**Уважаемые студенты!**

1. **Ход работы**

Вам необходимо выполнить указанные ниже задания, работая в приложении«JupyterNotebook», сохраняя скриншоты кодов в программе«Microsoft Word».

1. **Методические указания**

JupyterNotebook - это веб-приложение с открытым исходным кодом, которое можно использовать для создания и обмена документами, содержащими живой код, уравнения, визуализации и текст. Название Jupyter происходит от основных поддерживаемых языков программирования, которые он поддерживает: Julia, Python и R.

Выполненные задания по програмиированию в среде JupyterNotebookнеобходимо оформить в виде отчета – отдельного файла cназванием:

***«И\_ №группы\_ Фамилия\_Python\_П№11-12»***

Сохраняем выполненные задания с допустимыми расширениями для файлов: *.docx, .dox.*

Отчёт состоит из элементов:

1. Титульный лист

2. Содержание

3. Скриншоты кодов

4. Список использованных источников

**Задание 7**

**Тема 2. Условныйоператор. Типы данных.**

**1. Условный оператор**

**1.1. Выбор из двух**

*1.1.1.Условный оператор if-else*

Мы познакомились с базовыми строительными блоками программ, научились писать программы, обеспечивающие ввод, обработку и вывод данных. Более того, умеем работать со строками и числами, как мы делаем это в математике. Теперь научимся управлять ходом выполнения программы.

Программы должны уметь выполнять разные действия в зависимости от введенных данных. Для принятия решения программа проверяет, истинно или ложно определенное условие.

В Python существует несколько способов проверки условий, и в каждом случае возможны два исхода: истина (обозначается «True») или ложь (обозначается «False»).

Проверка условий и принятие решений по результатам этой проверки называется**ветвлением**(branching). Программа таким способом выбирает, по какой из возможных ветвей ей двигаться дальше.

В Python проверка условия осуществляется при помощи ключевого слова

if

Рассмотрим следующую программу:

answer = input('Какой язык программирования мы изучаем?')

if answer == 'Python'**:**

print('Верно! Мы изучаемPython =)')

print('Python - отличный язык!')

Программа просит пользователя ввести текст и проверяет результат ввода.Если введенный текст равен строке «Python», то выводит пользователю текст:

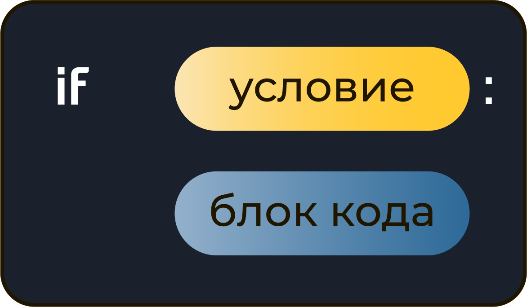
Верно! Мы изучаем Python =)

Python - отличный язык!

**Двоеточие (:)**в конце строки с инструкциейifсообщает интерпретатору Python, что дальше находится**блок команд**. В блок команд входят все строки с отступом под строкой с инструкциейif, вплоть до следующей строки без отступа.

Если условие истинно, выполняется весь расположенный ниже блок. В предыдущем примере блок инструкцийсоставляет третья и четвёртая строкипрограммы.

**Блоком кода**называют объединённые друг с другом строки, они связаны с определённой частью программы (например, с инструкциейif). В Python блоки кода формируются при помощи**отступов**.



Предыдущая программа выводит текст в случае, если условие истинно. Но если условие ложно, то программаничего не выводит. Для того, чтобы обеспечить возможность выполнять что-либо, если условие оказалось ложным, мы используем ключевое словоelse.

answer = input('Какой язык программирования мы изучаем?')

**if**answer == 'Python'**:**

print('Верно! Мы изучаемPython =)')

print('Python - отличный язык!')

**else:**

print('Не совсем так!')

В новой программе мы обрабатываем сразу два случая: если условие истинно (пользователь ввел «Python»), и еслиусловие ложно (пользователь ввел что угодно, кроме «Python»).



**Отступы**

**В некоторых языках программирования отступы — дело личного вкуса, и можно вообще обходиться без них. Однако в Python они неотъемлемая часть кода. Именно отступ сообщает интерпретатору Python, где начинается и где заканчивается блок кода.**

**Отступ**— небольшое смещение строки кода вправо. В начале такой строки находятся пробелы, и поэтому она на несколько символов отстоит от левого края.

Некоторым операторам в Python (например, оператор if) именно блок кода сообщает, какие действия следует предпринять. Послеifблок кода информирует интерпретатор Python, как действовать, если условие истинно, и как — если оно ложно.

По соглашению **PEP 8**, для отступа блоков кода используются4 пробела. Если в среде WingIDE нажать на клавишу Enter послеif, она автоматически выставит4 пробела.

*1.1.2. Операторы сравнения*

Можно заметить, что в проверке условия мы использовали двойное равенство (==), вместо ожидаемого одиночного (=). Не стоит путать**оператор присваивания**(=)с**условным оператором**(==).

Оператор присваивания (=) присваивает переменным значения:

num = 1992

s = 'I love Python'

Для проверки двух элементов на равенство Python использует удвоенный знак равно (==).

Так:

ifanswer == 'Python':

ifname == 'Gvido':

iftemperature == 40:

Путаница с операторами == и = является одной из самых распространенных ошибок в программировании.

В Python существует **6основных операторов сравнения**.

| **Выражение** | **Описание** |
| --- | --- |
| if x > 7 | если x больше 7 |
| if x < 7 | если x меньше 7 |
| if x >= 7 | если x больше либо равен 7 |
| if x <= 7 | если x меньше либо равен 7 |
| if x == 7 | если x равен 7 |
| if x != 7 | если x не равен 7 |

Рассмотрим пример:

num1 = int(input())

num2 = int(input())

if num1 < num2:

print(num1, 'меньшечем', num2)

if num1 > num2:

print(num1, 'большечем', num2)

if num1 == num2: # используем двойное равенство

print(num1, 'равно', num2)

if num1 != num2:

print(num1, 'неравно', num2)

*1.1.3. Цепочки сравнений*

Операторы сравнения в Python можно объединять в цепочки (в отличие от большинства других языков программирования, где для этого нужно использовать логические связки), например,a == b == cили1 <= x <= 10. Следующий кодпроверяет, находится ли значение переменнойageв диапазоне от 3 до 6:

age = int(input())

if 3 <= age <= 6:

print('Вы ребёнок')

Код, проверяющий равенство трех переменных, может выглядеть так:

if a == b == c:

print('числаравны')

else:

print('числа не равны')

*1.1.4. Транзитивность*

**Операция равенства является**[**транзитивной**](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C), то есть:

еслиa == bиb== c, то из этого следует, чтоa == c.

