

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Минцаев Магиред Шаралович

Должность: Ректор

Дата подписания: 04.10.2023 17:17:04

Уникальный программный ключ:

236bcc35c296f119d6aafdc22836b21db52dbc07971a86865a5825f91a4504cc

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ГРОЗНЕНСКИЙ ГОСУДАРСТВЕННЫЙ НЕФТЯНОЙ

ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

имени академика М.Д. Миллионщикова

Кафедра «Информационные технологии»

Д.А. Мачуева, Л.С. Умарова

**Методические указания к выполнению лабораторных работ
по дисциплине «Объектно-ориентированное программирование»**

Направление подготовки

09.03.02 Информационные системы и технологии

Направленность (профиль)

«Информационные системы и технологии»

«Информационные технологии в образовании»

«Информационные технологии в дизайне»

Квалификация

бакалавр

Грозный 2023

Содержание

Введение.....	3
Лабораторная работа 1. Введение в язык программирования Python. Ввод и вывод данных. Математические операции в Python	4
Лабораторная работа 2. Условные операторы ветвления	9
Лабораторная работа 3. Работа с циклами в Python.....	11
Лабораторная работа 4. Работа со строками.....	12
Лабораторная работа 5. Операции над списками в Python	15
Лабораторная работа 6. Функции и процедуры	19
Лабораторная работа 7. Работа с двумерными массивами.....	22
Лабораторная работа 8. Создание классов и объектов в Python. Конструктор класса.....	25
Лабораторная работа 9. Наследование. Множественное наследование	27
Лабораторная работа 10. Полиморфизм в Python	28
Лабораторная работа 11. Примеры композиции классов.....	29
Лабораторная работа 12. Создание и использование абстрактного метода	30
Лабораторная работа 13. Создание и использование статического метода	31
Использованные источники	34
Список литературы	34

Введение

Цель преподавания дисциплины «Объектно-ориентированное программирование» состоит в углублении студентами, получающими квалификацию бакалавра, знаний и навыков в области создания приложений, ознакомлении с принципами объектно-ориентированного подхода к проектированию и разработке программ.

Задачами дисциплины являются: изучение теоретических основ современного объектно-ориентированного программирования (ООП) и получение практических навыков применения парадигмы ООП при разработке сложных программ.

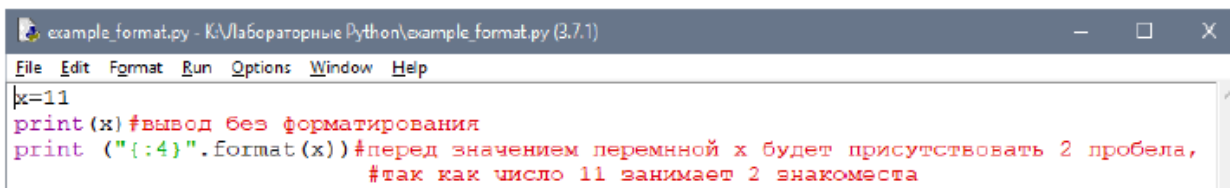
Данное методическое пособие содержит подробное описание лабораторных работ, каждая из которых направлена на освоение и закрепление определенной темы:

- Введение в язык программирования Python. Ввод и вывод данных. Математические операции в Python.
- Условные операторы ветвления.
- Работа с циклами в Python.
- Работа со строками.
- Операции над списками в Python.
- Функции и процедуры.
- Работа с двумерными массивами.
- Создание классов и объектов в Python. Конструктор класса.
- Наследование. Множественное наследование.
- Полиморфизм в Python.
- Примеры композиции классов.
- Создание и использование абстрактного метода.
- Создание и использование статического метода.

Лабораторная работа 1. Введение в язык программирования Python.

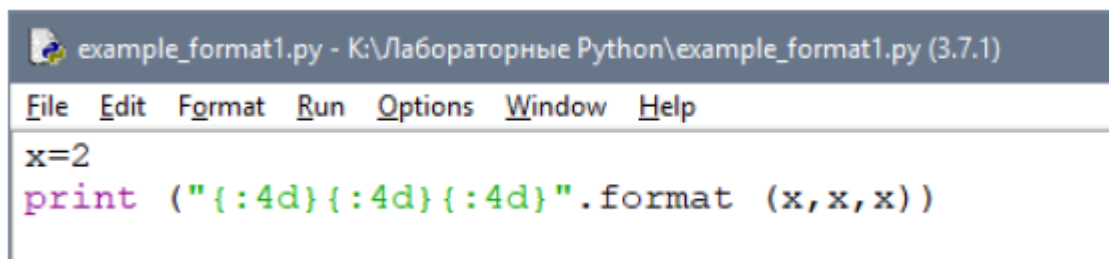
Ввод и вывод данных. Математические операции в Python

Пример форматированного вывода в Python:

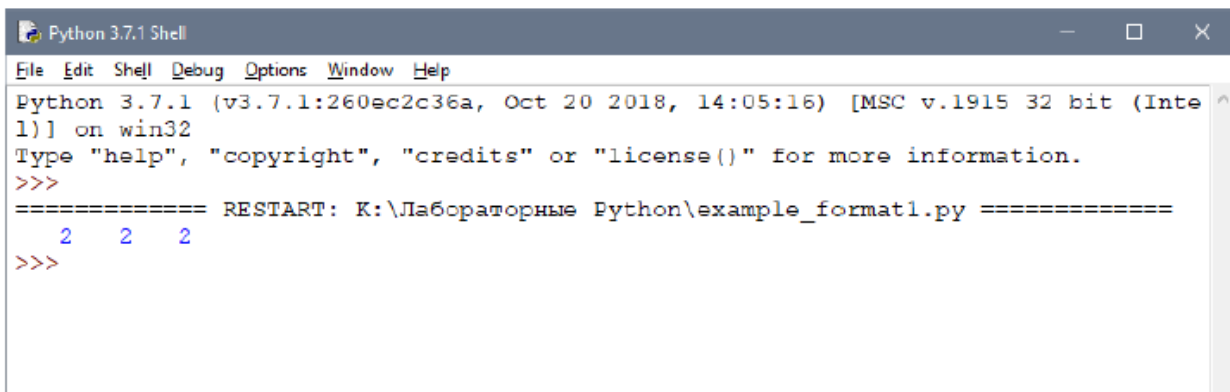


```
example_format.py - K:\Лабораторные Python\example_format.py (3.7.1)
File Edit Format Run Options Window Help
x=11
print(x) #вывод без форматирования
print("{:4}".format(x)) #перед значением переменной x будет присутствовать 2 пробела,
                        #так как число 11 занимает 2 знакоместа
```

С несколькими аргументами:



```
example_format1.py - K:\Лабораторные Python\example_format1.py (3.7.1)
File Edit Format Run Options Window Help
x=2
print("{:4d}{:4d}{:4d}".format(x,x,x))
```



```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.1915 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: K:\Лабораторные Python\example_format1.py =====
      2   2   2
>>>
```

Используемые обозначения для задания формата:

Тип	Значение
'd', 'i', 'u'	Десятичное число.
'o'	Число в восьмеричной системе счисления.
'x'	Число в шестнадцатеричной системе счисления (буквы в нижнем регистре).
'X'	Число в шестнадцатеричной системе счисления (буквы в верхнем регистре).
'e'	Число с плавающей точкой с экспонентой (экспонента в нижнем регистре).
'E'	Число с плавающей точкой с экспонентой (экспонента в верхнем регистре).

