программы, содержащей оператор ввода, следует учитывать, что пока пользователь не ввел константу в ответ на запрос своей программы, оператор ввода продолжает свою работу. Система Python в это время блокирует выход и закрытие интерактивного окна.

Условный оператор

Для программирования разветвляющихся алгоритмов в Python существуют условный оператор **if**.

В условный оператор входит логическое выражение, которое состоит из логических констант и переменных, арифметических выражений, связанных знаками логических операций и операций сравнения.

Перечень логических операций и правила их выполнения приведены ниже.

– and – логическое И: результат true, если оба операнда равны true;

– or – логическое ИЛИ: результат true, если хотя бы один операнд равен true;

– not – логическое НЕ: результат true, если операнд равен false и наоборот.

Перечень операций сравнения приведен в табл. 1.5.

Таблица 1.5

Операции сравнения в Python

Знак операции	**Название операции**	**Значение переменных**		**Результат**
		a	**b**	
==	равно	10 2	3 2	false true
!=	не равно	10 2	3 2	true false
>	больше	10	3	true
>=	больше или равно	10	3	true
<	меньше	10	3	false
<=	меньше или равно	10	3	false

В простых логических выражениях используется знак операции сравнения, например: ***a > b*** или ***c <= 5.8***. В сложных логических выражениях используются знаки логических операций, например:

a>b or b<0

(a > 0) and (b>0)

Условный оператор представлен в нескольких формах. Общий вид оператора полной формы 1 следующий:

```
if ЛогическоеВыражение:
    БлокИнструкций1
else :
    БлокИнструкций2
```

Порядок выполнения условного оператора, записанного в полной форме 1, следующий:

- логическое выражение вычисляется до константы (true или false);
- если логическое выражение истинно, то выполняется блок инструкций, стоящий после логического выражения, а затем следующая за оператором **if** инструкция;
- если логическое выражение ложно, то выполняется блок инструкций, стоящий после слова **else**, а затем следующая за оператором **if** инструкция.

Общий вид оператора полной формы 2 следующий:

```
if ЛогическоеВыражение1:
    БлокИнструкций 1
elif ЛогическоеВыражение2:
    БлокИнструкций 2
elif ЛогическоеВыражение3:
    БлокИнструкций 3
…
else :
    БлокИнструкций N
```

Порядок выполнения условного оператора, записанного в полной форме 2, следующий:

- логическое выражение первого оператора **if** вычисляется до константы (true или false);
- если логическое выражение1 истинно, то выполняется блок инструкций1, стоящий после логического выражения1, а затем следующая за инструкцией **if** инструкция;

- если логическое выражение ложно, то вычисляется логическое выражение, стоящее после слова **elif,** встречающегося первый раз;

- если это логическое выражение истинно, то выполняется блок инструкций, стоящий после логического выражения, а затем следующая за оператором **if** инструкция;

- процесс повторяется столько раз, сколько слов **elif** содержит оператор **if**;

- если последнее логическое выражение ложно, то выполняется, блок инструкций, стоящий после слова **else**, а затем следующая за оператором **if** инструкция.

Краткая форма оператора **if** имеет следующий общий вид:

if ЛогическоеВыражение:
БлокИнструкций 1

Порядок выполнения условного оператора, записанного в краткой форме, следующий:

- логическое выражение вычисляется до константы (true или false);

- если логическое выражение истинно, то выполняется блок инструкций, стоящий после логического выражения, а затем следующая за оператором **if** инструкция;

- если логическое выражение ложно, то выполняется следующая за оператором **if** инструкция.

Блок инструкций, должен быть оформлен с учетом табулирования, определяющего тело блока.

Практическая часть

Задание 1

Написать программу для решения одного из представленных ниже вариантов задач. Отладить программу для предложенных тестовых значений переменных.

Вариант № 1

Заземлитель в форме кольца радиусом ***r*** расположен в грунте на глубине ***h***. Его сопротивление при ***h >> r*** рассчитывается по формуле

$$R = \frac{1}{4\pi^2 rG}\left[\frac{\pi r}{h} + \ln\left(\frac{16r}{d}\right)\right]$$

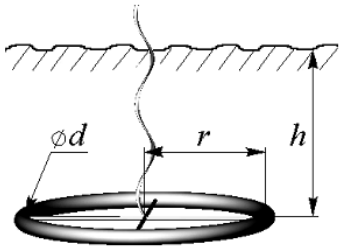
где π = 3,14…,

G = 0,03 1/Ом · м − электропроводность грунта,

d =0.025м− диаметр проводника из которого изготовлено кольцо,

r=0.219м – радиус,

h=0.9



Задание – вычислить значение сопротивления заземлителя.

Вариант № 2

Заземлитель, изготовленный в виде решетки прямоугольной формы из металлических труб, расположен горизонтально в грунте на глубине ***h***. Сопротивление заземлителя рассчитывается по формуле

$$R = \frac{\ln\left(\frac{L^2}{2rh}\right) + 4{,}95}{2\pi LG},$$

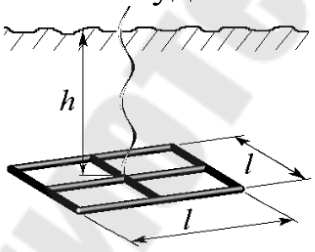
где π = 3,14…,

L = 8.43− суммарная длина труб,

r=0.01 − радиус труб,

h =1.5 − глубина,

G =0.02 − удельная электропроводность грунта.



Задание – вычислить значение сопротивления заземлителя.

Вариант № 3

В гибридных интегральных схемах используются плоские пленочные катушки индуктивности в виде квадратной спирали. Индуктивность такой катушки (в наногенри) приближенно определяется по формуле:

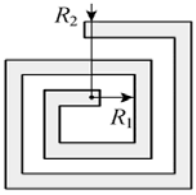
$$L = 2,41 \cdot a\ N^{\frac{5}{3}} \ln\left(\frac{8a}{c}\right),$$

где N =5 – число витков,

a=(R_1+R_2)/2,

c=R_2-R_1

R_1=2.5, R_2 =4 – размеры внутреннего и внешнего витков катушки.



Задание – вычислить значение индуктивности катушки.

Вариант № 4

Электрическая емкость двух коаксиальных плоских дисков (см. рисунок) при L/R<1 рассчитывается по формуле

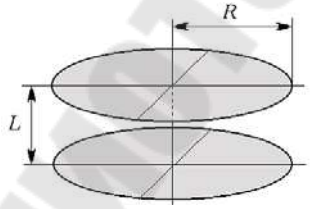
$$C = \varepsilon_1\varepsilon_0 R\left[\frac{\pi R}{L} + \ln\left(\frac{16\pi R}{L}\right) - 1\right],$$

где ε1=0.01 − относительная диэлектрическая проницаемость среды,

ε0 = 0.885 пФ/мм;

R=2.87 − радиус дисков,

L=1.2 − расстояние между дисками.



Задание – вычислить значение емкости.

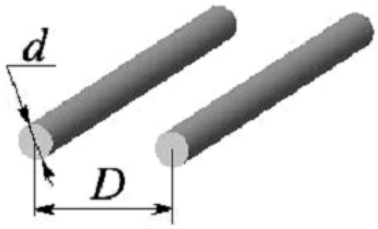
Вариант № 5

Волновое сопротивление двухпроводной линии рассчитывается по формуле

$$Z_0 = \frac{276}{\sqrt{\varepsilon}} \ln\left(\frac{D}{d} + \sqrt{1 + \frac{D^2}{d^2}}\right)$$

где $\varepsilon = 3.35$ − относительная диэлектрическая проницаемость среды, в которой находится двухпроводная линия,

d=1.5 и D=3.22 – соответственно диаметры проводников и расстояние между их осями.



Задание – вычислить значение волнового сопротивления.

Вариант № 6

В гибридных интегральных схемах в качестве одновитковой индуктивности может применяться тонкая металлическая полоска, нанесенная на диэлектрическую подложку в виде круглой петли. Индуктивность такой петли в наногенри приближенно определяется по формуле

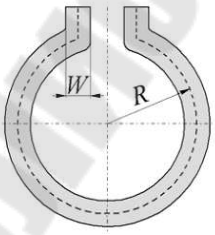
$$L = 1{,}257 \cdot R\left[\ln\left(\frac{8\pi R}{W + t}\right) - 2\right],$$

где R=5.85 − радиус средней линии петли,

W=0.53 − ширина металлической полоски,

T=0.03 − ее толщина.

Все размеры в формуле указаны в миллиметрах.



Задание – вычислить значение индуктивности.

Задание 2

Написать программу и провести ее тестирование для одного из следующих заданий.

Вариант № 1

Вычислить значение **z** при различных значениях k, значение k ввести с клавиатуры. Для тестирования взять k=2.7 и k=1.7.

$$z = \frac{a^2}{b} + \sqrt{b} \cdot \sin t.$$

$$b = \begin{cases} |t + a| & если \quad a < 20 \\ |t - a| & если \quad a \geq 20 \end{cases} \qquad a = \begin{cases} y & если \quad y > 6 \\ y^2 & если \quad y \leq 6 \end{cases}$$

$y = 3.6(\sin k - \cos k)^2$ t=7,8

Вариант № 2

Вычислить значение **p** при различных значениях k, значение k ввести с клавиатуры. Для тестирования взять k=2.3 и k=0.3.

$$p = \frac{b^2 + \sqrt{b} \cdot \sin t}{60} +$$

$$b = \begin{cases} \sqrt{|t + z|} & если \quad z < 0 \\ |t - z| & если \quad z \geq 0 \end{cases} \qquad z = 5(\sin^2 k - \cos k)$$

t = 12,7

Вариант № 3

Вычислить значение **k** при различных значениях **t**, значение t ввести с клавиатуры. Для тестирования взять t=1.8 и t=0.57

$$k = \frac{p + q}{m^2 + \sin^2 p}.$$

$$q = \begin{cases} p^2 + m^2 & если \quad p < 10.3 \\ |p - m| & если \quad p \geq 10.3 \end{cases}$$ m=3.57

$$p=\begin{cases} z & если \quad z>m \\ 0.1 & если \quad z\le m \end{cases} \qquad z=\sin(t)+0.865\sqrt{t}+t^2$$

Вариант № 4

Вычислить значение **f** при различных значениях **m**, значение **m** ввести с клавиатуры. Для тестирования взять m=15.8 и m=0.57.

$$f=\frac{b^2}{b+\sin^2 y}$$

$$b=\begin{cases} |t+y| & если \quad y<1 \\ |t-y| & если \quad y\ge 1 \end{cases} \qquad \text{t = 2,8}$$

$$y=\frac{3{,}6\cdot(\sin m-\cos^2 m)}{t}.$$

Вариант № 5

Вычислить значение **z** при различных значениях k, значение k ввести с клавиатуры. Для тестирования взять k=5.6 и k=28.6. Значение t = 3,8

$$b=\begin{cases} |t+a| & если \quad a<20 \\ |t^2-a| & если \quad a\ge 20 \end{cases} \qquad z=\frac{a^2\cdot\sin(t)}{(b+1)^2}$$

Значение ***a=y***, если ***y*** больше 7, и ***a=y²*** в остальных случаях

$$y=(\sin k-\cos k^2)+e^{t-2}$$

Вариант № 6

Вычислить значение **k** при различных значениях **t**, значение t ввести с клавиатуры. Для тестирования взять t=1.8 и t=0.57

$$k=\frac{0.04p+q}{m^2+\sin^2 p}.$$

$$q=\begin{cases} p^2+m^2 & если \quad p<10.3 \\ |p-m| & если \quad p\ge 10.3 \end{cases} \qquad \text{m=3.57}$$

$$p=\begin{cases} z & если \quad z>m \\ 0.1 & если \quad z\le m \end{cases} \qquad z=\sin(t)+0.865\sqrt{t}+t^2$$

Вариант № 7

Вычислить значение **k** при различных значениях **t**, значение t ввести с клавиатуры. Для тестирования взять t=1.8 и t=0.57

$$k = \frac{p+q}{m^2 + \sin^2 p}.$$

$$q = \begin{cases} p^2 + m^2 & если \quad p < 10.3 \\ |p - m| & если \quad p \geq 10.3 \end{cases} \qquad m=3.57$$

$$z = \sin(t) + 0.865\sqrt{t} + t^2$$

Значение ***p*** равно ***z***, если ***z*** больше ***m***, иначе ***p*** равно ***0.1***.