Именно поэтому предыдущий код, проверяющий равенство трёх переменных,работает какполагается.

Из курса математики вам могут быть знакомы другие примеры транзитивных операций:

**Отношение порядка:**еслиa >b*a*>*b*иb >c*b*>*c*, тоa >c*a*>*c*;

**Делимость:**если*a*делится на*b*и*b*делится на*c*, то*a*делится на*c*.

Наглядно транзитивностьотношения порядка можно понять на таком примере: если сосед слева старше вас(*a*>*b*), а вы старше соседа (*b*>*c*), то сосед слева точно старше соседа справа (*a*>*c*).

**Операция неравенства (!=), в отличие от операции равенства (==), является**[**нетранзитивной**](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D1%82%D1%80%D0%B0%D0%BD%D0%B7%D0%B8%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D1%8C).

Из того, чтоa != bиb != c,не следует, чтоa != c.

Действительно, если вас зовут не так, как соседа слеваи не так, как соседа справа, то нет гарантии, что у обоих соседей не окажутся одинаковые имена.

Таким образом, следующий кодвовсе не проверяет тот факт, что все три переменные различны:

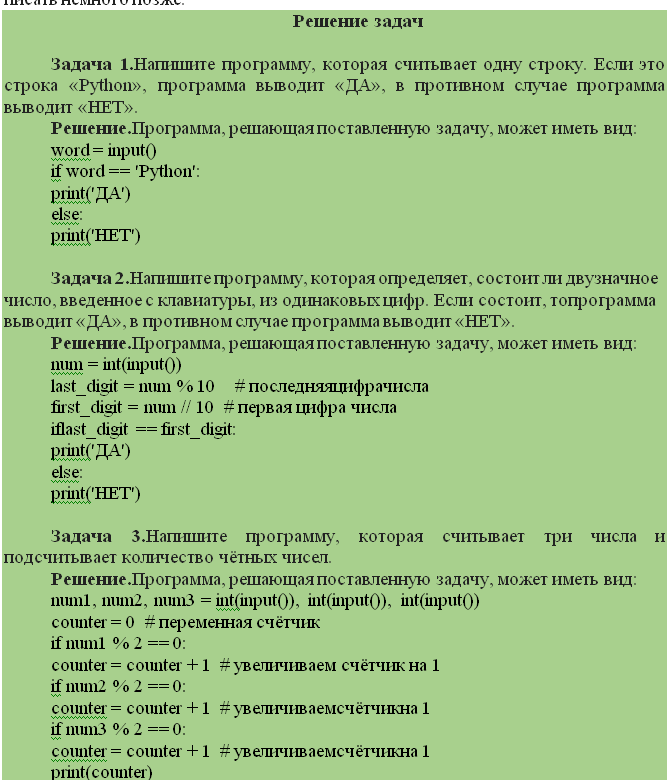
if a != b != c:

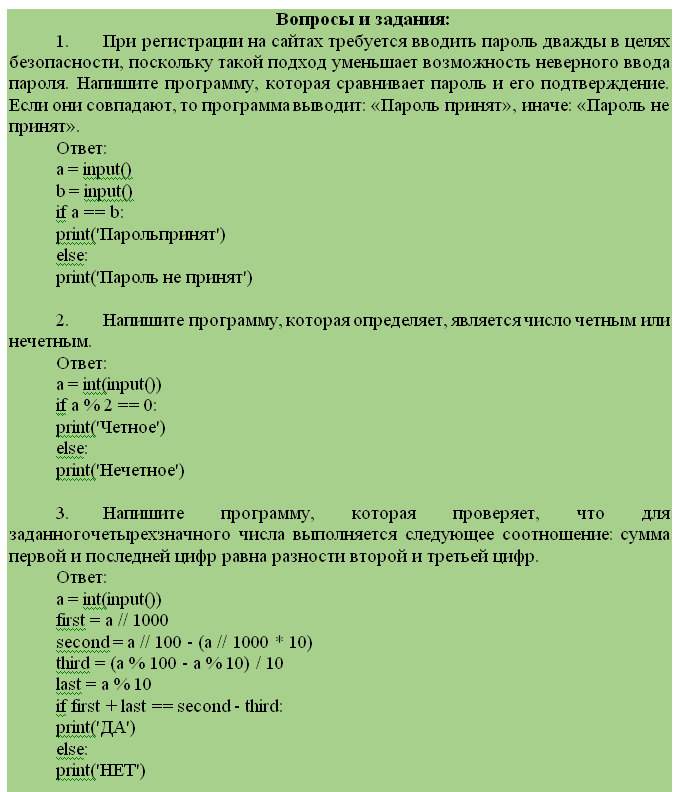
print('числа не равны')