'f', 'F'	Число с плавающей точкой (обычный формат).
'g'	Число с плавающей точкой. с экспонентой (экспонента в нижнем регистре), если она меньше, чем -4 или точности, иначе обычный формат.
'G'	Число с плавающей точкой. с экспонентой (экспонента в верхнем регистре), если она меньше, чем -4 или точности, иначе обычный формат.
'c'	Символ (строка из одного символа или число - код символа).
's'	Строка.
'%'	Число умножается на 100, отображается число с плавающей точкой, а за ним знак %.

Пример ввода и вывода данных:

```
a=input('Введите ваши фамилию, имя, отчество ')
b=input('Сколько вам лет? ')
c=input('Где вы живёте? ')
print('Ваше имя ',a)
print('Ваш возраст ',b)
print('Вы живете в ',c)
```

```
Введите ваши фамилию, имя, отчество Иванов Иван Иванович
Сколько вам лет? 15
Где вы живёте? Уссурийск
Ваше имя Иванов Иван Иванович
Ваш возраст 15
Вы живете в Уссурийск
```

Простые арифметические действия над числами в Python:

```
Python 3.4.1: example_prost_math.py - F://Лабораторные Python/example_prost_math.py
File Edit Format Run Options Windows Help
#простейшие математические операции
x=5
y=6
print('x = ',x)
print('y = ',y)
z=x+y
print('z = ',z)
z=x-y
print('z = ',z)
z=x*y
print('z = ',z)
z=x/y
print('z = ',z)
```

```

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
x = 5
y = 6
z = 11
z = -1
z = 30
z = 0.8333333333333334
>>> |

```

Математические операции над целыми числами:

<code>x // y</code>	Получение целой части от деления
<code>x % y</code>	Остаток от деления
<code>-x</code>	Смена знака числа
<code>abs(x)</code>	Модуль числа
<code>divmod(x, y)</code>	Пара (<code>x // y</code> , <code>x % y</code>)
<code>x ** y</code>	Возведение в степень

Пример:

```

x = 5
y = 2
z = 3
x+y = 7
x-y = 3
x*y = 10
x/y = 2.5
x//y = 2
x%y = 1
-x = -5
abs(-x) = 5
divmod(x, y) = (2, 1)
x**y = 25

```

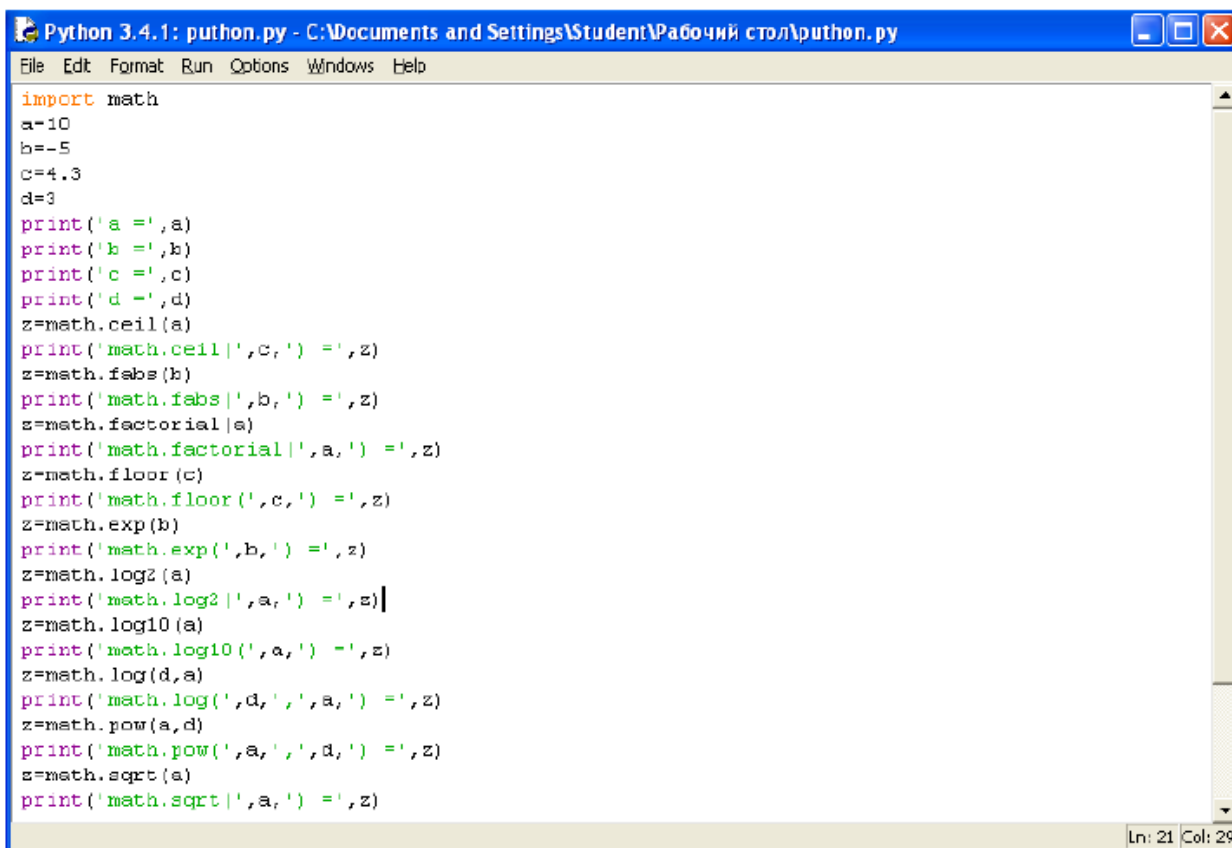
Функции модуля math:

<code>math.ceil(x)</code>	Возвращает ближайшее целое число большее, чем <code>x</code>
<code>math.fabs(x)</code>	Возвращает абсолютное значение числа <code>x</code>
<code>math.factorial(x)</code>	Вычисляет факториал <code>x</code>
<code>math.floor(x)</code>	Возвращает ближайшее целое число меньшее, чем <code>x</code>
<code>math.exp(x)</code>	Вычисляет e^{**x}
<code>math.log2(x)</code>	Логарифм по основанию 2
<code>math.log10(x)</code>	Логарифм по основанию 10
<code>math.log(x, base)</code>	По умолчанию вычисляет логарифм по основанию e , дополнительно можно указать основание логарифма
<code>math.pow(x, y)</code>	Вычисляет значение <code>x</code> в степени <code>y</code>
<code>math.sqrt(x)</code>	Корень квадратный от <code>x</code>