Задание 3

Закон движения руки робота-манипулятора задан в виде параметрических зависимостей x(t) и y(t). Написать программу вычисления координат положения руки робота для момента времени t, заданного с клавиатуры. На плоскости находится объект с координатами (0,0). Сделать вывод о том, где по отношению к объекту находится рука робота в плоскости, например: выше и левее, ниже и правее и т. д. Провести не менее трех тестов.

Варианты заданий приведены в табл. 1.6.

Таблица 1.6

Варианты задания 3

Вариант 1. x=2sin(2t) y=3+3cos(4t) Значение t выбирать из диапазона от 0.1 до 3	Вариант 2. $x=1/(t^2+2)$ $y=(t-1)/(t+2)^2$ Значение t выбирать из диапазона от 0.1 до 20
Вариант 3. $x= -t\cdot\sin t/(t^2+1)$ $y= (\cos(t)\cdot t^2)/(t^2+1)$ Значение t выбирать из диапазона от 0.1 до 6	Вариант 4. x=6(2t+sin(2t)) y=6(t-cos(2t)) Значение t выбирать из диапазона от 0.1 до 3
Вариант 5. x=16/(t+2) $y=(40-60t)/(t+2)^3$	Вариант 6. $x=(-2/(t^3+1) +1)$ $y= t^3-5$

Значение t выбирать из диапазона от 0 до 20	Значение t выбирать из диапазона от 0 до 2
Вариант 7. x=11cos(2t)(1+cos(2t)) y=11sin(2t)(1+cos(2t)) Значение t выбирать из диапазона от 0 до 3	Вариант 8. x=11sin(3t) y=21cos(3t)+12 Значение t выбирать из диапазона от 0 до 2

Задание 4

Написать программу и провести ее тестирование для следующего задания.

Робот находится в центре окружности диаметром не менее N см (рис.1.2). С помощью коротких отрезков окружность разделена на восемь равных частей, помеченных флажками**.** Рассчитать кратчайшее расстояние, пройденное роботом, чтобы переместиться от места его расположения сначала к флажку с номером 0, потом к заданному флажку (двигаясь строго по окружности), затем - опять в центр. Номер флажка отсчитывается по часовой стрелке, робот может двигаться в любом направлении.

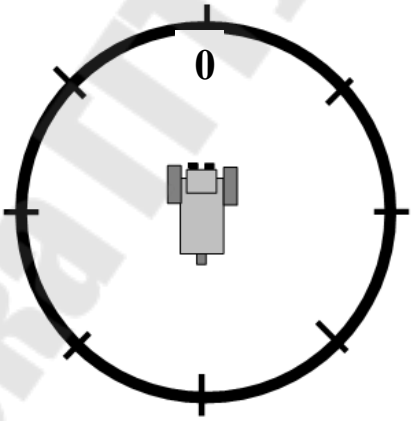


Рис.1.2. Иллюстрация движения робота

Лабораторная работа № 2
Программирование циклических алгоритмов

Цель работы: получить навыки составления программы по реализации циклических алгоритмам и готовить тесты для отладки этих программ, научиться решать задачу о табулировании функции.

Краткие теоретические сведения

Операторы цикла в Python

Операторы цикла предназначены для программирования циклических алгоритмов, они изменяют линейный порядок выполнения операторов программы и относятся к операторам управления.

Операторы цикла в Python можно классифицировать следующим образом:

- оператор цикла с параметрами **for**.
- оператор цикла с предусловием **while**.

При классификации циклических алгоритмов отмечалось, что цикл может управляться переменной цикла, имеющей параметры, либо цикл не содержит переменной цикла и управляется иным способом. Операторы цикла можно разделить по тому же принципу:

– для программирования циклов с переменной цикла и параметрами;

– для программирования циклов без переменной цикла.

Операторы являются взаимозаменяемыми, выбор для применения того или иного оператора зависит от программиста.

Оператор цикла for

Оператор цикла **for** предназначен для программирования циклических алгоритмов, когда переменная цикла явно выражена и изменяется в определенном диапазоне значений.

Общий вид оператора цикла for следующий:

for ПеременнаяЦикла **in** ПоследовательностьЗначений**:**
Тело цикла

Здесь **ПеременнаяЦикла** – простая переменная, которая управляет работой цикла.

Тело цикла – это любая инструкция или блок инструкций Python, которые должны быть выполнены в цикле несколько раз. Тело цикла еще называется рабочей частью цикла.

Параметр **ПоследовательностьЗначений** может быть задан:

– списком значений;

– с применением функции **range**.

Функция **range()** задает последовательность данных в зависимости от количества переданных в функцию значений:

– **range(num)** – возвращает список целых чисел от 0 до **num-1**;

– **range(num1, num2)** – возвращает список целых чисел от **num1** до **num2-1**;

– **range(num1, num2, num3)** – возвращает список целых чисел от **num1** до **num2-1** с шагом **num3**.

Примеры правильной записи оператора цикла **for** приведены ниже.

Фрагмент программы	Интерактивное окно
num=[1,5,4,7] **for i in num:** ** print(i,end=" ")**	**1 5 4 7**
for i in range(5): ** print(i)**	**0** **1** **2** **3** **4**
for i in range(1,10): ** print(i,end=", ")**	**1, 2, 3, 4, 5, 6, 7, 8, 9,**
for i in range(-10, 11, 5): ** print(i)**	**-10** **-5** **0** **5** **10**

Цикл **for** может принимать форму, в которой используется ключевое слово **else**. Эта форма позволяет реализовать инструкции в том случае, если последовательность значений переменной цикла исчерпана.

Общий вид оператора цикла **for** с использованием **else** следующий:

for ПеременнаяЦикла **in** ПоследовательностьЗначений**:**
Рабочая часть цикла
else:
Блок инструкций

Оператор цикла с предусловием

Оператор **while** предназначен для программирования любых циклов, где проверка условия повторения цикла производится перед выполнением рабочей части цикла.

Общий вид:

while ЛогическоеВыражение:
Рабочая часть цикла

Порядок выполнения оператора следующий:

- проверяется истинность логического выражения;
- до тех пор, пока оно истинно, выполняется блок инструкций рабочей части цикла;
- если логическое выражение стало ложным, то выполняется следующий за оператором цикла оператор программы.

Рекомендации по программированию и использованием этого оператора цикла приведены ниже.

Если переменная цикла явно выражена, то:

- нужно присвоить ей начальное значение до оператора цикла;
- нужно каким-либо образом изменять ее в рабочей части цикла;
- она должна использоваться в логическом выражении условия повторения цикла.

Если логическое выражение в заголовке цикла **while** заведомо ложно, то цикл не выполняется вообще. Примеры правильной записи оператора цикла **while** приведены ниже.

Фрагмент программы для вывода на экран дисплея четные целые числа от 2 до 20 имеет вид:

```
k=2
while k<=20:
   print(k)
   k=k+2
```

Еще один пример применения оператора цикла while: нужно выводить на экран дисплея сообщение «Нажмите клавишу 5» до тех пор, пока эта клавиша не будет нажата.

```
p=0
while p != 5:
   p=int(input('Нажмите 5'))
```

Практическая часть

Задание 1

Написать и оттестировать программу для решения следующих задач.

1.1. В соответствии с видом функции, приведенном в табл. 2.1, вычислить значения функции y=f(x) для значений аргумента x, изменяющегося в интервале от a до b с шагом dx. Значение шага предварительно рассчитать так, чтобы вычисление функции производилось в N заданных точках, значение N ввести с клавиатуры, оно должно быть не менее 10.

1.2. Определить, сколько раз функция меняет знак на отрезке от a до b.

Таблица 2.1

Варианты задания 1

Вариант	Функция y(x)	Пределы изменения аргумента функции
1	$x^2 \cdot \sin(2x)$	a=5 b=15
2	$\cos(x) - 2\sin(2x)$)	a=6 b=14
3	$2\sin(0{,}53x) - \cos^2(x)$)	a=0 b=15
4	$\sin(0{,}3x) - 1 - 2\cos(0{,}5x)$	a=5 b=35
5	$\sin(x)\cos(x-3{,}53) - \sin(x) - \sin(x+1{,}87) \cdot \cos(x)$	a=3 b=15
6	$1+\sin(2x)-2\cos(3x)\sin(x)$	a=4 b=10
7	$\sin^2(x-0{,}65) - 0{,}2x$	a=0 b=6
8	$3{,}5\sin(x - 3{,}5) - \ln(x)$	a=1 b=15
9	$\sin^2(x) - 0{,}2\ln(x)$	a=2 b=10
10	$1 - \cos(2x) - \ln(x)$	a=2 b=6

Задание 2

Разработать и отладить на тестовых примерах программу для решения следующей задачи.

Камера наблюдения регистрирует в автоматическом режиме скорость проезжающих мимо неё автомобилей, округляя значения скорости до целых чисел.

Если значение скорости равно 0, то камера считается выключенной и процесс регистрации завершен.

Индивидуальные варианты задания 1 приведены в табл. 2.2

Таблица 2.2

Варианты задания 2

	Основное задание	**Дополнительное задание**
1	Необходимо определить максимальную зарегистрированную скорость автомобиля. Если скорость трех и более автомобилей была меньше 30 км/ч, вывести сообщение «Есть автомобили со скоростью меньше 30», иначе выведите «Нет автомобилей со скоростью меньше 30».	Выдать сообщение «Камера неисправна», если первое зарегистрированное значение скорости равно 0
2	Необходимо определить минимальную зарегистрированную скорость автомобиля. Если скорость двух и более автомобилей была больше 80 км/ч, выведите «Есть автомобили со скоростью больше 80», иначе выведите «Нет автомобилей со скоростью больше 80».	Выдать сообщение «Камера неисправна», если два из зарегистрированных значений скорости больше 300. При обнаружении неисправности камера отключается
3	Необходимо определить среднюю зарегистрированную скорость всех автомобилей. Если скорость трех и более автомобилей была не меньше 60 км/ч, выведите «Есть автомобили со скоростью не меньше 60», иначе выведите «Нет автомобилей со скоростью не меньше 60».	Выдать сообщение «Камера неисправна», если первое зарегистрированное значение скорости равно нулю. При обнаружении неисправности камера отключается

Окончание табл. 2.2

	Основное задание	**Дополнительное задание**
4	Необходимо определить, сколько автомобилей нарушили скоростной режим (ограничение 40 км/ч) и каково средние превышение скорости при нарушениях	Выдать сообщение «Камера неисправна», если зарегистрировано значения скорости больше 250. При обнаружении неисправности камера отключается
5	Необходимо определить, среднюю скорость автомобилей, нарушивших скоростной режим (ограничение 40 км/ч) и количество автомобилей, ехавших без нарушений	Выдать сообщение «Камера неисправна», если первое зарегистрированное значение скорости равно нулю. При обнаружении неисправности камера отключается
6	Необходимо определить, максимальное значение скорости автомобилей, нарушивших скоростной режим (ограничение 40 км/ч) и каково средние превышение скорости при нарушениях	Выдать сообщение «Камера неисправна», если зарегистрированы два значения скорости больше 250. При обнаружении неисправности камера отключается
7	Необходимо определить, минимальную скорость автомобилей, нарушивших скоростной режим (ограничение 40 км/ч) и среднюю скорость автомобилей, ехавших без нарушений	Выдать сообщение «Камера неисправна», если первое зарегистрированное значение скорости равно нулю.