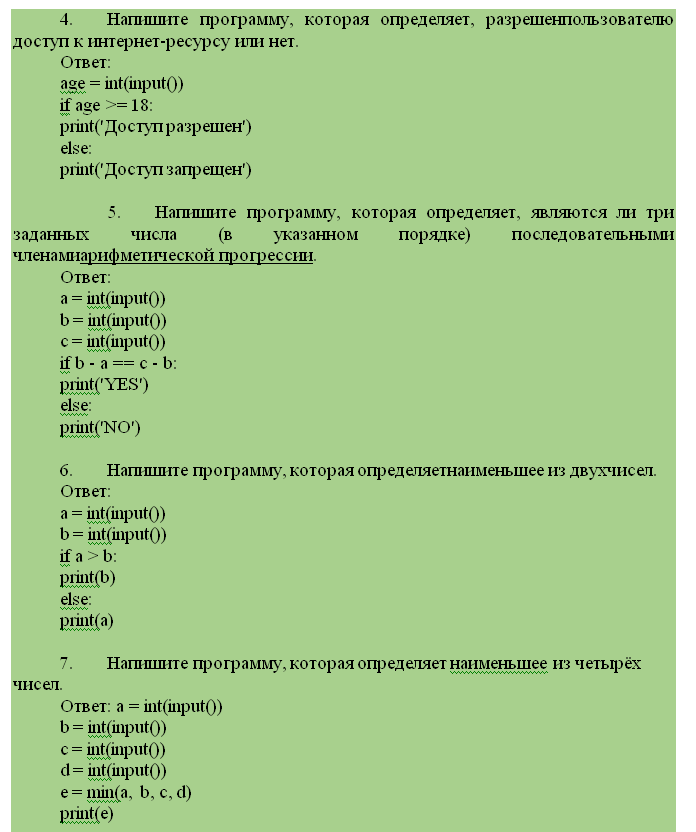
else:

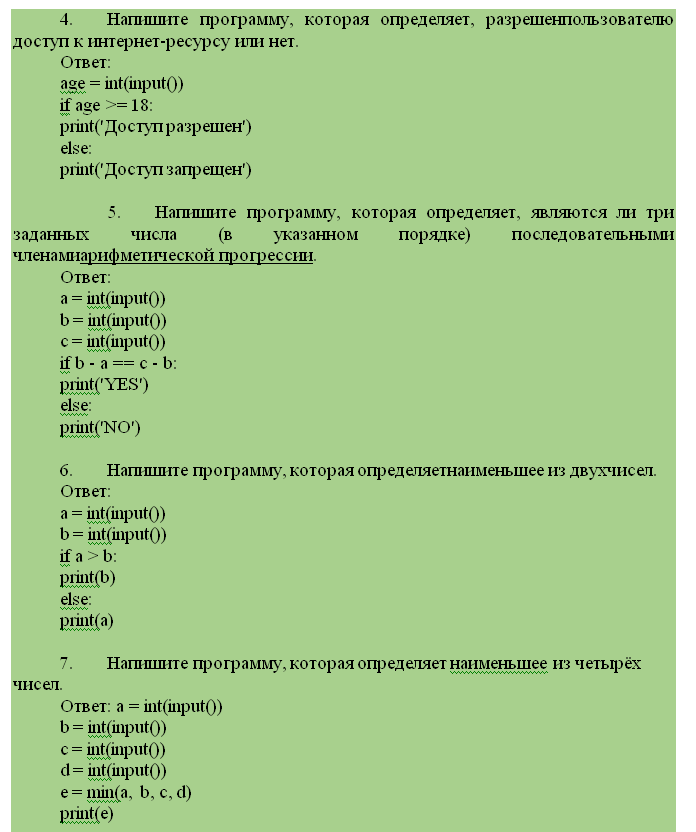
print('числа равны')

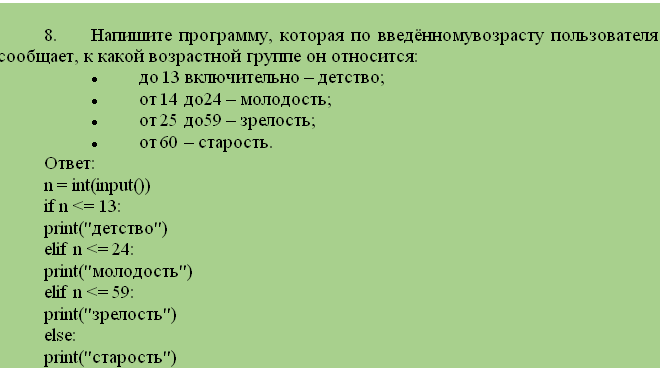
Код, проверяющий, что значения трёх переменных различны, мы научимся писать немного позже.











**1.2. Логические операции**

*1.2.1. Логические операторы*

Как быть в ситуации, когда у нас есть несколько условий? В Python есть три логических оператора, которые позволяют создавать сложные условия:

* and— логическое умножение;
* or— логическое сложение;
* not— логическое отрицание.

**Оператор and**

Предположим, мы написали программу для студентов от 19 лет, которые учатся, по крайней мерена 2 курсе. Доступ к ней тем, кто младше, надо запретить. Следующий код решает поставленную задачу:

age = int(input('Сколько вам лет?: '))

grade = int(input('На каком курсе вы учитесь?: '))

if age >= 19 and grade >= 2:

print('Доступразрешен.')

else:

print('Доступ запрещен.')

Мы объединили два условия при помощи оператораand. Оно означает, что в этом ветвлении блок кода выполняется только при выполненииобоих условийодновременно!

Операторandможет объединять произвольное количество условий:

age = int(input('Сколько вам лет?: '))

grade = int(input('На каком курсе вы учитесь?: '))

city = input('В каком городе вы живете?: ')

if age >= 19 and grade >= 2 and city == 'Москва':

print('Доступ разрешен.')

else:

print('Доступ запрещен.')

Таблица истинности для оператораand. В ней перечислены логические выражения, соединённые операторомand, показаны все возможные комбинации истинности и ложности и приведены результирующие значения выражений.

| **a** | **b** | **a and b** |
| --- | --- | --- |
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

Как показывает таблица, чтобы значение выражения с операторомandбыло истинным, должны быть истиннымиоба (все)объединенные им условия.

**Оператор or**

Операторorтакже применяется для объединения условий. Однако, в отличие отand, для выполнения блока кода достаточно выполненияхотя бы одного из условий.

city = input('В каком городе вы живете?: ')

ifcity == 'Москва' orcity == 'Санкт-Петербург' orcity == 'Екатеринбург':

print('Доступ разрешен.')

else:

print('Доступ запрещен.')

Доступ будет разрешен в случае, если хотя бы одно из условий выполнится.