Тригонометрические функции:

math.cos(x)	Возвращает cos числа X
math.sin(x)	Возвращает sin числа X
math.tan(x)	Возвращает tan числа X
math.acos(x)	Возвращает acos числа X
math.asin(x)	Возвращает asin числа X
math.atan(x)	Возвращает atan числа X

Пример применения:



```
Python 3.4.1: python.py - C:\Documents and Settings\Student\Рабочий стол\python.py
File Edit Format Run Options Windows Help
import math
a=10
b=-5
c=4.3
d=3
print('a =', a)
print('b =', b)
print('c =', c)
print('d =', d)
z=math.ceil(a)
print('math.ceil(', c, ') =', z)
z=math.fabs(b)
print('math.fabs(', b, ') =', z)
s=math.factorial(a)
print('math.factorial(', a, ') =', z)
z=math.floor(c)
print('math.floor(', c, ') =', z)
z=math.exp(b)
print('math.exp(', b, ') =', z)
z=math.log2(a)
print('math.log2(', a, ') =', z)
z=math.log10(a)
print('math.log10(', a, ') =', z)
z=math.log(d, a)
print('math.log(', d, ', ', a, ') =', z)
z=math.pow(a, d)
print('math.pow(', a, ', ', d, ') =', z)
s=math.sqrt(a)
print('math.sqrt(', a, ') =', z
Ln: 21 Col: 29
```

ЗАДАНИЯ

Задача 1

Вводятся два целых числа. Найти и вывести среднее арифметическое этих чисел.

Задача 2

Требуется значение функции $y = \frac{x^3}{2(x+5)}$ при произвольном отрицательном и положительном значениях x .

Задача 3

Составьте программу решения квадратного уравнения (при условии, что дискриминант $D > 0$).

Задача 4

Дается трехзначное число. Требуется разбить его на порядки: выделить сотни, десятки, единицы.

Задача 5

Даны два угла треугольника (в градусах). Определить, существует ли такой треугольник, и если да, то будет ли он прямоугольным.

Задача 6

По введенной температуре требуется определить состояние воды (твердое, жидкое, газообразное).

Задача 7

Требуется определить, является ли введенное число четным.

Задача 8

Выполните в среде Python следующие вычисления:

$$5 + 3^2 - 18 \cdot 4 + \frac{120}{5}$$

$$25,5 + 4,2^2 - \frac{125}{5}$$

Задача 9

Напишите программу, осуществляющую поочередно ввод двух чисел. Далее с помощью математических и арифметических действий осуществите операции:

1. Возведение первого введенного числа в степень второго введенного числа;
2. Результат возведения в степень поделите на первое введенное число.
3. Осуществите вывод на экран остатка от деления.

Задача 10

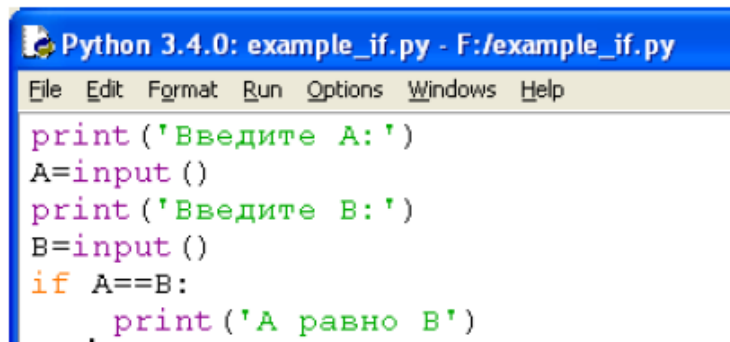
Напишите программу, осуществляющую ввод двух чисел. С помощью математических операторов осуществите операцию деления двумя способами (обычное деление, целочисленное деление). Выведите результат.

Задача 11

Напишите программу, осуществляющую ввод дробного и целого чисел. Далее с помощью арифметических действий найдите остаток от деления дробного числа на целое. Выполните вывод результата.

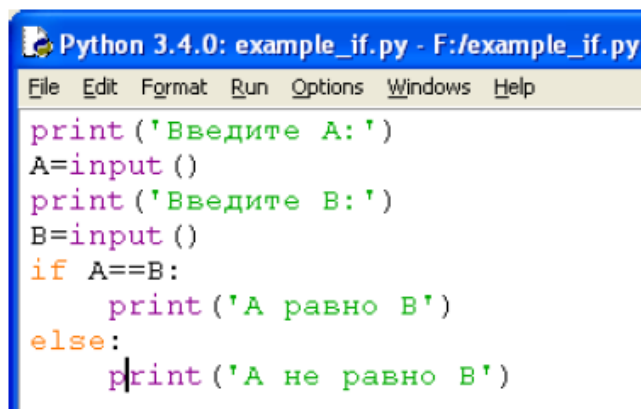
Лабораторная работа 2. Условные операторы ветвления

Пример программы с условным оператором if в Python:



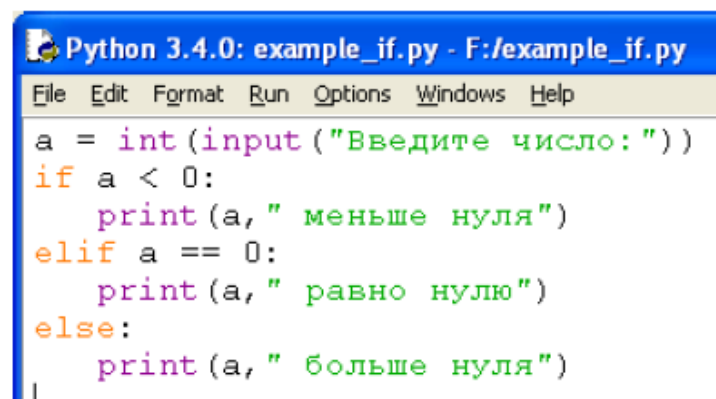
```
Python 3.4.0: example_if.py - F:/example_if.py
File Edit Format Run Options Windows Help
print ('Введите A:')
A=input ()
print ('Введите B:')
B=input ()
if A==B:
    print ('A равно B')
```

Пример конструкции if-else:



```
Python 3.4.0: example_if.py - F:/example_if.py
File Edit Format Run Options Windows Help
print ('Введите A:')
A=input ()
print ('Введите B:')
B=input ()
if A==B:
    print ('A равно B')
else:
    print ('A не равно B')
```

Конструкция if-elif-else:



```
Python 3.4.0: example_if.py - F:/example_if.py
File Edit Format Run Options Windows Help
a = int(input ("Введите число:"))
if a < 0:
    print (a, " меньше нуля")
elif a == 0:
    print (a, " равно нулю")
else:
    print (a, " больше нуля")
|
```

```

Введите число:41
41 больше нуля
>>> ===== RESTART =
>>>
Введите число:-5
-5 меньше нуля
>>> ===== RESTART =
>>>
Введите число:0
0 равно нулю
>>>

```

Пример:

```

#нахождение минимального из 3-х чисел
a=input('Введите целое число \n')
b=input('Введите целое число \n')
c=input('Введите целое число \n')
if a<b:
    if a<c:
        y=a
    else:
        y=c
else:
    if b<c:
        y=b
    else:
        y=c
print('Минимальное:',y)

```

```

Введите целое число
2
Введите целое число
5
Введите целое число
1
Минимальное: 1

```

ЗАДАНИЯ

Задача 1

Напишите программу, которая запрашивает у пользователя число. Если оно больше нуля, то в ответ на экран выводится число 1. Если введенное число не является положительным, то на экран должно выводиться -1.