Задание 3

3.1. Разработать программу проверки знания таблицы умножения, программа должна задавать вопросы по таблице умножения двух случайных целых чисел от 2 до 9 (5 вопросов) и принимать ответ пользователя, подсчитывать количестве правильных ответов, выдавать оценку по пятибалльной системе. Вывести количество ответов,

на которые пользователь потратил свыше заданного времени (например, двух или трех секунд) на ответ.

3.2. Разработать программу, которая позволяет пользователю отгадать сгенерированное компьютером число от 0 до 100. Пользователю дается несколько попыток, в ответ программа выдает сообщение, больше или меньше введенное число, чем сгенерированное число. Количество попыток ограничено. В результате выдается само число и количество попыток.

Лабораторная работа № 3
Пользовательские функции

Цель работы: получить навыки работы с пользовательские функции и модулями в Python.

Краткие теоретические сведения

Программирование пользовательских функций

Подпрограммой называется особым образом оформленная программная единица, которая описывается один раз и может вызываться по имени из любой программной единицы несколько раз. Как правило, подпрограммы реализуют какую-либо законченную часть задачи, многократно использующуюся в других задачах.

Для того, чтобы операторы подпрограммы стали известны в вызывающей программной единице, подпрограмма должна быть определена заранее.

В Python подпрограммы представлены подпрограммами-функциями, которые могут быть классифицированы так:

- глобальные функции, доступ к которым возможен из любой точки программы;
- локальные функции, доступ к которым возможен только в той функции, внутри которой они определены;
- анонимные лямбда-функции, являющиеся выражениями без имени.

Заголовок функции оформляется следующим образом:

def ИмяФункции(СписокВходныхПараметров):

Здесь **ИмяФункции** – идентификатор, по которому к функции выполняется обращение из вызывающей программы. Имя функции

формируется по правилам формирования имен переменных, но в имени функции, как правило, не пишут заглавных букв.

СписокВходныхПараметров – список констант, переменных, выражений, имен функций, разделенный знаком «,». Список параметров может отсутствовать, но скобки остаются.

Примеры правильной записи заголовка функции:

```
def  summa (a, b, c):
def  prim (x,y):
def  kluch ( ):
```

Блок или тело функции может содержать инструкции Python и заканчиваться необязательным ключевым словом **return,** после которого указывается выходной параметр функции. Если ключевое слово **return** отсутствует, то окончанием функции считается последняя инструкция, смещенная на меньшее число пробелов.

Инструкция **return** служит для передачи результатов расчетов (выходных параметров) из функции в вызывающую программу. Общий вид описания функции с использованием инструкции **return** следующий:

```
def ИмяФункции(СписокВходныхПараметров):
        БлокИнструкций
        return ВыходнойПараметр
```

ВыходнойПараметр – имя переменной или арифметическое выражение, значение которого будут передаваться в вызывающую программную единицу.

Все параметры, указанные в скобках после имени функции, называются входными, значения этих параметров передаются в в функцию, но не возвращаются назад в вызывающую программу.

Выходной параметр или параметры указываются сразу после слова **return**, они передают свои значения в вызывающую программу.

Например, в приведенном ниже описании функции выходным параметром является переменная M, входными параметрами – переменные **a, b, c**.

```
def  fun1(a, b, c):
        D=a+b**2+3*c
        M=D-(a+b)/2
        return M
```

В качестве выходного параметра может использоваться арифметическое выражение, например:

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

Если инструкция **return** отсутствует, то тогда выполняются все инструкции тела функции, и возвращается значение **None**.

Для обращения к функции необходимо указать имя функции и список фактических параметров, если он предусмотрен, в любой инструкции Python.

Общий вид:

ИмяФункции(СписокФактическихПараметров)

Примеры обращения к функциям с использованием операторов вывода, присваивания и условного оператора приведены ниже.

```
print(summa(a,b,S))
A=fun1(D,Y,X)
if  fn2(a,b) > 0:
        r=r+1
```

Параметры, используемые при описании функции, называются формальными. Параметры, используемые при вызове функции, называются фактическими.

Например:

def summa (a, b, c)

a, b, c – формальные параметры,

Z=summa(D,Y,X)

D,Y,X – фактические параметры.

В качестве фактических параметров могут использоваться константы, переменные, выражения, например,

Z=summa(5, a, X+Y)

Если нужно передать в вызывающую программу больше одного выходного параметра, то применяется объединение нескольких параметров в одно данное составного типа, например, в список или кортеж.

Например, необходимо вычислить в одной функции сумму квадратов и сумму кубов двух чисел, передать два результата в основную программу.

Фрагмент программы	Интерактивное окно
def sum23(a, b): **s2=a**2+b**2** **s3=a**3+b**3** **return [s2, s3]** **[S2, S3]=sum23(4, 5)** **print('сумма квадратов', S2)** **print('сумма кубов', S3)**	**сумма квадратов 41** **сумма кубов 189**

Если количество входных параметров функции заранее не определено, то в списке входных параметров указывается символ «*».

В приведенном ниже примере **s** – выходной параметр, ***elem** – входной параметр (перед именем обязательно наличие *), который позволяет получить доступ к неопределенному количеству параметров.

```
def primsum(*elem):
   s = 0
   for i in elem:
      s =s+i
   return s
sum1 = primsum (1, 2, 3, 4, 5, 6)    # 21
sum2 = primsum (3, 1, 5, 2)        # 11
print(sum1)
print(sum2)
```

При определении входных параметров функции существует возможность присвоить значения параметрам в заголовке функции. Такие параметры называются необязательными или опциональными. Если функция имеет один или несколько необязательные параметров, они должны находиться в конце списка входных параметров. Чтобы сделать параметр необязательным, ему нужно присвоить начальное значение.

В примере, приведенном ниже, параметр **a** является обязательным, а параметр **b** – необязательным, его значение будет равно 3, если при вызове функции замещающий его фактический параметр не указан. Если при вызове функции фактический параметр указан (в примере он равен 5), то в вычислениях будет использоваться именно это значение параметра.

Фрагмент программы	Интерактивное окно
def sum2(a, b=3): ** return a**2+b**2** **print(sum2(10))** **print(sum2(10, 5)**	 **109** **125**

Все параметры функции и переменные, используемые при ее программировании, называются локальными, область их действия ограничивается только телом самой функции. Если переменная, используемая в программе, определена вне функций и доступна в любом месте программы – она является глобальной. Если имя локальной и глобальной переменной совпадает, то в теле функции будет использоваться только локальное значение переменной.

Фрагмент программы	Интерактивное окно
x=2 **y=3** **def sum():** ** return(x+y)** **print("x+y =",sum())**	 **x+y = 5**

В теле функции также можно создать переменную с глобальной областью видимости, используя при этом ключевое слово **global.**

Фрагмент программы	Интерактивное окно
x=2 **y=3** **def sub():** ** global z** ** z=5** ** return(x-y)** **def mult():** ** return(x*y*z)** **print("x-y =", sub())** **print("x*y*z =", mult())**	 **x-y = -1** **x*y*z = 30**

Анонимная lambda-функция

Анонимная **lambda**-функция используется для создания короткой однострочной функции. Общий вид **lambda**-функции следующий:

ИмяПеременной = lambda СписокПараметров:Выражение

Например, если нужно создать функцию, находящую сумму квадратов двух чисел, то запрограммировать ее можно так:

```
sum2=lambda x, y: x**2+y**2
```

Имя переменной sum2 хранит результат работы функции, формальные параметры функции – переменные x и y.

Вызов **lambda**-функции выполняется по тем же правилам, что и вызов функции **def**.

Например, нужно вывести в интерактивное окно сумму квадратов чисел 3 и 5, для этого можно воспользоваться инструкциями:

```
s1=sum2(3, 5)
print(s1)
```

Практическая часть

Задание 1

Варианты 1и 2

Условие задачи. Транспортный робот находится в точке с координатами (x0, y0) и должен попасть в точку с координатами (xf, yf) по одной из двух траекторий. Найти кратчайшее расстояние, которое пройдет робот до финишной точки.

Исходные данные:

- функция траектории 1;
- функция траектории 2;
- координата x стартовой точки
- координата x финишной точки;
- шаг dx, с которым изменяется координата x робота

Варианты исходных данных приведены в табл. 3.1.

Математическая формула для вычисления расстояния между двумя точками имеет вид:

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Таблица 3.1

Варианты исходных данных

	функция траектории 1	**функция траектории 2**	***x0***	***xf***	***dx***
1	$f1(x) := -\left(3\cdot x^4 - 8x^3 + 6x\right)$	$f2(x) := \left(3.06\cdot x^2 - 8x^3 + 6x\right)$	*0*	*1.1*	*0.05*
2	$f1(x) := x\cdot\sqrt{2 - 1.2\cdot x^2}$	$f2(x) := x^2\cdot\sqrt{2 - 1.2\cdot x}$	*0*	*1*	*0.05*

Варианты 3и 4

Условие задачи. Два транспортных робота находится в стартовых точках с разными координатами (x0, y10) и(x0, y20) и должны попасть в точку с координатами (xf, yf). Найти расстояние, пройденное каждым роботом до финишной точки и определить, какой из роботов придет к финишу первым, если они движутся с одинаковой скоростью.

Исходные данные:

- функция траектории робота1;
- функция траектории робота2;
- координата x стартовой точки
- координата x финишной точки;
- шаг dx, с которым изменяется координата x роботов.

Варианты исходных данных приведены в табл. 3.2.

Математическая формула для вычисления расстояния между двумя точками имеет вид:

$$d = \sqrt{(x2 - x1)^2 + (y2 - y1)^2}$$

Таблица 3.2

Варианты исходных данных

	функция траектории робота1	**функция траектории робота2**	***x0***	***xf***	***dx***
3	$f1(x) := \frac{x^3}{1.6} + 1.2x^2$	$f2(x) := 5.8x + 1.95\cdot x^2$	*-3.5*	*0*	*0.1*
4	$f1(x) := (\sin(x) - 4.7)\cdot x^{\frac{2}{3}}$	$\mathbf{f2(x) := 5\cdot\sin(x - 3.1)}$	*6*	*0.56*	*0.2*

Задание 2

Условие задачи. Разработать меню простого выбора, в котором отработка его пунктов выполняется с использованием функций. Провести тестирование программы. Виды меню и задачи для каждого пункта приведены в табл.3.3. Все исходные данные и математические формулы подобрать самостоятельно.

Можно придумать тематику меню и его обработки самостоятельно.

Таблица 3.3

Виды меню и задачи для каждого пункта

	Меню
1	*Вычислить площадь следующих геометрических фигур* *1. Площадь трапеции* *2. Площадь кольца* *3. Площадь кругового сектора* *4. Выход*
2	*Вычислить площади боковой поверхности* *1. Площади боковой поверхности цилиндра* *2. Площади боковой поверхности пирамиды* *3. Площади боковой поверхности конуса* *4. Выход*
3	*Вычислить объем* *1. Объем шара* *2. Объем конуса* *3. Объем цилиндра* *4. Выход*
4	*Вычисление значений функций (табулирование функции) в полярной системе координат* *1. Спираль Архимеда* *2. Спираль Галилея* *3. Спираль Ферми* *4. Выход*

Задание 3

Создать модуль из функций заданий 1 и 2. Создать программу, в которой этот модуль подключается с помощью инструкции **include,** проверить корректность выполнения функций модуля.

Задание 4

Создать lambda-функцию Z(x,y) и вычислить ее значения в двух заданных точках (x1,y1), (x2,y2). Варианты задания 4 приведены в табл. 3.4.