Таблица истинности для оператораor. В ней перечислены логические выражения, соединённые операторомor, показаны все возможные комбинации истинности и ложности и приведены результирующие значения выражений.

| **a** | **b** | **a or b** |
| --- | --- | --- |
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | True |

Для того, чтобы выражениеorбыло истинным, требуется, чтобыхотя бы одноусловие оператораorбыло истинным. При этом не имеет значения, истинным или ложным является второе выражение.

Логическое выражениеX and Yистинно,еслиобазначения X и Y истинны.

Логическое выражениеX orYистинно,еслихотя бы одноиз значенийX и Y истинно.

Мы можем использовать оба логических оператора одновременно:

age = int(input('Сколько вам лет?: '))

grade = int(input('На каком курсе вы учитесь?: '))

city = input('В каком городе вы живете?: ')

if age >= 19 and grade >= 2 and (city == 'Казань' or city == 'Санкт-Петербург'):

print('Доступ разрешен.')

else:

print('Доступ запрещен.')

Такой код проверяет, что возраст студентов от 19 лет и учатся они по крайней мере на 2 курсе и живут в Казани или Санкт-Петербурге.

**Оператор not**

Операторnotпозволяет инвертировать(т.е. заменить на противоположный) результат логического выражения. Например, следующий код:

age = int(input('Сколько вам лет?: '))

if not (age < 19):

print('Доступразрешен.')

else:

print('Доступ запрещен.')

полностью эквивалентен коду:

age = int(input('Сколько вам лет?: '))

if age >= 19:

print('Доступразрешен.')

else:

print('Доступ запрещен.')

В первом примере мы поместили выражениеage< 19 в скобкидля того, чтобы было чётко видно, что мы применяем операторnotк значению выраженияage< 19, а не только к переменнойage.

Таблица истинности для оператора not:

| **a** | **not a** |
| --- | --- |
| False | True |
| True | False |

**Приоритеты логических операторов**

Логические операторы, подобно арифметическим операторам**(+, -, \*, /),**имеют приоритет выполнения.

Приоритет выполнения логическийх операторов следующий:

1. в первую очередь выполняется **логическое отрицаниеnot**;
2. далее выполняется **логическое умножениеand**;
3. далее выполняется **логическое сложениеor**.

Для**явного указания порядка**выполнения условных операторов**используют скобки**.

*Примечания*

**Примечание 1**.Частой ошибкой у начинающих программистов является путаница логических операторовandиor. Рассмотрим два условия:

ifx> 1 andx< 100:

ifx> 1 orx< 100:

Верным является только первое условие. В нём проверяется, что число*x*находится в диапазоне от 1 до100, другими словами,x \in (1; \, 100)*x*∈(1;100).

Второе условие проверяет, что число*x*или больше 1, или меньше 100. Однако такому условию удовлетворяет любое число!

**Примечание 2.**Другую частую ошибку видим в следующем примере кода:

ifage>= 19and<= 209:

Запуск такого кода приведет к появлению ошибки во время выполнения программы. Необходимо явно записывать условия:

if age >= 19 and age <= 20:

**Примечание 3.**Не забывайте, что в Python есть удобный способ для проверки принадлежности диапазону. Например, следующий код:

ifage>= 7 andage<= 9:

полностью эквивалентен коду:

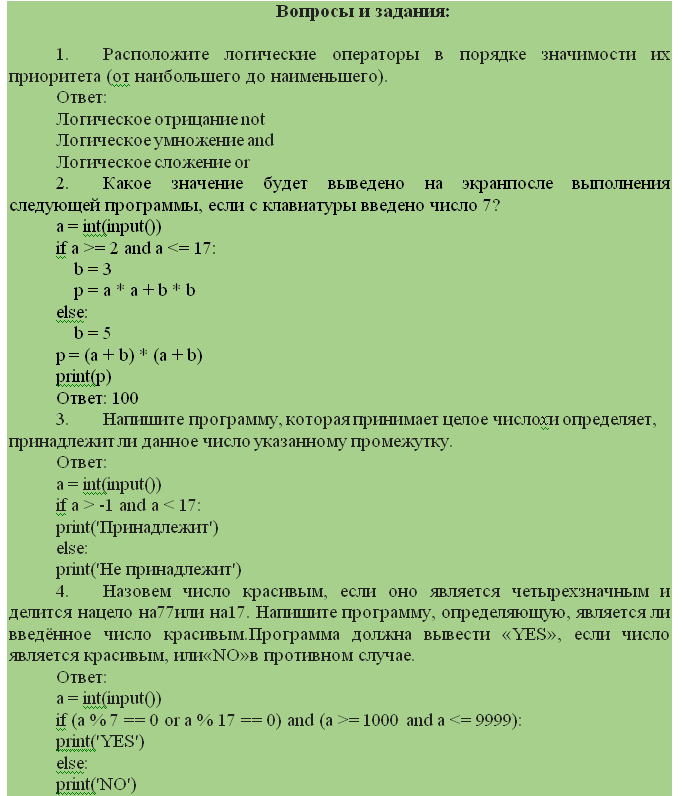
if 7 <= age<= 9:

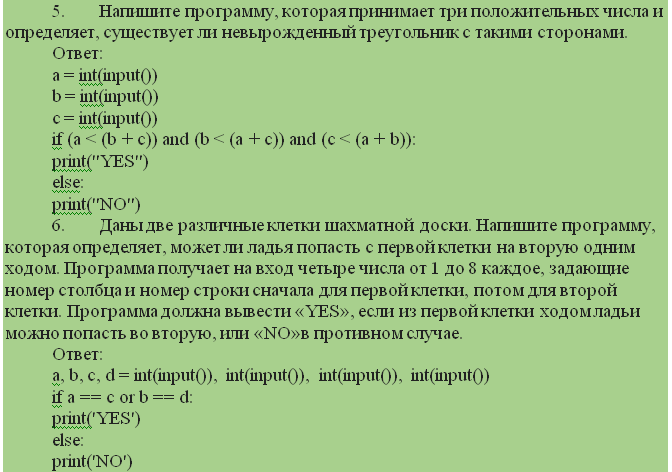
Последний код предпочтительнее.