Задача 2

Необходимо ввести с клавиатуры два вещественных числа и определить наибольшее из них.

Задача 3

Напишите программу, которая выводит на экран максимальное из четырех целых чисел, введенных с клавиатуры.

Лабораторная работа 3. Работа с циклами в Python

Пример программы с циклом while:

```
#!/ Программу по вычислению факториала
number = int(input("Введите число: "))
i = 1
factorial = 1
while i <= number:
    factorial *= i
    i += 1
print("Факториал числа", number, "равен", factorial)
```

Пример:

```
n = int(input('Введите целое число не меньше 2\n'))
i = 2
while n%i != 0:
    i+=1
print('наименьший натуральный делитель:', i)
```

Введите целое число не меньше 2

49

наименьший натуральный делитель: 7

Пример цикла for в Python:

```
n=int(input('Введите количество элементов последовательности: '))
x=1
s=0
print(x)
for i in range(n):
    s+=x
    x/=-2
    print(x)
print('Сумма ряда:', s)
```

ЗАДАНИЯ

Задача 1

Найдите сумму ряда чисел от 1 до 100. Полученный результат выведите на экран.

Задача 2

Вводится переменная a . Вычислите значение $a!$ (факториал)

Задача 3

Выведите на экран столько элементов ряда Фибоначчи, сколько указал пользователь. Например, если на ввод поступило число 6, то вывод должен содержать шесть первых чисел ряда Фибоначчи: 1 2 3 5 8 13.

Задача 4

Написать программу для игры: загадывается число (использовать функцию `random`). Пользователю предлагается угадать число. Если пользователь не угадывает, то ему предлагается угадать число снова и выдается подсказка, что число больше или меньше введенного. Так бесконечно, пока пользователь не введет слово `exit`. Бесконечный цикл организовать через «`while True:`».

Задача 5

Вводится два целых числа, найдите их наибольший общий делитель.

Примечание. Наибольший общий делитель (НОД, англ. `GCD`) – это наибольшее число, на которое нацело делятся два заданных числа.

Задача 6

В программе генерируется случайное целое число от 0 до 100. Пользователь должен его отгадать не более чем за 10 попыток. После каждой неудачной попытки должно сообщаться – больше или меньше введенное пользователем число, чем то, что загадано. Если за 10 попыток число не отгадано, то вывести загаданное число.

Лабораторная работа 4. Работа со строками

Функции и методы работы со строками:

Функция или метод	Назначение
$S1 + S2$	Конкатенация (сложение строк)
$S1 * 3$	Повторение строки
$S[i]$	Обращение по индексу
$S[i:j:step]$	Извлечение среза

len(S)	Длина строки
S.join(список)	Соединение строк из последовательности str через разделитель, заданный строкой
S1.count(S[, i, j])	количество вхождений подстроки s в строку s1. Результатом является число. Можно указать позицию начала поиска i и окончания поиска j
S.find(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или -1
S.index(str, [start],[end])	Поиск подстроки в строке. Возвращает номер первого вхождения или вызывает ValueError
S.rindex(str, [start],[end])	Поиск подстроки в строке. Возвращает номер последнего вхождения или вызывает ValueError
S.replace(шаблон, замена)	Замена шаблона
S.split(символ)	Разбиение строки по разделителю
S.upper()	Преобразование строки к верхнему регистру
S.lower()	Преобразование строки к нижнему регистру

Демонстрация использования:

```

example_string.py - К:\Лабораторные Python\example_string.py (3.7.1)
File Edit Format Run Options Window Help
s1="Пропаганда"
s2="Сенсация"
s3="Сенсация*Сенсация*Сенсация*Сенсация"
s4='ОхОхОхАх'
print('s1 = ',s1)
print('s2 = ',s2)
print('s3 = ',s3)
print('s4 = ',s4)
print('s1+s2 = ',s1+s2) #сложение двух строк
print('s1*3 = ',s1*3) #умножение строки на 3, т.е.строка выведется 3 раза
print('s1[2] = ',s1[2]) #вывод элемента строки s1 с индексом 2
print('s1[2,4] = ',s1[2:4]) #извлечение среза строки s1 начиная с индекса 2
#и заканчивая индексом 4
print('s3.count = ',s3.count(s2)) #количество вхождений подстроки s2 в s3,
#в результате выведется число
print('s1.find('a') = ',s1.find('a')) #поиск подстроки 'a' в строке s1
#результатом будет номер первого вхождения
print('s1.index('n') = ',s1.index('n'))#поиск подстроки 'n' в строке s1
#результатом будет номер первого вхождения
print('s1.rindex('d') = ',s1.rindex('d'))#поиск подстроки 'a' в строке s1
#возвращает номер последнего вхождения
print('s4.replace('Ох','Ах',2) = ',s4.replace('Ох','Ах',2))#замена шаблона. Строка 'Ох' - это шаблон
#строка 'Ах' - это замена
#в строке 4 последовательность 'Ох' будет заменена
#на 'Ах' с шагом 2
print('s3.split('*') = ',s3.split('*'))#разбиение по разделителю *
print('s1.upper = ',s1.upper())#перевод символов в верхний регистр
print('s1.lower = ',s1.lower())#перевод символов в нижний регистр
Ln: 20 Col: 40

```

```
Python 3.7.1 Shell
File Edit Shell Debug Options Window Help
Python 3.7.1 (v3.7.1:260ec2c36a, Oct 20 2018, 14:05:16) [MSC v.191] on win32
Type "help", "copyright", "credits" or "license()" for more infor
>>>
===== RESTART: К:\Лабораторные Python\example_string.py =
s1 = Пропаганда
s2 = Сенсация
s3 = Сенсация*Сенсация*Сенсация*Сенсация
s4 = ОхОхОхАх
s1+s2 = ПропагандаСенсация
s1*3 = ПропагандаПропагандаПропаганда
s1[2] = о
s1[2,4] = оп
s3.count = 4
s1.find(а) = 4
s1.index(п) = 3
s1.rindex(д) = 9
s4.replace(Ох,Ах,2) = АхАхОхАх
s3.split(*) = ['Сенсация', 'Сенсация', 'Сенсация', 'Сенсация']
s1.upper = ПРОПАГАНДА
s1.lower = пропаганда
```

Пример программы:

```
s=input('Введите строку \n')
flag=1
string=''
for i in range(len(s)):
    if s[i]!=' ':
        string+=s[i]
print(string)
for i in range(len(s)//2):
    if string[i]!=string[-i-1]:
        flag=0
        break
if flag: print('Палиндром')
else: print('не палиндром')
```

```
Введите строку
а роза упала на лапу азора
арозаупаланалапуазора
Палиндром
```

ЗАДАНИЯ

Задача 1

Задайте последовательность действий, которая при вводе любой строки заменяет в ней первый и последний символ на восклицательные знаки.