Таблица 3.4

Варианты задания 4

№	*Функция Z(x,y)*	*x1*	*x2*	*y1*	*y2*
1	$e^{7,2xy} + \cos\left(x^2 + y\right)$	0,15	1,3	1,01	1,81
2	$\frac{2}{3} tg^2 \frac{x}{y^3} - \frac{4x + y}{y - \sin x}$	1,3	2,8	0,8	2,1
3	$0{,}17xy + \sqrt{\lvert xy \rvert}$	1,12	1,87	1,38	2,03
4	$\frac{x + 2y}{\lvert x - y \rvert} + \sin^2(y - 3x)$	0,1	-0,4	0,6	1,8
5	$e^{0,25xy} + sin^2(xy)$	0,01	5,73	0,09	6,01
6	$5{,}32xy + \sin^3\left(x^2\right)$	0,23	0,26	0,87	0,79
7	$\frac{y + 2}{x^3} - 4{,}8\cos^2(2xy)$	-0,7	1,5	0,83	0,4
8	$\frac{x + y}{\cos x} + e^{2x+y}$	1,8	0,73	0,386	-2,683
9	$sin(2xy) + tg(3x^2 y^2)$	0,263	0,352	0,35	0,0753
10	$5\,ln(2x) - \sqrt{\lvert 2x + 3xy \rvert}$	0,271	- 0,73	0,384	0,236

Лабораторная работа № 4
Формирование структурированных данных

Цель работы: получить навыки формирования структурированных данных с использованием функций библиотеки *numpy*.

Краткие теоретические сведения

Общий обзор составных данных

В Python все типы данных делятся на простые и составные (структурированные). Переменные составных типов данных строятся из наборов значений, к ним относятся:

- списки – *list;*
- кортежи – *tuple*;
- строки – *str;*
- множества – *set*, *frozenset* ;
- словари – *dict*;
- массивы – *array*.

Переменные составных типов делятся на неизменяемые и изменяемые. К неизменяемым типам относятся строки и кортежи, к изменяемым – списки, множества, словари и массивы.

Обзор возможностей библиотеки numpy

При решении задач инженерного и научного направления основными данными составного типа являются массивы данных (одномерные массивы, двумерные массивы, многомерные массивы).

Массив – это последовательность однотипных элементов, снабженных индексами. Отличие списка от массива заключается в том, что данные в списке не обязательно должны быть одного типа. Набор хранящихся в массиве данных должен иметь одинаковый тип. В дальнейшем будут использоваться числовые массивы, хранящие числовые данные. Вектором принято называть одномерный массив. Матрица – это двумерный массив.

Для работы с массивами в Python используется библиотека **numpy**, имеющая следующие основные возможности:

- формирование элементов массивов;
- вычисление математических функций от элементов массивов;
- преобразование массивов;

- выполнение операций с массивами и т. д.

Библиотека **numpy** содержит в своем составе набор модулей и базовых функций. К набору модулей библиотеки **numpy** можно отнести следующие:

- **random** (работа со случайными данными);
- **linalg** (линейная алгебра);
- **fft** (быстрое преобразование Фурье);
- **polynomial** (работа с полиномами) и т. д.

Базовые функции библиотеки можно классифицировать следующим образом:

- математические функции (тригонометрические, экспоненциальные, логорифмические и т. д.);
- функции формирования новых массивов;
- функции поиска суммы, произведения;
- функции поиска минимума, максимума и др.

Ниже приведено описание некоторых приемов работы с библиотекой **numpy.** Следует учитывать, что для всех приведенных ниже примеров библиотека предварительно описана в программе так:

import numpy as np

Для обращения к функциям библиотеки следует использовать префиксную запись, например, **np.arange(3, 10), np.sin(A)** и т. д.

Занесение числовых значений в массив

Занести числа в одномерный и двумерный массивы можно несколькими способами. Существуют:

- непосредственный ввод с использованием списка;
- ввод с использованием диапазона.

Чтобы занести числовые данные в одномерный или двумерный массив с использованием списка, нужно воспользоваться функцией ***array*** библиотеки **numpy**.

Общий вид обращения к функции приведен ниже.

ИмяМассива=array([СписокЭлементов]).

Список в Python можно рассматривать как последовательность данных, каждое их которых имеет номер (индекс). Если список используется для задания массива с помощью функции **array**, то все элементы списка должны иметь числовой тип.

Из числовых значений элементов массива следует создать список, перечислив их в квадратных скобках, разделяя запятой.

Например,

V=np.array([2, 3, 8, 0, 7, 5])

Обращение к элементу одномерного массива выполняется указанием имени массива и номера элемента в массиве в квадратных скобках. Например, **V[2]** – это элемент массива с номером 2, **V[4]** – элемент массива с номером 4.

Нужно отметить, что нумерация элементов массива начинается с 0, поэтому числовым значением **V[0]** из предыдущего примера будет число 2, а для **V[4]** – число 7.

Ниже приведен фрагмент программы, в котором создается одномерный массив списком элементов, выводится в интерактивное окно весь массив и элемент с заданным номером.

Фрагмент программы	Интерактивное окно
A=np.array([2, 5, 9, -4, 12, 3])	
print(A)	**[2 5 9 -4 12 3]**
print(A[3])	**-4**

Для создания двумерного массива необходимо последовательно указать списки строк массива, разделенные запятой.

Общий вид формирования двумерного массива приведен ниже.

ИмяМассива=array([[СписокСтроки1], [СписокСтроки2], …)

Например, для формирования матрицы из двух строк и трех столбцов необходимо задать:

M= np.array ([[3, 1, -3], [2, 10, 1.5]])

Для указания отдельного элемента матрицы после имени матрицы в квадратных скобках указываются два номера: номер строки и номер столбца, отсчет номеров строк и столбцов матрицы начинается с 0.

Например, **M[1, 2]** – это элемент матрицы M в первой строке и втором столбце, его числовое значение равно **1.5**.

Ниже приведен фрагмент программы, в котором создается двумерный массив перечислением элементов, выводится в интерактивное окно весь массив и элемент с заданными номерами.

Фрагмент программы	Интерактивное окно
A=np.array([[2, 5, 9], [-4, 12, 13]]) **print(A)** **print(A[1,2])**	**[[2 5 9]** **[-4, 12, 13]]** **13**

Для создания массивов с использованием диапазона в Python применяются функции **arange** и **linspace.**

Общий вид функции **arange** приведен ниже.

np.arange(xn, xk, dx)

xn – начальное значение диапазона (необязательный параметр);

xk – конечное значение диапазона;

dx – шаг изменения значений диапазона (необязательный параметр).

Для формирования одномерного массива следует задать:

X= np.arange(xn, xk, dx)

В результате будет сформирован одномерный массив, первый элемент которого равен xn, второй – xn+dx, третий – xn+dx+dx и т.д. Последний элемент будет не больше xk-dx для положительного шага dx, и не меньше xk+dx – для отрицательного.

Например, запись вида:

a= np.arange(1, 9, 2)

приведет к формированию массива, содержащего числа

[1, 3, 5, 7].

Если **xn** и **dx** не заданы, то будет сформирован массив элементов, значение первого из которых равно 0, шаг изменения значений элементов равен 1, значение последнего элемента равно **xk-1**.

Например, запись вида:

a= np.arange(6)

приведет к формированию массива, содержащего числа

[0, 1, 2, 3, 4, 5].

С помощью функции **linspace** можно создать одномерный массив, значения элементов которого изменяются в диапазоне от начального до конечного заданное число раз. Общий вид это функции следующий:

np. linspace (xn, xk, n)

xn –начальное значение диапазона;

xk – конечное значение диапазона;

n – количество элементов массива (необязательный параметр).

Для формирования одномерного массива следует задать:

X= np. linspace (xn, xk, n)

В результате будет сформирован одномерный массив, первый элемент которого равен **xn**, последний – **xk**, количество элементов будет равно **n**. Если **n** не задано, то количество элементов в массиве будет равно 50.

Примеры правильной записи задания одномерного массива с использованием равномерного распределения элементов внутри диапазона приведены ниже.

Фрагмент программы	Интерактивное окно
X= np. linspace (-1, 8, 10) **print(X)** **print(X[3])**	**[-1. 0. 1. 2. 3. 4. 5. 6. 7. 8.]** **2.0**
Y= np. linspace (5.5, 8, 6) **print(Y)** **print(Y[2])**	**[5.5 6. 6.5 7. 7.5 8.]** **6.5**

Практическая часть

Задание 1

1.1. Создать одномерный массив X1, содержащий числа из диапазона с заданными пределами изменения xn и xk с шагом dx с помощью функции **arange**. Сформировать новый одномерный массив Y1, содержащий значения функции от элементов исходного массива, в соответствии с видом функции, приведенном в табл. 4.1**.** Вывести в интерактивную область значения X1 и Y1.

1.2. Создать одномерный массив X2, содержащий числа из диапазона с заданными пределами изменения xn и xk и заданным количеством значений внутри диапазона N с помощью функции **linspace**. Сформировать новый одномерный массив Y2, содержащий значения функции от элементов исходного массива, в соответствии с видом функции, приведенном в табл. 4.1**.** Вывести в интерактивную область значения X2 и Y2.

1.3. Вывести в интерактивную область срезы массива X1 и массива Y1 со второго по пятый элементы.

Таблица 4.1

Варианты задания 1

№ вар.	С использованием arange массив Y1	Массив X1			С использованием linspace массив Y2	xn	xk	N
		xn	xk	dx				
1.	$y = \frac{arctg(x)}{1+\sin^2 x}$	2	5	0.3	$y = \frac{1+\sqrt{0{,}5x}}{0{,}5+\sin^2 x}$	2	4	10
2.	$y = \ln(x^2+2x+2)$	-3	0	0.2	$y = e^x \sin x \cos^3 x$	-1	1	11
3.	$y = \frac{\sin^2 x}{\sqrt{x}+x}$	2	5	0.2	$y = \frac{e^{0,1x}+1}{1+\cos^2 x}$	0	3	16
4.	$y = 2x^3 - 6x^2 - 18x + 7$	-2	2	0.4	$y = 2x^3 - 3x^2$	1	2	11
5.	$y = x\sqrt{1+x^2} \cdot \sin x$	1	5	0.4	$y = -\ln(x^2-4x+5)$	1	4	15
6.	$y = e^x(\sin 3x - 3\cos 3x$	1	4	0.2	$y = (x^2-2x)\ln x - \frac{3}{2}x^2 + 4x$	1	4	16
7.	$y = \frac{\sin^3 x}{x+2}$	0	3	0.3	$y = \frac{3x^2+4x+4}{x^2+x+1}$	-1	1	11
8.	$y = \frac{(1+2x)^2}{1.8+\cos^3 x}$	2	4	0.1	$y = \frac{1}{\ln(x^4+4x^3+30)}$	-1	3	12

Задание 2

Постановка задачи. Координата **Y** траектории перемещения захвата робота-манипулятора описывается функцией Y(X)=f(x) в декартовой системе координат.

Найти значение координату X, при которой координата **Y** траектории превысит пороговое значение, равное R. Известно, что координата X изменяется от Xn до Xk (взять не менее 15 значений), функция возрастающая или убывающая в зависимости от задания. Варианты приведены в табл. 4.2.

Таблица 4.2

Варианты задания 2

№	Y(X)	Xn	Xk	Пороговое значение R	
1	$f(x) := \frac{x^3}{3.5} + x^2$	-4	-2,5	0,5	возрастает
2	$f(x) := 3 \cdot x^4 - 8x^3 + 6x$	1	1,8	-2	убывает

Окончание табл. 4.2

№	Y(X)	Xn	Xk	Пороговое значение R	
3	$f(x) := x\cdot\sqrt{2 - 1.8\cdot x^2}$	0	0,6	0,4	возрастает
4	$f(x) := 3x^2 + 38x + 2$	-12	-8	60	убывает
5	$f(x) := (3.5\sin(x - 3.5)) - \ln(x)$	2	4	-5	возрастает
6	$f(x) := \sin(x)^2 - (0.2\cdot\ln(x))$	2	3	0,25	убывает
7	$f(x) := \sin(x - 0.65)^2 - (0.2\cdot x)$	6	6,8	-1	убывает
8	$f(x) := (x - 4.8)\cdot x^{\frac{2}{3}}$	2	3	-4	возрастает
9	$f(x) := \frac{x^3}{3.5} + x^2$	-2	-1	1,2	убывает
10	$f(x) := 3\cdot x^4 - 8x^3 + 6x$	2	2,6	3	возрастает

Лабораторная работа № 5
Обработка графической информации в Python

Цель работы: Получить навыки работы с графической информацией в Python, научиться строить графики функций различных видов, программно редактировать графики.