**Примечание 4**.Оба оператора,andиor, вычисляются поукороченной схеме.

Rак это работает с операторомand. Если условие слева от оператораandявляется ложным, то условие справа от него не проверяется, так как результат выражения будет гарантированно ложным и проверка оставшегося условия —трата процессорного времени.

Аналогично работает операторor.Если условие слева от оператораorистинное, то условие справа от него не проверяется. Действительно, результат будет гарантированно истинным и проверка оставшегося условия станет тратой процессорного времени.





**1.3. Вложенные и каскадные условия**

*1.3.1. Вложенный оператор*

Внутри условного оператора можно использовать любые операторы языка Python, в том числе и условный оператор. Получаем вложенное ветвление: после одной развилки в ходе исполнения программы появляется другая развилка. При этом вложенные блоки имеют больший размер отступа (+4 пробела для каждого следующего уровня).

if условие1:

блок кода

else:

if условие2:

блок кода

else:

if условие3:

блок кода

...

В предыдущем задании мы разбирали задачу об определении координатной четверти точки. Программу можно переписать с использованием вложенного оператора:

x = int(input())

y = int(input())

if x > 0:

if y > 0:

print('Первая четверть')

else:

print('Четвертая четверть')

else:

if y > 0:

print('Вторая четверть')

else:

print('Третья четверть')

В данном случае уровень вложенностиравен двум, так что программа одинаково хорошочитается как с помощью использования логического оператораand, так и с помощью вложенного оператора.

Рассмотрим программу, которая переводит стобалльную оценку в пятибалльную. Для еёреализации нужно воспользоваться вложенным условным оператором:

grade = int(input('Введите вашу отметку по 100-балльной системе: '))

if grade >= 90:

print(5)

else:

if grade >= 80:

print(4)

else:

if grade >= 70:

print(3)

else:

if grade >= 60:

print(2)

else:

print(1)

В этом примере уровень вложенности настолько глубок, что код становится трудно понять.

Выбор из нескольких альтернатив –это обычное дело, здесь имеет смысл избегать глубокого вложения. Для этого в Python есть**каскадный условный оператор**.

Мы не могли написать 5 независимыхif-ов, поскольку в таком случае было бы напечатано сразу несколько значений пятибалльной оценки.

*1.3.2. Каскадныйусловный оператор*

**Каскадный условный оператор**– это специальный условный оператор в языке Python, который используется если требуется проверить **несколько условий**.

Синтаксис каскадного условного оператора имеет следующий вид:

if условие1:

блок кода

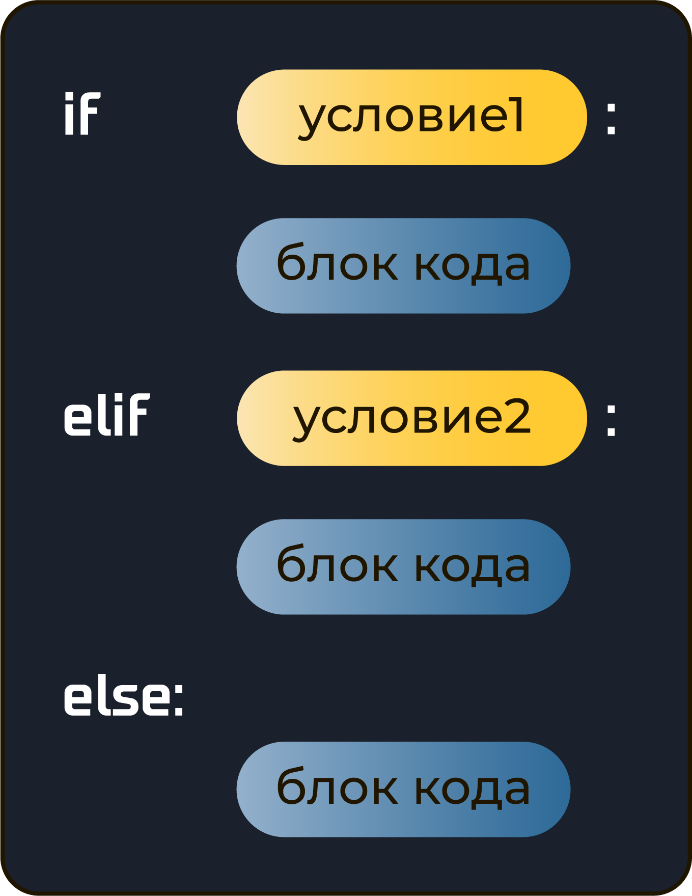
elif условие2:

блок кода

...

else:

блок кода



При исполнении каскадного условного оператора:

сначала проверяется условие 1. Если оно является истинным, то исполняется блок кода, который следует сразу после него, вплоть до выраженияelif. Остальная часть конструкции игнорируется.

Если условие 1 является ложным, то программа перескакивает непосредственно к следующему выражениюelifи проверяет условие 2. Если оно истинное, то исполняется блок кода, который следует сразу после него, вплоть до следующего выраженияelif. И остальная часть условного оператора тогда игнорируется.

Этот процесс продолжается до тех пор, пока не будет найдено условие, которое является истинным, либо пока больше не останется выраженийelif.

Если ни одно условие не является истинным, то исполняется блок кода после выраженияelse.