Задача 2

Вводится слово. Определите его длину, первую и последнюю букву.

Задача 3

Пользователь вводит число (k) – максимально возможную длину строки. Затем вводится произвольная строка, и если ее длина превышает k, то «лишние» символы с конца строки копируются и выводятся отдельно.

Задача 4

Вводится строка – два слова через пробел (например, фамилия и имя человека). Выделите и выведите отдельно второе слово.

Задача 5

Вводится строка слов, разделенных пробелами. Найдите самое длинное слово и выведите его на экран. Случай, когда самых длинных слов может быть несколько, не обрабатывать.

Задача 6

Посчитайте количество строчных (маленьких) и прописных (больших) букв в введенной строке. Учитывать только английские буквы.

Лабораторная работа 5. Операции над списками в Python

Примеры создания списков:

```
l=[5,75,-4,7,-51] # список целых чисел
l=[1.13,5.34,12.63,4.6,34.0,12.8] # список из вещественных чисел
l=["Оля", "Владимир", "Михаил", "Дарья"] # список из строк
l=["Москва", "Иванов", 12, 124] # смешанный список
l=[[0, 0, 0], [1, 0, 1], [1, 1, 0]] # список, состоящий из списков
l=['s', 'p', ['isok'], 2] # список из значений и списка
```

Получение списка из строки:

```
stroka = "Здравствуй, Дедушка Мороз" #stroka - строка
lst=stroka.split(",") #lst - список
print('stroka = ',stroka)
print('lst=stroka.split(","):',lst)
```

Результат:

```
===== RESTART: C:/Users/maxim/Desktop/ex_list_
stroka = Здравствуй, Дедушка Мороз
lst=stroka.split(","): ['Здравствуй', ' Дедушка Мороз']
```

Примеры использования генераторов списков:

```
Создание списка из строки.
l = list (строка):
['c', 't', 'p', 'o', 'k', 'a']

Создание списка при помощи функции Split().
stroka=" Hello, friend "
lst=stroka.split(","):
['Hello', ' friend']

Генераторы списков.
Первый способ.
l = [1]*10:
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

Второй способ. Пример 1.
l = [i for i in range(10)]:
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Второй способ. Пример 2.
c=[c*3 for c in "list"]:
['lll', 'iii', 'sss', 'ttt']
```

```
Заполнить список квадратами чисел от 0 до 9, используя генератор списка.
l = [i*i for i in range(10)]:
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
Заполнить список числами, где каждое последующее число больше на 2.
l = [(i+1)+i for i in range(10)]:
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

```
from random import *
l = [randint(10,80) for i in range(10)]
print('10 чисел, сгенерированных случайным образом в диапазоне (10,80).')
print('l = [randint(10,80) for x in range(10)]:')
print(l)
print()

l = [random() for i in range(10)]
print('10 чисел сгенерированных в диапазоне от 0 до 1.')
print('l = [random() for i in range(10)]:')
for i in range(len(l)):
    print ('{:.2f}'.format(l[i]), end = " ")
```



```
10 чисел, сгенерированных случайным образом в диапазоне (10, 80).  
l = [randint(10,80) for x in range(10)]:  
[70, 33, 79, 61, 34, 27, 11, 55, 52, 31]
```

```
10 чисел сгенерированных в диапазоне от 0 до 1.  
l = [random() for i in range(10)]:  
0.66 0.97 0.87 0.57 0.54 0.83 0.57 0.65 0.04 0.07
```

Ввод списков:

```
print('Ввод списка. Пример 1:')  
x=[]  
for i in range(4):  
    x.append(int(input()))  
print(x)  
  
x=[]  
print('Ввод списка. Пример 2:')  
x = [ int(input()) for i in range(4) ]  
print(x)
```

```
Ввод списка. Пример 1:  
45  
4  
85  
2  
[45, 4, 85, 2]  
Ввод списка. Пример 2:  
4  
5  
7  
8  
[4, 5, 7, 8]
```

Методы для работы со списками:

Метод	Что делает
list.append(x)	Добавляет элемент в конец списка
list.extend(L)	Расширяет список list, добавляя в конец все элементы списка L
list.insert(i, x)	Вставляет перед i-ым элементом значение x
list.remove(x)	Удаляет первый элемент в списке, имеющий значение x. ValueError, если такого элемента не существует
list.pop([i])	Удаляет i-ый элемент и возвращает его. Если индекс не указан, удаляется последний элемент
list.index(x, [start [, end]])	Возвращает положение первого элемента со значением x (при этом поиск ведется от start до end)

<code>list.count(x)</code>	Возвращает количество элементов со значением x
<code>list.reverse()</code>	Разворачивает список
<code>list.copy()</code>	Поверхностная копия списка
<code>list.clear()</code>	Очищает список

Примеры применения:

```

a=[0,2,2,2,4] #список a
b=[5,6,7,2,9] #список b
print('Исходный список a:',a)
print('Исходный список b:',b)
x=99
y=5

a.append(x)
print('a.append(x):',a)

a.extend(b)
print('a.extend(b):',a)

a.insert(3,x)
print('a.insert(3,x):',a)

a.remove(x)
print('a.remove(x):',a)

print('a.pop(5):',a.pop(5))
print(a)

print('a.index(y,0,len(a)):',a.index(y,0,len(a)))

print('a.count(2):',a.count(2))

a.reverse()
print('a.reverse():',a)

```

```

Исходный список a: [0, 2, 2, 2, 4]
Исходный список b: [5, 6, 7, 2, 9]
a.append(x): [0, 2, 2, 2, 4, 99]
a.extend(b): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.insert(3,x): [0, 2, 2, 99, 2, 4, 99, 5, 6, 7, 2, 9]
a.remove(x): [0, 2, 2, 2, 4, 99, 5, 6, 7, 2, 9]
a.pop(5): 99
[0, 2, 2, 2, 4, 5, 6, 7, 2, 9]
a.index(y,0,len(a)): 5
a.count(2): 4
a.reverse(): [9, 2, 7, 6, 5, 4, 2, 2, 2, 0]

```

ЗАДАНИЯ

Задача 1

В среде Python введите любой список и примените к нему операции: обращения к элементу по его индексу, замены элемента, добавления и удаления элементов, дублирования списка.

Задача 2

Дан список, состоящий из N целочисленных элементов. Требуется найти минимальный элемент. Вывести индекс минимального элемента на экран.

Задача 3

Дан список целых чисел. Перепишите все положительные элементы во второй массив, а остальные – в третий.

Задача 4

В списке D длиной n вычислите сумму элементов с нечетными индексами. Выведите на экран список D и полученную сумму.

Задача 5

Дан список из 8 элементов. Замените все элементы списка меньше 15 их удвоенными значениями. Выведите на экран преобразованный массив.

Задача 6

Даны два списка чисел, которые могут содержать до 10000 чисел каждый. Выведите все числа, которые входят как в первый, так и во второй список в порядке возрастания.

Лабораторная работа 6. Функции и процедуры

Пример работы процедуры в Python:

```
def printChar(s):
    print(s)
sim = input('введите символ: |')
printChar(sim) # первый вызов, вывод введенного символа
printChar('*') # второй вызов, вывод *
```

```
>>>
введите символ: 41
41
*
```

```
Python 3.4.1: ex_procedure3.py - C:/Documents and Settings/St
File Edit Format Run Options Windows Help
x = 10 # глобальная переменная
def pr(a): # процедура с параметром
    print(a) # 4
pr(x) # передача параметра глобальной переменной (3)

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:3
tel)] on win32
Type "copyright", "credits" or "license()" for more
>>> ===== RESTART =====
>>>
10
>>> |
```

Пример изменения значения глобальной переменной:

```
Python 3.4.1: ex_procedure4.py - C:\Docume
File Edit Format Run Options Windows Help
x=3 # глобальная переменная
print('Начальное значение: ',x)
def pr(): # процедура без параметров
    global x
    x=pow(x,10)
    print('Изменённое значение: ',x)
pr()

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May
tel)] on win32
Type "copyright", "credits" or "licens
>>> ===== I
>>>
Начальное значение: 3
Изменённое значение: 59049
>>>
```

Оператор return для возвращения значения в функции:

```

def sumD(n): # определение функции с параметром
    summa = 0
    while n != 0:
        summa += n % 10
        n = n // 10
    return summa # возврат значения функции
# основная программа
print (sumD(int(input()))) # вызов функции с параметром

```

```

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
Python 3.4.1 (v3.4.1:c0e311e010fc, May 18 2014, 10:38:22
tel)] on win32
Type "copyright", "credits" or "license()" for more info
>>> ===== RESTART =====
>>>
123456789
45
>>>

```

Пример процедуры, меняющей местами первый и последний элементы списка (массива):

```

def zam(X):
    tmp=X[0]
    X[0]=X[len(X)-1]
    X[len(X)-1]=tmp
A=[]
m=int(input('Введите длину массива:'))
for i in range(m):
    print('Введите ',i,'элемент массива')
    A.append(int(input()))
print(A)
zam(A)
print(A)

```

```

Введите длину массива:5
Введите 0 элемент массива
0
Введите 1 элемент массива
1
Введите 2 элемент массива
2
Введите 3 элемент массива
3
Введите 4 элемент массива
4
[0, 1, 2, 3, 4]
[4, 1, 2, 3, 0]

```

ЗАДАНИЯ

Задача 1

Напишите функцию, которая вычисляет количество цифр числа.

Задача 2

Напишите функцию, которая вычисляет факториал натурального числа N.

Лабораторная работа 7. Работа с двумерными массивами

Создание двумерного массива (матрицы) из одномерных списков:

```
n=3
m=3
A=[0]*n
for i in range(n):
    A[i]=[0]*m
print('A:',A)
```

```
A: [[0, 0, 0], [0, 0, 0], [0, 0, 0]]
>>> |
```

```
n=3
m=4
A = []
for i in range(n):
    A.append([0]*m)
print(A)
```

```
A: [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
>>> |
```

Пример ввода и вывода массива:

```
n=3
A = []
#Ввод массива
for i in range(n):
    B = []
    for i in range(n):
        B.append(int(input()))
    A.append(B)
```

```

#Вывод массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end=' ')
    print()

```

```

1
2
3
4
5
6
7
8
9
1 2 3
4 5 6
7 8 9

```

Пример обработки двумерного массива:

```

A=[[1, 2, 3,4],[ 5, 6,7,8]]
#Вывод при помощи цикла for и метода join
print('Массив A:')
for i in A:
    print(' '.join(list(map(str, i))))
#Пример 1. Подсчёт суммы всех элементов
s = 0
for i in range(len(A)):
    for j in range(len(A[i])):
        s += A[i][j]
print('Пример 1. Сумма элементов:', s)
#Пример 2. Подсчёт суммы всех элементов
s = 0
for row in A:
    for elem in row:
        s += elem
print('Пример 2. Сумма элементов:', s)

```

Массив A:

```

1 2 3 4
5 6 7 8

```

Пример 1. Сумма элементов: 36

Пример 2. Сумма элементов: 36

Замена элементов выше и ниже главной диагонали квадратной матрицы:

```
n=3
A=[]
#заполняем массив 9-ми
for i in range(n):
    A.append([9]*n)
#Вывод исходного массива
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    for j in range(n):
        if i < j:
            A[i][j] = 1
        elif i > j:
            A[i][j] = 2
        else:
            A[i][j] = 0
#Вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```

```
9 9 9
9 9 9
9 9 9

0 1 1
2 0 1
2 2 0
>>> |
```

Другой вариант:

```
#заменяем элементы главной диагонали, выше и ниже неё
for i in range(n):
    for j in range(0, i):
        A[i][j] = 2
    A[i][i] = 0
    for j in range(i + 1, n):
        A[i][j] = 1
#Вывод изменённого массива
print()
for i in range(n):
    for j in range(n):
        print(A[i][j], end = ' ')
    print()
```


ЗАДАНИЯ

Задача 1

Дан двумерный массив размером $m \times n$. Сформируйте новый массив, заменив положительные элементы единицами, а отрицательные нулями. Выведите оба массива.

Задача 2

Дана целая квадратная матрица n -го порядка. Определите, является ли она магическим квадратом, т.е. такой матрицей, в которой суммы элементов во всех строках и столбцах одинаковы.

Задача 3

Требуется упорядочить по возрастанию элементы каждой строки матрицы размером $n \times m$.

Лабораторная работа 8. Создание классов и объектов в Python. Конструктор класса

Класс содержит имя студента `full_name`, номер группы `group_number` и список полученных оценок `progress`. В программе вводится список студентов. Далее список сортируется по имени, потом выводятся студенты, имеющие неудовлетворительные оценки.

```
class Student:
    def __init__(self, full_name="", group_number=0, progress=[]): # конструктор
        self.full_name = full_name # имя

        self.group_number = group_number # номер группы
        self.progress = progress # оценки
    def __str__(self): # печатаемое представление экземпляра класса
        txt = 'Студент: ' + self.full_name + ' Группа: ' + self.group_number
        txt += ' Оценки:'
        for x in self.progress:
            txt += ' ' + str(x) # добавляем список оценок
        return txt

#-----
def SortParam(st): # функция определяющая атрибут для сортировки
    return st.full_name
#-----
```

```

st_size = 5 # количество студентов

students = [] # создание пустого списка
for i in range(st_size): # цикл для ввода st_size студентов
    print("Введите полное имя студента: ")
    full_name = input() # ввод фамилии
    print("Введите номер группы: ")
    group_number = input() # ввод группы
    n=5
    print('Введите ',n,' оценок в столбик: ') # у каждого студента n оценок
    progress = []
    for i in range(n):
        score = int(input()) # ввод оценок
        progress.append(score) # добавление оценок
    # создание экземпляра класса Student:
    st = Student(full_name, group_number, progress)
    students.append(st) # добавление экземпляра в список

print("Students list:")
for st in students: # вывод полного списка студентов
    print(st)

# сортировка по фамилии, ключ сортировки определяется функцией SortParam:
students = sorted(students, key=SortParam)

print("Sorted students:")
for st in students: # вывод отсортированного списка
    print(st)

print("bad students:")
n=0 # счетчик количества неуспевающих
for st in students: # вывод неуспевающих
    for val in st.progress:
        if val<3 : # есть плохая оценка
            print(st) # выводим студента с плохой оцекой
            n += 1
            break
if n == 0:
    print("no matches were found.")