Краткие теоретические сведения

Обзор видов графической информации

Python позволяет обрабатывать различные виды графической информации:

- построение двумерных графиков в декартовой и полярной системах координат;
- построение графиков n-мерных поверхностей;
- внесение рисунков из других программных систем;
- создание анимации.

В Python основной библиотекой для работы с графической информацией является библиотека **matplotlib**, которая содержит как функции, так и модули, позволяющие строить графики, редактировать их, выполнять сервисные функции для работы с графическими данными (информацией). Библиотека построена на прямой аналогии с

системой компьютерной математики Matlab, поэтому работа с графикой достаточно проста для реализации.

В набор модулей **matplotlib** на уровне сценариев входит модуль **pyplot,** поэтому инструкция, дающая возможность работать с графикой с использованием его функций, может иметь вид:

import matplotlib.pyplot as plt

В этом разделе данная инструкция будет всегда включена в программу, поэтому все инструкции, позволяющие работать с графическими объектами, будут иметь префикс «**plt**».

Любое графическое изображение в Python строится из графических объектов и имеет иерархическую структуру. Объектом наивысшего уровня иерархии в этой структуре является объект **figure** – графическое окно, который, в свою очередь, содержит ряд подчиненных ему объектов. К одному из таких объектов относится объект **axes**, представляющий область рисования в графическом окне **figure** и являющийся сам по себе сложным и многокомпонентным объектом. В его состав включаются графические элементы, вид которых зависит от типа графического объекта.

Графические объекты в Python строятся в графическом окне, одновременно может быть открыто несколько графических окон, каждому из которых присваивается номер. Одно из открытых окон обязательно должно быть текущим (активным) на данный момент работы программы. Для перехода к окну с номером N или открытия нового графического окна необходимо ввести инструкцию:

plt.figure(N)

Кроме того, первое обращение к любой графической инструкции автоматически вызывает появление графического окна, которому присваивается первый номер.

Для того, чтобы графический объект был выведен в графическое окно, необходимо применить функцию:

plt.show( )

Двумерный график функции может быть создан графическими инструкциями:

- **plt.plot –** построение двумерного графика общего вида;
- **plt .scatter –** построение точечного двумерного графика;
- **plt.polar –** построение двумерного графика в полярной системе координат;
- **plt.fill** – построение двумерного графика с заливкой областей;
- и т. д.
-

Инструкция plt.plot

Графическая инструкция **plt.plot** предназначена для построения графиков функций одной переменной в декартовой системе координат и имеет следующие формы:

- **plt.plot (X, Y);**
- **plt. plot (Y);**
- **plt. plot (X, Y, S) ;**
- **plt. plot (X, Y, S, OP).**

График функций строится по ряду точек, соединяя или не соединяя (в зависимости от элементов форматирования) их отрезками прямых.

Инструкция **plt.plot(X,Y)** – строит график функции, координаты точек которой берутся из массивов одинаковой размерности X и Y. Если Y – матрица, то строится семейство графиков по данным, содержащимся в столбцах матрицы.

Инструкция **plt.plot(X,Y,S)** аналогична инструкции **plt.plot(X,Y)**, в которой формат линии графика можно задавать с помощью строковой константы S. Символы, которые могут использоваться в параметре S, приведены в табл. 5.1.

Таблица 5.1

Символы форматирования линий графика

Тип линии		Тип маркера		Цвет линии	
-	Сплошная	**.**	Точка	**y**	Желтый
:	Двойной пунктир	**o**	Окружность	**m**	Фиолетовый
-.	Штрих-пунктир	**x**	Крест	**c**	Голубой
--	Штриховая	**+**	Плюс	**r**	Красный
		*****	Звездочка	**g**	Зеленый
		s	Квадрат	**b**	Синий
		d	Ромб	**w**	Белый
		v	Треугольник	**k**	Черный

Заголовки, текст, легенда

Инструкция **plt.grid()** позволяют задавать построение сетки на поле графика, в качестве параметра функции может быть указана константа True. Ниже приведен фрагмент программы и график функции с нанесенной на него сеткой.

Заголовок графика и надписи осей графика можно вывести с помощью инструкций:

- **plt.title (title) – название графика;**
- **plt.xlabel(xstr) – название оси X;**
- **plt.ylabel(ystr) – название оси Y.**

Здесь

title – символьная константа, содержащая название графика;

xstr – символьная константа, содержащая название оси X;

ystr – символьная константа, содержащая название оси Y.

Идентификацию кривых графика (создание легенды) можно выполнить с использованием инструкции **plt.legend**.

Общий вид инструкции следующий:

plt.legend(Список),

где **Список** – перечень наименований линий графика, организованный по правилам формирования списков в Python.

Например, инструкция

plt.legend(["sin(x)","cos(x)"])

позволяет создать легенду для двух линий графика с соответствующими наименованиями.

Инструкция **plt.scatter(X,Y)** предназначена для представления графика в виде точек без соединения их отрезками прямых, X и Y – массивы координат исходных точек графика.Инструкция **plt.polar(fi,R)** предназначена для построения графиков в полярной системе координат. Здесь **fi** – полярный угол, **R** – полярный радиус.

Практическая часть

Задание 1

В соответствии с видом функции, приведенном в табл. 5.2, вычислить значения функции y=f(x) для значений аргумента x, изменяющегося в интервале от a до b в N заданных точках, значение N должно быть не менее 50. Построить график функции f(x), нанести координатную сетку.

Таблица 5.2

Варианты задания 1

Вариант	Функция y(x)	Пределы изменения аргумента функции
1.	$x^2 \cdot \sin(2x)$	a=5 b=15
2.	$\cos(x) - 2\sin(2x)$)	a=6 b=14
3.	$2\sin(0{,}53x) - \cos^2(x)$)	a=0 b=15
4.	$\sin(0{,}3x) - 1 - 2\cos(0{,}5x)$	a=5 b=35
5.	$\sin(x)\cos(x-3{,}53) - \sin(x) - \sin(x+1{,}87) \cdot \cos(x)$	a=3 b=15
6.	$1+\sin(2x)-2\cos(3x)\sin(x)$	a=4 b=10
7.	$\sin^2(x-0{,}65) - 0{,}2x$	a=0 b=6
8.	$3{,}5\sin(x - 3{,}5) - \ln(x)$	a=1 b=15
9.	$\sin^2(x) - 0{,}2\ln(x)$	a=2 b=10
10.	$1 - \cos(2x) - \ln(x)$	a=2 b=6

Задание 2

Построить на одном поле графики двух функций, промаркировать точки графиков, задать типы линий, подписать оси и весь график, создать легенду, нанести координатную сетку. Варианты задания 2 приведены в табл. 5.3.

Таблица 5.3

Варианты задания 2

№ вар.	Функция	x_n	x_k	Функция	x_n	x_k
1.	$y = \frac{arctg(x)}{1+\sin^2 x}$	2	5	$y = \frac{10+\sqrt{0{,}5x}}{0{,}5+x}$	3	6
2	$y = 50\cos\left(x^2+2x+2\right)$	2	4	$y = e^x \sin x \cos x$	3	6
3	$y = \frac{10\sin^2 x}{\sqrt{x}+x}$	2	5	$y = \frac{e^{0,1x}+1}{1+\cos^2 x}$	0	3
4	$y = x^3 - 12x^2 + 10\sin(10x)$	-2	4	$y = 2x^3 - 3x^2$	2	6
5	$y = x\sqrt{1+x^2} \cdot \sin x$	1	5	$y = \left(3x^2 - 25x + 5\right)$	1	4
6	$y = e^x\left(\sin 3x - 3\cos 3x\right)$	3	5	$y = 100\left(x^2 - 2x\right)\ln x$	1	4
7	$y = \frac{100\sin 5x}{x+2}$	2	4	$y = 3x^2 + 4x + 4$	0	3
8	$y = 5x\sin x + \cos x - \frac{1}{4}x^2$	-2	2	$y = 3\cos^2 x - \cos^3 x$	1	5

Задание 3

Построить график кусочно-непрерывной функции. Пределы изменения аргумента подобрать так, чтобы перекрывались все три диапазона. Нанести координатную сетку, сделать надписи осей и заголовка графика, изменить тип, цвет, линии графика, нанести маркеры на линии графика. . Варианты задания 3 приведены в табл. 5.4.

Таблица 5.4

Варианты задания 3

№	Вид функции	№	Вид функции
1.	$y=\begin{cases}\sqrt{x+100} & \text{если } x>20\\ x & \text{если } 1\le x\le 20\\ 2x^2 & \text{в остальных случаях}\end{cases}$	2.	$y=\begin{cases}2x^2 & \text{если } x>10\\ -20x & \text{если } x\le 2\\ 4x & \text{в остальных случаях}\end{cases}$
3.	$y=\begin{cases}\sqrt{x} & \text{если } x>20\\ \dfrac{1}{x} & \text{если } 1\le x\le 20\\ x+2 & \text{в остальных случаях}\end{cases}$	4.	$y=\begin{cases}\sin 2x & \text{если } x>20\\ \sqrt{x} & \text{если } 1\le x\le 20\\ \cos x & \text{в остальных случаях}\end{cases}$
5.	$y=\begin{cases}\dfrac{1}{x} & \text{если } x>10\\ \sqrt[3]{x} & \text{если } 1\le x\le 8\\ \dfrac{x}{5} & \text{в остальных случаях}\end{cases}$	6.	$y=\begin{cases}15\sin(x) & \text{если } x>3\\ 50\cos(x) & \text{если } x\le 0\\ x^2+2 & \text{в остальных случаях}\end{cases}$
7.	$y=\begin{cases}\dfrac{x^2}{50} & \text{если } x>8\\ -2x & \text{если } x\le 0\\ \sqrt{x} & \text{в остальных случаях}\end{cases}$	8.	$y=\begin{cases}8x & \text{если } x>5\\ x^2 & \text{если } -5\le x\le 5\\ \sin(x) & \text{в остальных случаях}\end{cases}$
9.	$y=\begin{cases}\sqrt{x} & \text{если } x>5\\ \dfrac{1}{x} & \text{если } -10\le x\le -1\\ \cos(x) & \text{в остальных случаях}\end{cases}$	10.	$y=\begin{cases}1-3x & \text{если } x>5\\ x-5\sin(x) & \text{если } -5\le x\le 5\\ x^2 & \text{в остальных случаях}\end{cases}$

Задание 4

Построить график параметрически заданной функции (фигуры Лиссажу). . Варианты задания 4 приведены в таблице 5.5.

Таблица 5.5

Варианты задания 4

N	Функции	tn	tk	dt
	$x(t) := 5\cdot\cos(5t)$ $y(t) := 10\cdot\cos(3\cdot t + 0.5\cdot\pi)$	0	10	0.1
	$x(t) := 2\cdot\cos(5t)$ $y(t) := 10\cdot\cos(7\cdot t + 0.25\cdot\pi)$	0	15	0.1
	$x(t) := 5\cdot\cos(10t)$ $y(t) := 2\cdot\cos(11\cdot t + 0.25\cdot\pi)$	0	20	0.1
	$x(t) := 0.5\cdot\cos(24t)$ $y(t) := 1\cdot\cos(36\cdot t - 0.25\cdot\pi)$	0	10	0.1
	$x(t) := 0.54\cdot\cos(9t)$ $y(t) := 10\cdot\cos(7\cdot t + 0.25\cdot\pi)$	0	25	0.15
	$x(t) := 0.54\cdot\cos(5t)$ $y(t) := 0.12\cdot\cos(20\cdot t + 0.5\cdot\pi)$	0	15	0.05

Лабораторная работа № 6
Прикладные задачи на обработку одномерных массивов

Цель работы: Получить навыки программирования на языке Python прикладных задач по обработке одномерных массивов.