Приведенный ниже фрагмент кода является примером каскадного условного оператораif-elif-else. Этот фрагмент кода работает так же, как предыдущий код, использующий вложенный условный оператор.

grade = int(input('Введитеваш балл: '))

if grade >= 90:

print(5)

elif grade >= 80:

print(4)

elif grade >= 70:

print(3)

elif grade >= 60:

print(2)

else:

print(1)

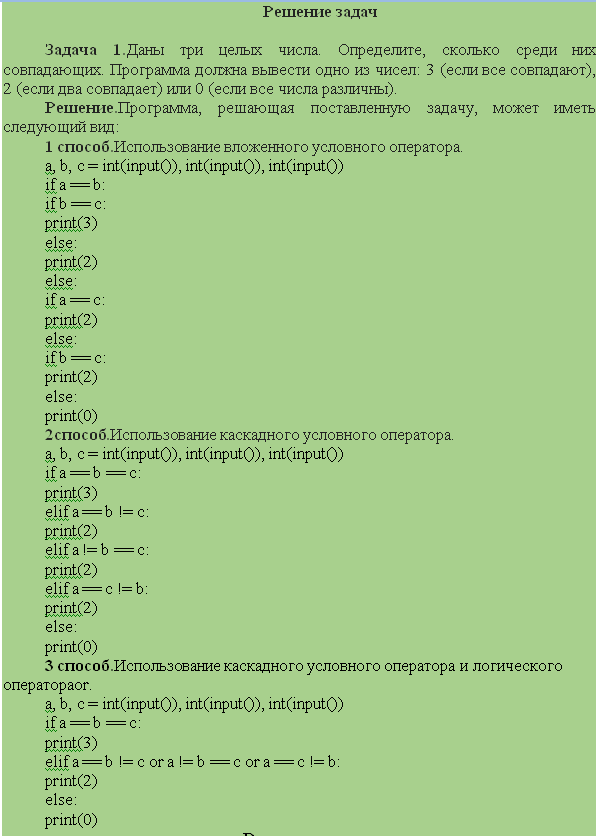
Обратите внимание на выравнивание и выделение отступом, которые применены в операторе if-elif-else: выраженияif, elifиelseвыравненыи исполняемые по условию блоки выделены отступом.

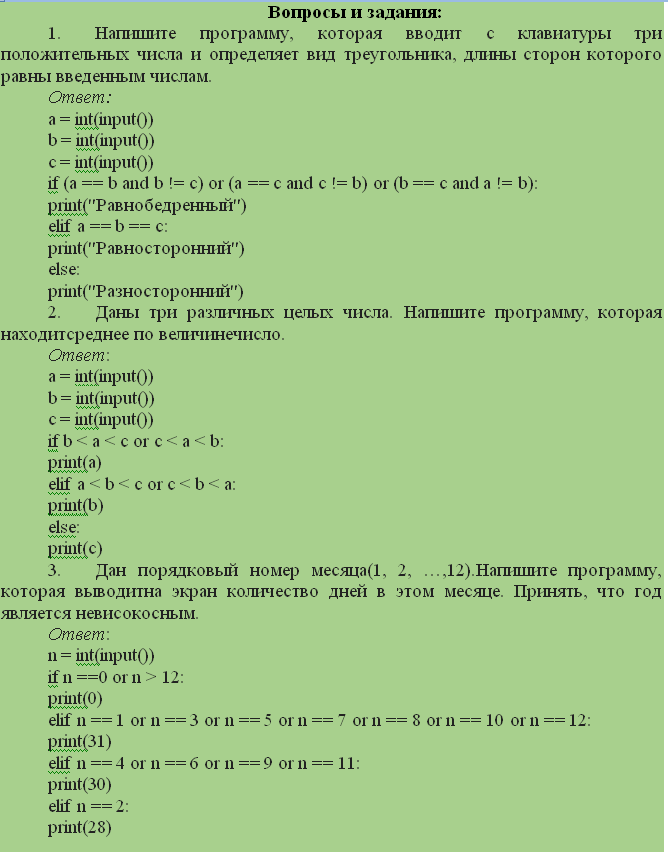
Условный оператор if-elif-elseне является обязательной, потому что ее логика может быть запрограммирована вложенными инструкциямиif-else. Однако длинная серия вложенных условных операторов if-elseимеет два характерных недостатка:

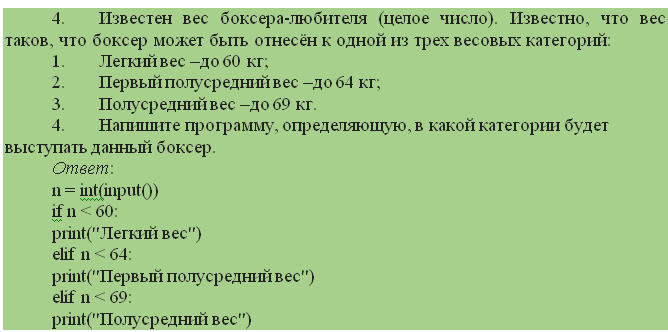
1. Программный код может стать сложным и трудным для восприятия;
2. Из-за необходимого выделения отступом продолжительная серия вложенных инструкцийif-elseможет стать слишком длинной, чтобы целиком уместиться на экране монитора без горизонтальной прокрутки.

Логика условного оператора if-elif-elseобычно прослеживается легче, чем длинная серия вложенных инструкцийif-else. И поскольку в операторе if-elif-elseвсе выражения выравнены, длина строк в данной инструкции, как правило, короче.

**Запомните.**Заключительный блокelseв оператореif-elif-elseявляется необязательным.







**2. Типы данных**

**2.1. Числовые типы данных: int, float**

*2.1.1. Целочисленный тип данных*

**Целые числа** в Python представлены типом данныхint(сокращение int происходит от английского слова integer).Для определения целого числа типаintиспользуется последовательность цифр от 0 до 9.

**Целочисленный литерал –** это явно указанное численное значение в коде программы.

Когда Python встречает целочисленный литерал, он создает объект типаint, хранящий указанное значение.

n = 17 # целочисленный литерал

m = 7 # целочисленный литерал

Целочисленный тип данныхintиспользуют не только потому, что онвстречается в реальном мире, но и потому, что онестественным образом возникает при создании большинства программ.

*2.1.2. Преобразование строки в целое число*

Для преобразования строки в целое число, мы используем команду int():

num = int(input()) # преобразование считанной строки в целое число

Для преобразования строки в целое число не обязательно использовать командуinput().

Следующий код преобразует строку12345в целое число:

n = int('12345') # преобразование строки в целое число

Если строка не является числом, то при преобразовании возникнет ошибка.

*2.1.3.Целочисленные операторы*

Язык Python предоставляет четыре основных арифметических операторовдля работы сцелыми числами**(+, −, \*, /),**а также три дополнительных **(%** для остатка, **//** для целочисленного деленияи **\*\*** для возведения в степень).