```

Лабораторная работа 9. Наследование. Множественное наследование

Класс ForeignPassport является производным от класса Passport. Метод PrintInfo существует в обоих классах. PassportList представляет собой список, содержащий объекты обоих классов.

```
class Passport():
    def __init__(self, first_name, last_name, country, date_of_birth,
numb_of_pasport):
        self.first_name = first_name
        self.last_name = last_name
        self.date_of_birth = date_of_birth
        self.country = country
        self.numb_of_pasport = numb_of_pasport

    def PrintInfo(self):
        print("\nFullname: ",self.first_name, " ",self.last_name)
        print("Date of birth: ",self.date_of_birth)
        print("County: ",self.country)
        print("Passport number: ",self.numb_of_pasport)

class ForeignPassport(Passport):
    def __init__(self, first_name, last_name, country, date_of_birth,
numb_of_pasport,visa):
        super().__init__(first_name, last_name, country, date_of_birth,
numb_of_pasport)
        self.visa = visa

    def PrintInfo(self):
        super().PrintInfo()
        print("Visa: ",self.visa)

PassportList=[]
request = ForeignPassport('Ivan', 'Ivanov', 'Russia', '12.03.1967', '123456789',
'USA')
PassportList.append(request)

request = Passport('Иван', 'Иванов', 'Россия', '12.03.1967', '45001432')
PassportList.append(request)

request = ForeignPassport('Peter', 'Smit', 'USA', '01.03.1990', '21435688',
'Germany')
PassportList.append(request)

for emp in PassportList:
    emp.PrintInfo()
```

Лабораторная работа 10. Полиморфизм в Python

Классы Printer, Scanner и Xerox являются производными от класса Equipment. Метод str() перегружен только в классе Printer, для остальных используется метод из базового класса. Метод action() перегружен для всех производных классов. Вызов этих методов для каждого элемента списка демонстрирует их полиморфное поведение.

```
class Equipment:
    def __init__(self, name, make, year):
        self.name = name # производитель
        self.make = make # модель
        self.year = year # год выпуска
    def action(self):
        return 'Не определено'
    def __str__(self):
        return f'{self.name} {self.make} {self.year}'
#-----
class Printer(Equipment):
    def __init__(self, series, name, make, year):
        super().__init__(name, make, year)
        self.series = series # серия
    def __str__(self):
        return f'{self.name} {self.series} {self.make} {self.year}'
    def action(self):
        return 'Печатает'
#-----
class Scanner(Equipment):
    def __init__(self, name, make, year):
        super().__init__(name, make, year)
    def action(self):
        return 'Сканирует'
#-----
class Xerox(Equipment):
    def __init__(self, name, make, year):
        super().__init__(name, make, year)
    def action(self):
        return 'Копирует'
#-----

sklad = []
# создаем объект сканер и добавляем
scanner = Scanner('Mustek', 'BearPow 1200CU', 2010)
sklad.append(scanner)
# создаем объект ксерокс и добавляем
xerox = Xerox('Xerox', 'Phaser 3120', 2019)
sklad.append(xerox)
# создаем объект принтер и добавляем
printer = Printer("1200", 'hp', 'Laser Jet', 2018)
sklad.append(printer)
```

```

# выводим склад
print("На складе имеются:")
for x in sklad:
    print(x,end=' ')
    print(x.action())
# забираем со склада все принтеры
for x in sklad:
    if isinstance(x,Printer):
        sklad.remove(x)
# выводим склад
print("\nНа складе осталось:")
for x in sklad:
    print(x,end=' ')
    print(x.action())

```

Лабораторная работа 11. Примеры композиции классов

В классе Battle реализована композиция: он включает два объекта типа Soldier.

```

from random import randint
class Soldier: # класс описывающий одного солдата
    def __init__(self,name='Noname',health = 100): # конструктор
        self.name = name # задаем имя воина
        self.health = health # задаем начальное здоровье
    def set_name(self, name):
        self.name = name # есть возможность поменять имя
    def make_kick(self, enemy): # метод моделирующий атаку на солдата епему
        enemy.health -= 20 # при атаке здоровье врага уменьшаем на 20
        if enemy.health<0 :
            enemy.health = 0
        self.health += 10 # а собственное здоровье увеличиваем на 10
        print(self.name, "бьет", enemy.name) # выводим кто кого бьет
        print('%s = %d' % (enemy.name, enemy.health)) # выводим состояние врага
#-----
class Battle:
    def __init__(self,u1,u2): # конструктор
        # композиция: класс включает двух солдат u1 и u2
        self.u1 = u1
        self.u2 = u2
        self.result = "Сражения не было" # строка для хранения состояния сражения
    def battle(self): # метод моделирующий сражение
        while self.u1.health > 0 and self.u2.health > 0:
            n = randint(1, 2) # определяем, кто атакует
            if n == 1:
                self.u1.make_kick(self.u2) # если атакует первый
            else:
                self.u2.make_kick(self.u1) # если атакует второй
        if self.u1.health > self.u2.health:# определяем, кто победил

```

```

        self.result = self.u1.name + " ПОБЕДИЛ"
    elif self.u2.health > self.u1.health:
        self.result = self.u2.name + " ПОБЕДИЛ"
    def who_win(self): # вывод результата
        print(self.result)
#-----
first = Soldier('Mr. First',140) # создаем 1 солдата с именем Mr. First и здоровьем
140
second = Soldier() # создаем 2 солдата с параметрами по умолчанию
second.set_name('Mr. Second') # меняем имя 2 солдата
b = Battle(first,second) # создаем объект Battle
b.battle() # запускаем сражение
b.who_win() # выводим итог

```

Лабораторная работа 12. Создание и использование абстрактного метода

Пример класса с абстрактным методом.

```

import abc
class Class1(abc.ABC):
    def __init__(self, val):
        self.x = val
    @abc.abstractmethod # Абстрактный метод
    def func(self):
        raise NotImplementedError("Нельзя вызывать абстрактный метод")

class Class2(Class1): # Наследуем абстрактный метод
    def another_func(self): # Определяем другой метод
        print(-self.x)

class Class3(Class2): # Наследуем два метода
    def func(self): # Переопределяем абстрактный метод
        print(self.x)

try: # Перехватываем исключения
    c = Class1(10) # Ошибка. Метод func() не переопределен
except TypeError as msg:
    print(msg) # вывод: Can't instantiate abstract class Class1 with abstract ...

try: # Перехватываем исключения
    c = Class2(10) # Ошибка. Метод func() не переопределен
except TypeError as msg:
    print(msg) # вывод: Can't instantiate abstract class Class1 with abstract ...

c = Class3(30)
c.func() # вывод: 30
c.another_func() # вывод: -30