Практическая часть

Задание 1

Постановка задачи. Робот сортирует партию из N деталей круглой формы на конвейере, проводит взвешивание каждой детали и измерение ее диаметра. Если вес детали превышает пороговое значение (PV), деталь считается бракованной. Если диаметр детали превышает пороговое значение (PD), деталь считается бракованной. Если в партии больше 30 % бракованных деталей, то бракуется вся партия.

Вычислить указанные в индивидуальных заданиях (табл. 6.1) параметры проверки качества партии деталей, полученных с конвейере.

Таблица 6.1

Варианты задания 1

№	Основное задание	Пороговые значения
1	Вычислить средний вес годных деталей в партии и найти номер первой встреченной бракованной по диаметру детали. Сформировать массив из бракованных по диаметру деталей.	PV=30 PD=5
2	Вычислить количество годных и количество бракованных деталей в партии, найти номер первой встреченной бракованной по весу детали. Проверить, будет ли данная партия из N деталей бракованной.	PV=20 PD=10
3	Вычислить количество бракованных деталей по двум параметрам одновременно, найти номер первой встреченной бракованной по диаметру детали. Сформировать массив из годных деталей.	PV=50 PD=20
4	Вычислить средний вес бракованных деталей в партии и узнать, чего больше - количества годных деталей или количества бракованных деталей. Проверить, будет ли данная партия из N деталей бракованной.	PV=15 PD=10
5	Вычислить суммарный вес бракованных деталей в партии и узнать, чего больше - бракованных по диаметру деталей или бракованных по весу деталей. Сформировать массив из годных деталей.	PV=25 PD=15
6	Вычислить суммарный вес годных деталей в партии и узнать, чего больше - бракованных по диаметру деталей или бракованных по весу деталей. Сформировать массив из годных деталей.	PV=25 PD=15

Задание 2

Постановка задачи. Траектория движения робота задана параметрически:

x(t)=f1(t)

y(t)=f2(t)

Робот должен остановиться, когда ордината его траектории станет больше заданной Yp(tp).

Необходимо:

1) Построить график траектории движения робота, нанести координатную сетку, подписать сам график и оси графика.

2) Найти координаты точки остановки робота, нанести точку остановки на график.

Индивидуальные варианты заданий приведены в табл. 6.2.

Таблица 6.2.

Варианты задания 2

N	Траектория движения робота x(t)=f1(t) y(t)=f2(t)	t (сек)	Yp(tp)
1	$x(t) := 2\cdot\sin(2\cdot t) \qquad y(t) := \sin(3\cdot t)$	1 - 4	-0.8
2	$x(t) := \frac{-t\cdot\sin(t)}{t^2+1} \qquad y(t) := \frac{\cos(t)\cdot t^2}{t^2+1}$	0 - 9	0.8
3	$x(t) := t\cdot\sin(t) \quad y(t) := t\cdot\cos(t)$	0 - 10	-5
4	$x(t) := 8\cdot\left(\cos(t) - \frac{\cos(4t)}{4}\right)$ $y(t) := 8\cdot\left(\sin(t) - \frac{\cos(4t)}{4}\right)$	0 – 6.5	-8
5	$x(t) := 11\cdot\cos(2t)\cdot(1+\cos(2t))$ $y(t) := 11\cdot\sin(2\cdot t)\cdot(1+\cos(2\cdot t))$	0 - 8	-9

Задание 3

Постановка задачи. Три независимых эксперта проводили измерения шероховатости поверхности металлических образцов, полученных после обработки на станке с заданными параметрами. Результаты измерений представлены в виде матрицы значений независимых переменных и экспериментальных значений шероховатости поверхности. Первый столбец матрицы - номер опыта, второй и третий столбцы – значения независимых переменных: S – подача инструмента в мм/об; m – скорость м/с. В четвертом, пятом и шестом столбцах расположены эмпирические данные шероховатости поверхности, полу-

ченные тремя независимыми экспертами. Пример таблицы измерений приведен в табл. 6.3.

Таблица 6.3

Результаты измерений шероховатости поверхности

№	S	m	t1	t2	t3
1	13	14	11,79	11,23	10,68
2	68	18	61,46	58,54	55,63
3	41	11	34,94	33,27	31,61
4	20	10	33,2	31,96	32,84
5	80	22	58,11	59,13	60,15

Вариант 1. Найти среднее значение шероховатости поверхности для всех опытов по всем измерениям экспертов.
Вариант 2. Найти минимальное значение шероховатости поверхности для всех опытов по всем измерениям экспертов.
Вариант 3. Сформировать массив из средних значений шероховатости поверхности для каждого опыта.
Вариант 4. Сформировать массив из максимальных значений шероховатости поверхности для каждого опыта.
Вариант 5. Сформировать массив из минимальных значений шероховатости поверхности для каждого опыта.
Вариант 6. Найти номер опыта, подачу инструмента и скорость для максимального значения шероховатости поверхности.
Вариант 7. Найти среднее значение шероховатости поверхности для всех опытов по каждому эксперту.
Вариант 8. Найти номер опыта и подачу инструмента для минимального значения шероховатости поверхности.
Вариант 9. Сформировать массив средних значений шероховатости поверхности, измеренных каждым экспертом за все опыты.
Вариант 10. Сформировать массив максимальных значений шероховатости поверхности, измеренных каждым экспертом за все опыты.

Задание 4

Постановка задачи. При обработке сигналов необходимо формировать сигнал с заданными характеристиками из двух исходных сигналов. В задании дан вид двух исходных сигналов как функций времени – S1(t) и S2(t).

Необходимо:

- сформировать результирующий сигнал S3(t), отвечающий заданным критериям на заданном диапазоне изменения времени t;
- вывести графики исходных сигналов в первой области графического окна, график результирующего сигнала по первому критерию – во второй области, график результирующего сигнала по второму критерию в третьей область графического окна (область делятся по горизонтали).

Индивидуальные варианты заданий приведены в табл. 6.4.

Таблица 6.4.

Варианты задания 4

№	Входные сигналы	Диапазон времени	Первый критерий	Второй критерий
1	S1(t)=sin(t) S2(t)=-1.5sin(t)	0 - 14	S3(t) включает только положительные значения	S3(t), если S3(t)меньше 0.5, иначе S3(t)=0.5
2	S1(t)=-1.2cos(2t) S2(t)=1.5cos(2t)	0 - 10	S3(t) включает только отрицательные значения	S3(t), если S3(t) больше -0.5, иначе S3(t)=-0.5
3	S1(t)=sin(t) S2(t)=-sin(t)	0 - 20	Если t меньше 10с, то S3(t) включает только положительные значения, иначе S3(t) включает только отрицательные значения	S3(t), если S3(t) меньше 0.5, иначе S3(t)=0.5
4	S1(t)=cos(t) S2(t)=-cos(t)	0 - 24	Если t меньше 12с, то S3(t) включает только отрицательные значения, иначе S3(t) включает только положительные значения	S3(t), если S3(t) больше -0.8, иначе S3(t)=-0.8

Лабораторная работа №7
Работа со строками, кортежами, списками, словарями

Цель работы: Получить навыки составления алгоритмов обработки структурированных данных: строк, списков, словарей. Научиться составлять программу на языке Pyhon с использованием этих типов данных.

Краткие теоретические сведения

Общий обзор составных данных

Строка – это упорядоченная последовательность символов для хранения текстовой информации. Строковые константы и переменные содержат любой набор символов, заключенный в одинарные или двойные апострофы, например:

A="Гомельский технический университет"

B= ' ФАИС '

D=' Кафедра "Информатика" '

Для создания строки можно использовать один из следующих способов:

1) явное задание строки

Gr="Группа ИП-11"

2) создание строки путем изменения типа

A=str(123.85)

3) ввод строки с клавиатуры

N=input("Задайте строку")

К каждому элементу строки можно обратиться по его номеру, например, **A[3], B[1]**. Нумерация символов начинается с нуля.

len(St) – вычисление длины строки.

Для использования в вычислениях нескольких элементов строк как единого объекта, можно воспользоваться операцию среза (символ «:»).

Общий вид среза для строк:

A[N1:N2:N3]

N1 – начальный индекс;

N2- конечный индекс;

N3- шаг изменения индекса.

По умолчанию начальный индекс 0, конечный индекс – длина массива -1, шаг 1.

Фрагмент программы	Интерактивное окно
A1= 'Привет, студенты 1 курса ' **B1=A1[3:12]** **print(A1)** **print(B1)**	**Привет, студенты 1 курса** **вет, студ**

К строкам применимы следующие операции и функции:

St1+St2 – объединение строк в новую строку;

St*k – многократное (k-кратное) повторение строки;

len(St) – вычисление длины строки.

Кортеж (от англ. *tuple*) – это неизменяемая упорядоченная коллекция объектов произвольного типа. Сами объекты называются *элементами кортежа*, а доступ к ним может быть получен при помощи целочисленного индекса или среза.

Кортеж в Python создается при помощи круглых скобок, внутри которых элементы кортежа перечисляются через запятую. Однако, если кортеж состоит из одного элемента, то запятую после него нужно указывать обязательно, иначе интерпретатор не сможет отличить кортеж от простого значения или выражения.

Фрагмент программы	Интерактивное окно
kor3 = (27, 'ИП-11', 78.5, True)	
print(kor3)	**(27, 'ИП-11', 78.5, True)**
kor4 = (1, 2, 3,)	
print(kor4)	**(1, 2, 3,)**
kor5 = (['abc', (4, 5)], 0.3,)	
print(kor5)	**(['abc', (4, 5)], 0.3)**

Обратиться к элементу кортежа можно по его иомеру, как и к элементу строки, срезы кортежей выполняются аналогично со срезами строк.

Фрагмент программы	Интерактивное окно
kor1 = (1, 2, 3, 4, 5, 6, 7)	
print(kor1[0:2])	**(1, 2)**
print(kor1[2:5])	**(3, 4, 5)**
print(kor1)	**(1, 2, 3, 4, 5, 6, 7)**

При поиске элементов в строках и кортежах можно применять операцию in – проверки вхождения. Пример отработки этой операции для строки приведен ниже.

Фрагмент программы	Интерактивное окно

```
a="группа ИП-11"
b="группа ИП-12"
c=a+' и '+b
print(b)
print(c)
if 'ИП' in c:
    print('подстрока есть в строке')
else:
    print('подстроки нет в строке')
```

```
группа ИП-12
группа ИП-11 и группа ИП-12
подстрока есть в строке
```

Список – это упорядоченная последовательность данных, имеющих одинаковый или различный тип, снабженных индексами (порядковыми номерами).

Список является базовым составным типом данных, для работы со списками нет необходимости подключения дополнительных модулей.

Формирование новых списков можно выполнить несколькими способами:

- формирование перечислением элементов;
- формирование с помощью генераторов (встроенных в список инструкций).

Для создания списка перечислением элементов нужно указать в квадратных скобках перечень элементов списка, разделенный запятыми. Например,

L1 = [2, 5.55, 7, 8, 12.3]

L2 = ["один", "два", 8.44, 250, 777]

Для формирования списков с помощью генераторов нужно указать конструкцию генератора внутри квадратных скобок, например:

Фрагмент программы	Интерактивное окно

```
S3= [2*i+3 for i in range(8)]
print(S3)
```

```
[3, 5, 7, 9, 11, 13, 15, 17]
```

Можно формировать сложные генераторы, например:

```
S=[ [ a,  b ]  for a  in range(3) for b  in range(2)]
S1=[  a**2    for a  in range(10) if a  != 5  ]
```

Результатами работы этих генераторов будут списки вида:

[[0, 0], [0, 1], [1, 0], [1, 1], [2, 0], [2, 1]]

[0, 1, 4, 9, 16, 36, 49, 64, 81]

К каждому элементу списка можно обратиться по его номеру, указав его в квадратных скобках после имени списка, например:

L1[3], L2[5].

Отсчет номеров элементов списка начинается с 0.