Следующая программа демонстрирует все целочисленные операторы:

a = 13

b = 7

total = a + b

diff = a - b

prod = a \* b

div1 = a / b

div2 = a // b

mod = a % b

exp = a \*\* b

print(a, '+', b, '=', total)

print(a, '-', b, '=', diff)

print(a, '\*', b, '=', prod)

print(a, '/', b, '=', div1)

print(a, '//', b, '=', div2)

print(a, '%', b, '=', mod)

print(a, '\*\*', b, '=', exp)

В результате работы такой программы будет выведено на экран:

13 + 7 = 20

13 - 7 = 6

13 \* 7 = 91

13 / 7 = 1.8571428571428572

13 // 7 = 1

13 % 7 = 6

13 \*\* 7 = 62748517

При обычном делении (/) получаетсячисло не являющееся целым. Деление на ноль приводит к ошибке.

*2.1.4. Длинная арифметика*

Отличительной особенностью языка Python является неограниченность целочисленного типа данных. Размер числа зависит только от наличия свободной памяти на компьютере.

atom = 10 \*\* 80 # количество атомов во вселенной

print('Количество атомов =', atom)

Результатом выполнения программы будет число с 81 цифрой:

Количество атомов = 100000000000000000000000000000000000000000000000000000000000000000000000000000000

*2.1.5. Символ разделитель*

Для удобного чтения чиселможно использовать символ подчеркивания:

num1 = 25\_000\_000

num2 = 25000000

print(num1)

print(num2)

Результатом выполнения такого кода будет:

25000000

25000000

*2.1.6. Числа с плавающей точкой*

Наравне с целыми числами в Python есть возможность работы с дробными (вещественными) числами.Так, например, числа  
, π–являются вещественными и целого типа intнедостаточно для ихпредставления.

**Числа с плавающей точкой в информатике**– это дробные (вещественные) числа.

Для представления чисел с плавающей точкойв Python используется тип данныхfloat.

e = 2.71828 # литерал с плавающей точкой

pi = 3.1415 # литерал с плавающей точкой

В отличие от математики, где разделителем является запятая, **в информатике используется точка** для отделения целой и дробной частей числа.

*2.1.7. Преобразование строки к числу с плавающей точкой*

Для преобразования строки к числу с плавающей точкоймы используем командуfloat():

num = float(input()) # преобразование считанной строки в число с плавающей точкой

Для преобразования строки к числу с плавающей точкой необязательно использовать командуinput().

Следующий код преобразует строку1.2345 к числу с плавающей точкой:

n = float('1.2345') # преобразование строки к числу с плавающей точкой

Если строка не является числом, то при преобразовании возникнет ошибка.

*2.1.8. Арифметические операторы*

Язык Python предоставляет четыре основных арифметических операторадля работы с числами с плавающей точкой (+,−,\*, /) и один дополнительный (\*\* для возведения в степень).

Следующая программа демонстрирует все целочисленные операторы:

a = 13.5

b = 2.0

total = a + b

diff = a - b

prod = a \* b

div = a / b

exp = a \*\* b

print(a, '+', b, '=', total)

print(a, '-', b, '=', diff)

print(a, '\*', b, '=', prod)

print(a, '/', b, '=', div)

print(a, '\*\*', b, '=', exp)

В результате работы такой программы будет выведено:

13.5 + 2.0 = 15.5

13.5 - 2.0 = 11.5

13.5 \* 2.0 = 27.0

13.5 / 2.0 = 6.75

13.5 \*\* 2.0 = 182.25

Деление на ноль приводит к ошибке.

*2.1.9. Преобразование между int и float*

Неявное преобразование.Любое целое число (типint)можно использовать там, где ожидается число с плавающей точкой (типfloat), поскольку при необходимости Python автоматически преобразует целые числа в числа с плавающей точкой.

Явное преобразование.Число с плавающей точкой нельзя неявно преобразовать в целое число. Для такого преобразования необходимо использовать явное преобразование с помощью командыint().

num1 = 17.89

num2 = -13.56

num3 = int(num1)

num4 = int(num2)

print(num3)

print(num4)

Результатом выполнения такого кода будет:

17

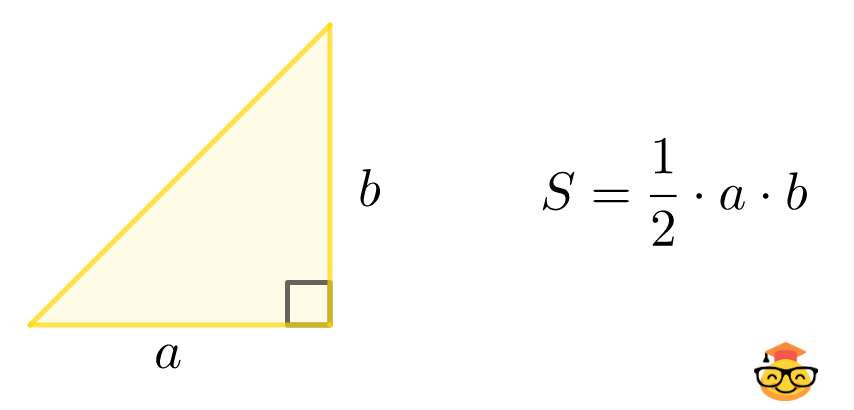
-13

Обратите внимание, что преобразованиечисел с плавающей точкой в целое производится с округлением в сторону нуля, то естьint(1.7) = 1,int(-1.7) = -1.

Не путайте операциюпреобразованияи округления. Для округления чисел с плавающей точкой используются дополнительные команды. О них расскажем позже.

**Вопросы и задания:**

1. Напишите программу, которая считывает длины двух катетов в прямоугольном треугольнике и выводит его площадь.