```

Лабораторная работа 13. Создание и использование статического метода

Класс, представляющий рациональную дробь (num – числитель, den – знаменатель). Класс содержит конструктор и перегруженные методы умножения и деления (дроби на дробь и дроби на целое число). Метод создания случайной дроби из заданного диапазона целых чисел объявлен как статический.

```
from math import gcd
from random import randint

class My_Fraction:
    def __init__(self, num, den):
        if num != 0 and den != 0:
            k = gcd(num, den) # находим НОД
            self.num = num // k # числитель
            self.den = den // k # знаменатель
        else:
            raise ValueError

    @staticmethod
    def generate(num_min, num_max, den_min, den_max):
        return My_Fraction(randint(num_min, num_max), randint(den_min, den_max))

    def __str__(self): # Метод преобразования дроби в строку
        return f'{self.num}/{self.den}'

    def __mul__(self, other): # Умножение дробей
        if isinstance(other, My_Fraction): # перегрузка умножения на дробь
            return My_Fraction(self.num * other.num, self.den * other.den)
        if isinstance(other, int): # перегрузка умножения на целое число
            return My_Fraction(self.num * other, self.den)
        return self # для остальных типов возвращаем значение самого
# объекта

    def __truediv__(self, other): # Деление дробей
        if isinstance(other, My_Fraction): # перегрузка деления на дробь
            return My_Fraction(self.num * other.den, self.den * other.num)
        if isinstance(other, int): # перегрузка деления на целое число
            return My_Fraction(self.num, self.den*other)
        raise TypeError # для остальных типов вызываем исключение
#-----

# Список из 5 случайных дробей:
a = [My_Fraction.generate(1, 9, 1, 9) for i in range(5)]
for f in a:
    b = My_Fraction.generate(1, 9, 1, 9) # дробь для правого операнда
    cm = f * b
    print(f'{f} * {b} = {cm}') # пример умножения на дробь
    cd = f / b
    print(f'{f} / {b} = {cd}') # пример деления на дробь
    n=randint(1, 9) # число для правого операнда
    cm = f * n
    print(f'{f} * {n} = {cm}') # пример умножения на число
    cd = f / n
    print(f'{f} / {n} = {cd}') # пример деления на число
```


ЗАДАНИЯ

Задача 1

Создайте класс «Мебель» с полями «Марка», «Название», «Цена» и методом для вывода подробной информации о предмете. От класса «Мебель» необходимо унаследовать класс «Стол» с унаследованными полями класса «Мебель» и новыми полями «Спинка» (True/False), «Кол-во ножек» и методом для вывода подробной информации.

Задача 2

Создайте базовый класс «Транспортное средство» и производные классы «Автомобиль», «Велосипед», «Повозка». Подсчитайте время и стоимость перевозки пассажиров и грузов каждым транспортным средством.

Задача 3

Создайте базовый класс «Домашнее животное» и производные классы «Собака», «Кошка», «Попугай» и др. С помощью конструктора установите имя каждого животного и его характеристики.

Задача 4

Создайте абстрактный класс Shape для рисования плоских фигур. Необходимо построить производные классы Square (квадрат, который характеризуется координатами левого верхнего угла и длиной стороны), Circle (окружность с заданными координатами центра и радиусом), Ellipse (эллипс с заданными координатами вершин описанного вокруг него прямоугольника), позволяющие рисовать указанные фигуры, а также передвигать их на плоскости.

Задача 5

Создайте приложение, в котором необходимо разработать базовый класс Man. Объекты этого класса содержат справочную информацию о конкретном человеке (фамилию, инициалы, телефон, адрес, возраст). Создайте два производных от него класса: Manager и Secretary. Объекты класса Manager дополнительно включают номер отдела и количество подчиненных. Объекты класса Manager дополнительно включают номер отдела и количество подчиненных. Объекты класса Secretary дополнительно включают фамилию начальника. Данные о менеджерах и секретарях введите с клавиатуры и выведите на экран дисплея.

Задача 6

Разработайте класс Book: Автор, Название, Издательство, Год, Количество страниц. Создайте массив объектов. Выведите:

- а) список книг заданного автора;
- б) список книг, выпущенных заданным издательством;

в) список книг, выпущенных после заданного года.

Задача 7

Разработайте класс Word: Слово, Номера страниц, на которых слово встречается (от 1 до 10), Число страниц. Создайте массив объектов. Выведите:

- а) слова, которые встречаются более чем на N страницах;
- б) слова в алфавитном порядке;
- в) для заданного слова номера страниц, на которых оно встречается.

Задача 8

Разработайте класс Равнобочная трапеция, члены класса – координаты 4-х точек. Предусмотрите в классе конструктор и методы: проверка, является ли фигура равнобочной трапецией; вычисление и вывод сведений о фигуре: длины сторон, периметр, площадь.

Использованные источники

1. Лабораторные работы «Язык программирования Python». – Режим доступа: http://agpu.net/fakult/ipimif/fpiit/kafinf/MethodicheskoyeObespecheniye/KPP_SChernyshov.pdf
2. Задорожный С.С., Фадеев Е.П. Объектно-ориентированное программирование на языке Python. – М.: Физический факультет МГУ им. М. В. Ломоносова, 2022. – 40 с.

Список литературы

1. Объектно-ориентированное программирование. В 3-х частях. Ч.1: учебное пособие / П.П. Степанов, А.А. Кабанов, В.А. Никонов, Т.С. Павлюченко. – Омск: Омский государственный технический университет, 2021. – 112 с. – Режим доступа: <https://www.iprbookshop.ru/124850.html> (ЭБС «IPRbooks»).
2. Букунов, С.В. Объектно-ориентированное программирование на языке Python: учебное пособие / С.В. Букунов, О.В. Букунова. – Санкт-Петербург: Санкт-Петербургский государственный архитектурно-строительный университет, ЭБС АСВ, 2020. – 119 с. – Режим доступа: <https://www.iprbookshop.ru/117194.html> (ЭБС «IPRbooks»).
3. Зыков, С.В. Введение в теорию программирования. Объектно-ориентированный подход: учебное пособие / С.В. Зыков. – 3-е изд. – Москва: Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. – 187 с. – Режим доступа: <https://www.iprbookshop.ru/102007.html> (ЭБС «IPRbooks»).
4. Щерба, А.В. Программирование на Python: первые шаги / А.В. Щерба. – Москва: Лаборатория знаний, 2022. – 251 с. – Режим доступа: <https://www.iprbookshop.ru/120878.html> (ЭБС «IPRbooks»).