Фрагмент программы	Интерактивное окно
L4=[’Гомель’ , ’Витебск’ , 104 , 18]	
print(L4[0])	**’Гомель’**
print(L4[-1])	**18**
L5=[[1,2,3],[4,5,6],[7,8,9]]	
print(L5[1][2])	**6**

Под редактированием списков понимается:

- редактирование элементов списка,
- добавление новых элементов в конец списка,
- вставка новых элементов в список,
- удаление элементов списков.

Для вставки и удаления необходимо применять следующие методы работы со списками:

- **A.append(B)** – добавление элемента B в конец списка A;
- **A.remove(B)** – удаление элемента со значением B из списка A;
- **A.insert(j, B)** – вставка элемента со значением B на позицию j в списке A.
-

Фрагмент программы	Интерактивное окно
S3= [2*i+3 for i in range(8)]	
print(S3)	**[3, 5, 7, 9, 11, 13, 15, 17]**
S3.append(111)	
print(S3)	**[3, 5, 7, 9, 11, 13, 15, 17, 111]**
S3.remove(7)	
print(S3)	**[3, 5, 9, 11, 13, 15, 17, 111]**
S3.insert(1,'t')	
print(S3)	**[3, 't', 5, 9, 11, 13, 15, 17, 111]**

Функция list() позволяет преобразовать любую последовательность, указанную в ее аргументе, в список. Если параметр не указан, то создается пустой список.

list(“ИП-12") - преобразует строку в список [‘И', ‘П', ‘-', ‘1', ‘2']
list() # создается пустой список

Словарь – изменяемый неупорядоченный набор данных типа «ключ:значение». Ключом может быть любой неизменяемый объект, например, число, строка или кортеж. Элементами словаря могут быть объекты произвольного типа данных. Чтобы получить элемент словаря, нужно указать ключ, который использовался при сохранении значения.

Создать словарь можно несколькими способами:

- с помощью функции dict(...);
- с помощью {….}.

dict имеет несколько форматов, основным из которых является следующий:

dict(<Ключ1>=<Значение1>[, ... , <КлючN>=<ЗначениеN>])

Пример:

```
d1 = dict(a=5, b=8)
print(d1)
```

Словари формируются с применением фигурных скобок, например:

```
D= {"k1":10, "k2":5, "k3":2}
```

Заполнить словарь можно поэлементно, при этом ключ указывается внутри квадратных скобок.

```
d = {}      # Создаем пустой словарь
d["a"]=5    # Добавляем ключ "a"
d["b"]=8  # Добавляем ключ "b"
print(d)
```

Результат - **{'b': 8, 'a': 5}**

Доступ к данным, хранящимся в словаре, осуществляется по ключу, а не по номеру, как в списках или строках, например: **D["k1"].**

Используя ключ, можно изменить элемент словаря. Если элемента с указанным ключом в словаре не окажется, то он будет добавлен. Например:

Фрагмент программы	Интерактивное окно
d = { "a": 5, "b": 8 } **d['a']=1000** **d['c']=‘ИП-12'** **print(d)**	**{'a': 1000, 'b': 8, 'c': ‘ИП-12'}**

Практическая часть

Задание 1

Вариант 1

Дан список студентов группы ИП-13 и среднего балла сдачи сессии, например,
Sp=[(‘Иванов’ ,8),(‘Петоров’ , 6),(‘Сидоров’, 5),(‘Зайцев’, 6),
(‘Волков’ , 4), (‘Мышкин’ ,9), (‘Кошкин’, 5)]
Дана зависимость стипендий от среднего балла (оформить как словарь).
4,5,6 -100руб
7,8 -150руб
9, 10 – 200руб
Вывести фамилии студентов и назначенные им стипендии.
Вычислить сумму стипендии на группу.
Сформировать список фамилий студентов и назначенных им стипендий, например:
Sp1=[(‘Иванов’ ,150),(‘Петоров’ , 100),…..]

Вариант 2

Дан список студентов группы ИП-13 и список кортежей оценок по четырем экзаменам сдачи сессии этих студентов, например,
Sp=[‘Иванов’, ‘Петоров’ , ‘Сидоров’, ‘Зайцев’, ‘ Волков', ‘Мышкин’, ‘Кошкин’]
R=[(4,2,6,7), (6,2,6,7), (8,5,8,7), (9,9,6,7), (4,2,4,5), (8,7,6,7), (9,8,8,7)]
Вывести фамилии студентов и средний балл сдачи сессии этими студентами
Сформировать список студентов и средних баллов сдачи сессии, например:
Sp1=[(‘Иванов’ ,8),(‘Петоров’ , 6),…..]
Вычислить средний балл сдачи сессии по всей группе.
Вычислить количество пересдач в группе.

Вариант 3

Дан список студентов группы ИП-13 и список кортежей часов пропусков этих студентов за три месяца, например,
Sp=[‘Иванов’, ‘Петоров’ , ‘Сидоров’, ‘Зайцев’, ‘ Волков', ‘Мышкин’, ‘Кошкин’]

R=[(40,16,20), (20,12,30), (18,10,8), (12,10,6), (40,14,24), (8,18,16), (28,18,18)]
Если количество пропусков у студента больше 40, то нужно отправить письмо родителям, а если количество пропусков у студента больше 60, то студенту нужно готовиться к отчислению.
Вывести фамилии студентов и рядом комментарий - нужно ли писать письмо родителям или готовить студента к отчислению.
Сформировать список фамилий студентов и суммарного количества пропусков студентом за три месяца, например:
Sp1=[(‘Иванов’ ,80),(‘Петоров’ , 60),…..]
Вычислить суммарное количество пропусков в группе.

Вариант 4
Дан список студентов группы ИП-13, список кортежей часов пропусков этих студентов за три месяца и список оценок сдачи сессии студентами, например,
Sp=[‘Иванов’, ‘Петоров’ , ‘Сидоров’, ‘Зайцев’, ‘ Волков', ‘Мышкин’, ‘Кошкин’]
R=[(40,16,20), (20,12,30), (18,10,8), (12,10,6), (40,14,24), (8,18,16), (28,18,18)]
R1=[(4,5,4,7), (4,2,4,5), (8,5,8,7), (9,9,6,7), (4,5,4,2), (8,7,6,7), (4,4,5,4)]
Вывести фамилии студентов и общее количество пропусков этими студентами.
Сформировать список фамилий студентов, суммарного количества пропусков студентом за три месяца и среднего балла сдачи сессии, например:
Sp1=[(‘Иванов’ ,80,4),(‘Петоров’ , 60, 4.5),…..]
Сформировать список студентов, которым нужно будет пересдавать экзамены.

Задание 2

Разработать программу, которая выводит графические объекты разными цветами. Перечень цветов ввести в виде строки номеров цветов. Для выбора цветов использовать словарь соответствия цвета и 16-ричного кода (файл «Таблица кодировки цветов в Python».

Программа должна выполнять следующие функции:
1) предоставить пользователю перечень цветов с номерами (не менее 7 цветов), например:

1 – красный
2 – синий
3 – зеленый
….

2) принять строку ответа пользователя, например:

2357

Это значит, что первый объект должен рисоваться синим цветом, второй зеленым и т.д.

3) построить столько графических объектов (например, отрезков прямых), сколько цветов выбрал пользователь, каждый объект окрашивается выбранным цветом.

Помощь в разработке программы приведена ниже.

Шаг 1. Создаем словарь, в котором ключом будет номер цвета, а значением его 16-ричный код.

s1 = {1:'#FF4500', 2:'#7FFF00',…..}

Можно значением словаря сделать кортеж, содержащий не только 16-ричный код, но и название цвета. Например: 2:('#7FFF00', 'зеленый')

Шаг 2. Вывести пользователю меню номеров и наименований цветов для выбора. Если наименования включены в словарь, то можно просто вывести все элементы словаря, например:

```
for key in s1:
   print(key, s1[key][1])
```

Шаг 3. Принять от пользователя его выбор из нескольких цветов в виде списка (list).

Шаг 4. Создать цикл, в котором применить инструкцию построения графического элемента с указанием опции color, например,

plt.plot([x,5],[1,5],color=s1[i][0])

Эта инструкция строит прямую линию по двум точкам с координатами (x,1),(5,5). Так как **x** изменяется в этом цикле с определенным шагом, а остальные координаты – нет, то получается пучок лучей, сходящихся в одной точке, окрашенных разными цветами.

Помним, что перебрать все элементы списка можно таким циклом (i-переменная цикла, cv1- список)

```
for i in cv1:
```

Графические элементы могут быть одни и те же, но размешенные в разных областях графического окна (plt.subplot), номер графической области может тоже меняться в цикле.

В качестве графических элементов можно выбрать прямые, окружности, прямоугольники, графики математических функций и др.

Задание 3

Варианты 1-4

Постановка задачи. Дана таблица затрат электроэнергии станками одного цеха за квартал на производственном предприятии.

Сформировать словари и списки для хранения исходной информации, вывести исходную информацию, можно в виде таблицы.

Выполнить расчеты по вариантам заданий (табл.7.1.), вывести результаты расчетов в интерактивное окно.

Таблица 7.1

Потребление электроэнергии станочным парком сборочного цеха

Станок	Плановое потребление на квартал	январь	февраль	март
Сверлильный	180	70	50	30
Фрезерный	210	120	70	40
Токарный	326	130	110	80
Расточной	330	140	110	90
Шлифовальный	270	100	80	120

Вариант 1. Найти станки и месяцы, на которых был перерасход электроэнергии.

Вариант 2. Найти станки и месяцы, на которых фактический расход электроэнергии был меньше планового.

Вариант 3. Найти станки, на которых суммарный фактический расход электроэнергии по всем месяцам был больше планового.

Вариант 4. Найти станки, на которых суммарный фактический расход электроэнергии по всем месяцам был меньше планового.

Варианты 5-8

Постановка задачи. Дана таблица загрузки станочного парка одного цеха за квартал на производственном предприятии.

Сформировать словари и списки для хранения исходной информации, вывести исходную информацию, можно в виде таблицы.

Выполнить расчеты по вариантам заданий (табл.7.2.), вывести результаты расчетов в интерактивное окно.

Таблица 7.2

Загрузка станочным парком сборочного цеха в часах

Станок	Плановая загрузка станков на квартал	апрель	май	июнь
Сверлильный	600	182	210	232
Фрезерный	630	160	270	240
Токарный	570	230	210	280
Расточной	660	240	210	190
Шлифовальный	630	200	280	180

Вариант 5. Найти станки и месяцы, на которых было превышено плановое время загрузки станков.

Вариант 6. Найти станки и месяцы , на которых фактическое время загрузки станков было меньше планового.

Вариант 7. Найти станки, на которых суммарное фактическое время загрузки по всем месяцам было больше планового.

Вариант 8. Найти станки, на которых суммарное фактическое время загрузки по всем месяцам было меньше планового.

Лабораторная работа № 8
Вычисление интегралов, производных, решение уравнений и систем

Краткие теоретические сведения

Обзор функций библиотеки scipy, применяемых для реализации численных методов

Функции библиотеки **scipy** являются расширением библиотеки **numpy**. Эта библиотека предназначен для выполнения научных и инженерных расчетов и содержит модули и функции для:

- оптимизации;
- интегрирования;
- специальных функций;
- обработки сигналов;
- обработки изображений;
- генетических алгоритмов;
- решения обыкновенных дифференциальных уравнений и т.д**.**

В данном разделе будут рассматриваться функции следующего набора модулей **scipy:**

- **scipy.integrate** – содержит функции вычисление интегралов;
- **scipy.linalg** – содержит функции линейной алгебры;
- **scipy.optimize** – содержит функции оптимизации и решения уравнений;
- **scipy.misc** – предназначен для обработки изображений, содержит функцию вычисления производной.

Функции вычисления определенных интегралов

В Python вычисление определенного интеграла можно выполнить с использованием базовых функций из различных модулей. Перечень функций приведен ниже:

- функция **trapz()** входит в состав библиотеки **numpy** и основана на методе трапеций;
- функция **quad()** входит в состав модуля **scipy.integrate** и использует для вычисления определенного интеграла алгоритма квадратурных формул;
- функция **integrate()** входит в состав библиотеки **sympy** и вычисляет определенный интеграл сначала в аналитическом, а затем в численном виде.

Функция **trapz()** имеет следующий общий вид:

np.trapz(y, x=None, dx=1.0)

где **x** – одномерный массив значений аргумента подынтегральной функции на отрезке интегрирования (необязательный параметр);

y – одномерный массив значений, полученных при вычислении подынтегральной функции для элементов вектора **x**, если он указан;

dx – задает шаг интегрирования, если параметр x не указан**.**

Для вычисления определенного интеграла с использованием алгоритма квадратурных формул предназначена функция:

quad(f, a, b)

где **f** – имя подынтегральной функции;

a, b – пределы интегрирования.

Эта функция входит в состав модуля **scipy.integrate**, поэтому перед использованием она должна бы подключена к программе с помощью инструкции:

from scipy.integrate import quad

Результатом работы функции является кортеж **(integral,err),** содержащий численное значение определенного интеграла и значение абсолютной погрешности его вычисления. По умолчанию значение абсолютной погрешности равно 1.49e-08.

Для вычисления определенного интеграла с использованием модуля символьных преобразований предназначена функция:

integrate(f,(x,a,b))

где **f** – символьное выражение, которое будет интегрироваться;

x – переменную интегрирования;

a, b – пределы интегрирования.

Для вычисления определенного интеграла в числовом виде в качестве параметра используется кортеж, состоящий из имени переменной и ее нижнего и верхнего предела. Если второй аргумент – имя, то вычисляется неопределенный интеграл, т.е. первообразная подынтегральной функции.

Эта функция входит в состав библиотеки **sympy,** поэтому перед использованием она должна бы подключена к программе с помощью инструкции:

from sympy import *

Для вычисления производной в точке в Python используется функция **derivative** модуля **scipy.misc.** Производная вычисляется в численном виде при конкретном числовом значении аргумента. Общий вид функции вычисления производной таков:

derivative(y, x, dx=1e-6)

где **x** – значение аргумента функции от которой нужно взять производную;

y – имя функции, производную от которой нужно вычислить;

dx – точность вычисления.

Функции решения нелинейных уравнений в Python

Для решения нелинейных уравнений в Python используются следующие функции библиотеки **scipy**:

fsolve – позволяет найти корень уравнения f(x)=0;

root_scalar – позволяет найти корень уравнения f(x)=0 (f(x) - скалярная функция);

bisect – позволяет найти корень уравнения f(x)=0 методом половинного деления;

newton – позволяет найти корень уравнения f(x)=0, используя метод Ньютона-Рафсона ;

brentq – позволяет найти корень уравнения f(x)=0 на заданном интервале изоляции.

Чаще всего при решении инженерных задач для поиска корня уравнения используется функция **fsolve**, имеющая следующий общий вид:

fsolve(f, x0) ,

где **x0 –** начальное приближение корня,

f – функция, описывающая левую часть уравнения **f(x)=0**.

Функция **fsolve** определена в модуле **scipy.optimize**, поэтому для ее использования в программе должна применяться инструкция:

from scipy.optimize import fsolve

Функция левой части уравнения может быть задана с использованием lambda-функции языка Python, тогда инструкции программы примут следующий вид:

```
F=lambda x: np.sin(2*x)-np.cos(3*x**2)-np.sin(3*x)
k=fsolve(F,7)
print(k)
```

Для решения систем нелинейных уравнений можно также использовать функцию

fsolve(f,x0),

где **x0 –** список начальных приближений для неизвестных системы,

f – функция, определяющая левые части систему нелинейных уравнений.

Функция f может быть описана с помощью инструкции **def f(x)**, например:

```
def f(x):
    z=array([0,0],float)
    z[0]=x[0]+x[1]-2
    z[1]=x[0]*x[1]-x[1]**2
    return z
```

Выходным параметром функции будет одномерный массив z, который при вызове функции рассчитывает значения левых частей системы нелинейных уравнений при заданных значениях входного одномерного массива x. Решение системы нелинейных уравнений можно получить с помощью инструкции:

x=fsolve(f,[1,1]).

Здесь [1,1] – это список начальных числовых значений для корней системы.

При описании левых частей системы можно пользоваться оператором присваивания вида:

```
def F(X):
    x,y=X
```

Тогда вид системы выглядит естественным, и нет необходимости выполнять замену переменных.

Решение систем линейных уравнений в Python

С помощью функций модулей **numpy.linalg** и **scipy.linalg**, которые реализуют основные операции линейной алгебры, можно решить системы линейных уравнений матричным методом. Для этих целей будут использованы базовые функции вычисления обратной матрицы **inv** и матричного умножения **dot** модуля **numpy.linalg**, а также функция **solve** того же модуля.

Пусть задана система линейных алгебраических уравнений в матричном виде: **AX=B**, где A – квадратная матрица коэффициентов линейных уравнений системы (основная матрица), B – столбец свободных членов.

Решение системы может быть получено с использованием следующих инструкций:

Z=np.linalg.inv(A) – вычисляет обратную матрицу Z;

X=np.dot(Z,B) – вычисляет одномерный массив корней системы линейных уравнений X.

Следует отметить, что одномерный массив **B** должен представлять собой столбец матрицы, поэтому задать его можно, например, так:

```
B = np.array([[5.], [6.]])
```

Второй способ решения систем линейных уравнений заключается в использовании функции **solve** модуля **numpy.linalg.** Общий вид функции следующий:

solve(A, B)

A и B имеют то же самое смысловое назначение, что и в предыдущем примере.

Практическая часть

Задание 1

Заготовка для детали описывается функцией y(x), функциями x=0 и x=b и осью x. Дана норма затрат металла на единицу площади.

Составить и отладить программу для того, чтобы определить площадь заготовки и количество металла на изготовление N заготовок. Индивидуальные варианты задания 1 приведены в табл. 8.1.

Дать графическую интерпретацию площади заготовки, например, как на рис. 8.1.

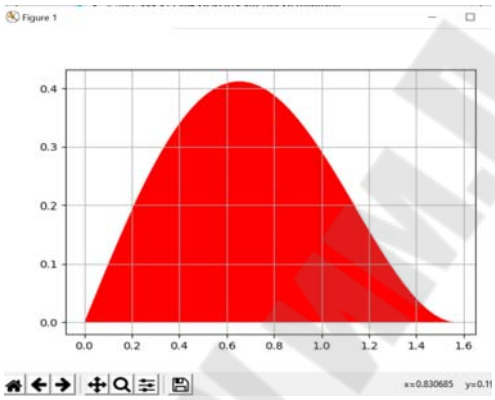


Рис. 8.1. Графическая интерпретация площади заготовки

Таблица 8.1

Варианты задания 1

	функция y(x)	**a=0 b=**	**Норма металла на заготовку**	**Количество заготовок**
1	$y(x) := x \cdot \sin(x) + 2x$	4	0.06	1200
2	$y(x) := -x \cdot \cos(x + 3) + 2.5$	2	0.22	1600
3	$y(x) := x \cdot e^{-x} + 2 \cdot \sin(x)$	2	0.5	2500
4	$y(x) := \sin(x)^2 + 2$	4	0.08	3500
5	$y(x) := -\cos(x)^2 + 1.4$	4	0.25	2200
6	$y(x) := x \cdot \sin\left(x^2\right) + 2.2$	2	0.35	1850
7	$y(x) := x \cdot \cos\left(x^2\right) + 3$	1.7	0.055	2300
8	$y(x) := 5.5x^2 \cdot e^{-2x}$	4.4	0.73	1250

Задание 2

Варианты 1и 2

Условие задачи. Транспортный робот находится в точке с координатами (x0, y0) и должен попасть в точку с координатами (xf, yf) по одной из двух траекторий. Найти кратчайшее расстояние, которое пройдет робот до финишной точки. Дать графическую интерпретацию результатов.

Исходные данные:

- функция траектории 1;
- функция траектории 2;
- координата x стартовой точки
- координата x финишной точки;
- шаг dx, с которым изменяется координата x робота.

Варианты исходных данных приведены в табл. 8.2.

Математическая формула для вычисления длины дуги имеет вид:

$$L = \int_{x_0}^{x_1} \sqrt{1 + (y'(x))^2}\, d\,x$$

Таблица 8.2

Варианты задания 2

	функция траектории 1	**функция траектории 2**	**x0**	**xf**	**dx**
1	$f1(x) := -\left(3\cdot x^4 - 8x^3 + 6x\right)$	$f2(x) := \left(3.06\cdot x^2 - 8x^3 + 6x\right)$	0	1.1	0.01
2	$f1(x) := x\cdot\sqrt{2 - 1.2\cdot x^2}$	$f2(x) := x^2\cdot\sqrt{2 - 1.2\cdot x}$	0	1	0.01

Варианты 3и 4

Условие задачи. Два транспортных робота находится в стартовых точках с разными координатами (x0, y10) и(x0, y20) и должны попасть в точку с координатами (xf, yf). Найти расстояние, пройденное каждым роботом до финишной точки и определить, какой из роботов придет к финишу первым, если они движутся с одинаковой скоростью. Дать графическую интерпретацию результатов.

Исходные данные:

- функция траектории робота1;
- функция траектории робота2;
- координата x стартовой точки

- координата x финишной точки;
- шаг dx, с которым изменяется координата x роботов.

Варианты исходных данных приведены в табл. 8.3.

Математическая формула для вычисления длины дуги имеет вид:

$$L = \int_{x_0}^{x_1} \sqrt{1 + (y'(x))^2}\, d\,x$$

Таблица 8.3

Варианты задания 2

	функция траектории робота1	**функция траектории робота2**	**x0**	**xf**	**dx**
3	$f1(x) := \frac{x^3}{1.6} + 1.2x^2$	$f2(x) := 5.8x + 1.95 \cdot x^2$	-3.5	0	0.01
4	$f1(x) := (\sin(x) - 4.7) \cdot x^{\frac{2}{3}}$	**f2(x)=5sin(x-3.1)**	6	0.56	0.02

Задание 3

Разработать программу, вычисляющую корень уравнения с помощью функции **fsolve** и методом бисекции**,** сравнить полученные результаты.

Доказать графически, что корень найден правильно. Варианты исходных данных приведены в табл. 8.4.

Таблица 8.4.

Варианты задания 3

Вариант	**Уравнение**
1.	$x^4 - 5.8x^3 - 4.2x^2 - 18.6x - 3.6 = 0$ a=5 b=8
2.	$x^4 - 2.5x^3 - 2.54x^2 + 6.3x + 1.38 = 0$ a= -2,5 b= -1
3.	$x^3 + 4x^2 - 6x + 2 = 0$ a= -6 b= -3,5
4.	$2x^3 + 5.8x^2 - 2x + 12 = 0$ a= -5 b= -2
5.	$x^4 - 1.8x^3 - 2.8x^2 + 2.4x + 0.6 = 0$ a= 1,3 b= 3
6.	$x^4 - 0.2x^3 - 7.07x^2 + 8.73x - 1.98 = 0$ a= -5 b= -1
7.	$0{,}5x^4 - 1.1x^3 - 7x^2 + 13.7x - 6.6 = 0$ a= 3 b= 5
8.	$0{,}8x^4 - 12.8x^3 + 2.15x^2 - 2.8x + 1.15 = 0$ a= 15 b= 18
9.	$-0{,}5x^4 + 1.3x^3 - 1.3x^2 + 1.3x + 2.3 = 0$ a= 2 b= 5

Содержание

Трохова Татьяна Анатольевна

ЯЗЫК ПРОГРАММИРОВАНИЯ PYTHON

Практикум
по выполнению лабораторных работ
для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения

Подписано к размещению в электронную библиотеку
ГГТУ им. П. О. Сухого в качестве электронного
учебно-методического документа 17.04.25.

Рег. № 146Е.
http://www.gstu.by