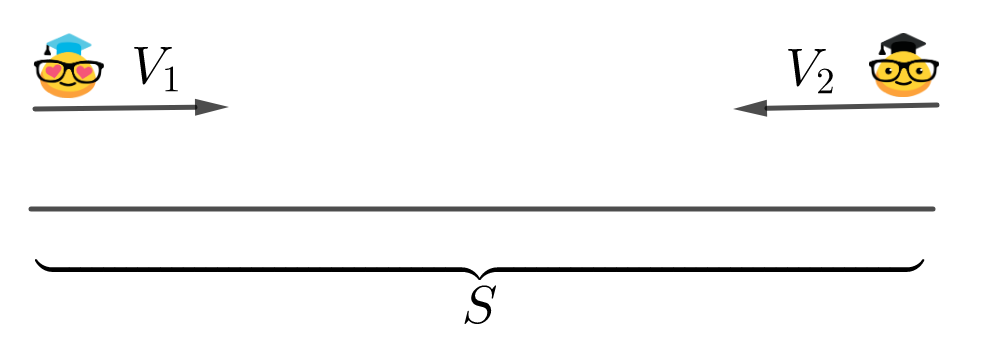
Ответ:

a = float(input())

b = float(input())

print(1 / 2 \* a \* b)

1. Два путник идут навстречу другдругу с постоянными скоростями *V*1 и *V*2 км/ч. Определите, через какое время путники встретятся, если расстояние между ними равно *S*км.



Ответ:

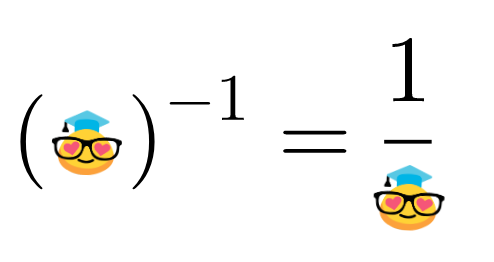
s = float(input())

v1 = float(input())

v2 = float(input())

print(s / (v1+v2))

1. Напишите программу, которая считывает с клавиатуры одно числои выводитобратное ему. Если при этом введённое с клавиатуры число – ноль, то вывести «Обратного числа не существует»(без кавычек).



Ответ:

n = float(input())

if n == 0:

print('Обратного числа не существует')

else:

print(1 / n)

1. На вход программе подается число n – количество кошачьх лет. Напишите программу, которая вычисляет возраст кошки (кота) в человеческих годах.

Ответ:

n = float(input())

res = n \* 10.5 if n <= 2 else (n-2) \* 4 + 21

print(res)

**2.2. Строковый тип данных**

Строковый тип данных, как и числовой, очень часто используется в программировании.В Python строковый тип данных имеет названиеstr(сокращение от string —струна, ряд).

Для создания строковойпеременной (литерала), мы должны заключить необходимыйтекст в кавычки. В Python можно использовать как одинарные кавычки, так и двойные:

s1 = 'Python rocks!'

s2 = "Python rocks!"

Напомним, что по умолчанию, командаinput()считывает именно строку текста:

s = input() # переменная s имеет строковый тип str

Для задания пустой строки, мы используем две кавычки одинакового типа:

**s1 = '' # пустая строка**

**s2 = ' ' # строка состоящая из одного символа пробела**

Не стоит путать пустую строку и строку состоящую из одного символа пробела. Это абсолютно разные строки.

*2.2.1. Длина строки*

**Длина строки** - это количество символов из которых состоит строка. Дл того, чтобы подсчитать длину строки используют **встроенную функциюlen()**(от слова length –длина).

Следующий программный код:

s1 = 'abcdef'

length1 = len(s1) # считаем длину строки из переменной s1

length2 = len('Python rocks!') # считаем длину строкового литерала'Pythonrocks!

print(length1)

print(length2)

Компьютер выведет длину строки и строкового литерала:

6

13

**При подсчете длины строки считаются все символы, включая пробелы.**

*2.2.2. Преобразование чисел в строку*

Для преобразования **строки кчислу**мы использовали функции**int()иfloat().**

Для обратного преобразования, то есть **из числа в строку** мы используем функцию**str()**.

Рассмотрим следующий программный код:

num1 = 1777 # целое число

num2 = 17.77 # число с плавающей точкой

s1 = str(num1) # преобразовали целое число в строку '1777'

s2 = str(num2) # преобразовали число с плавающей точкой в строку '17.77'

Иногда работать со строками намного проще, чем с числами. Даже если в условии задачисказано, что дается число, нам ничто не мешаетработать с ним как со строкой.

*2.2.3. Конкатенация строк*

**Конкатенация** или**сцепление строк –** это операция сложения строк, их можно складывать так же как и числа.

Рассмотрим следующий программный код:

s1 = 'ab' + 'bc'

s2 = 'bc' + 'ab'

s3 = s1 + s2 + '!!'

print(s1)

print(s2)

print(s3)

Результатом выполнения такого кодабудет:

abbc

bcab

abbcbcab!!

**Операция сложения строк в отличиеот операции сложения чисел не являетсякоммутативной,то есть, от перестановки мест слагаемых-строк результат меняется!**

С помощью конкатенации строк можно эмулировать вывод данных, который раньше мы делали используянеобязательные параметрыsepиend. Следующие две строки кода делают одно и тоже:

print('a', 'b', 'c', sep='\*', end='!')

print() # переход на новую строку

print('a' + '\*' + 'b' + '\*' + 'c' + '!')

Результатом выполнения такого кодабудет:

a\*b\*c!

a\*b\*c!

*2.2.4. Умножение строки на число*

В Python так же можно умножать строку на число. Такой оператор повторяет строку указанное количество раз.

Рассмотрим следующий программный код:

s = 'Hi' \* 4

print(s)

Результатом выполнения такого кодабудет:

HiHiHiHi

Оператор умножения строки на число(repetition) очень удобен на практике. Например, мы хотим распечатать строку состоящую из 75символов-. Мы можем это сделать с помощью кода:

print('-' \* 75)

Результатом выполнения такого кодабудет:

---------------------------------------------------------------------------

**Строку можно умножать на число, но нельзя умножать на строку.**

*Примечания*

**Примечание 1.**Тройные кавычки в Python используются для многострочного (multiline)текста. Например,

text = '''Python is an interpreted, high-level, general-purpose programming language.

Created by Guido van Rossum and first released in 1991, Python design philosophy emphasizes code readability with its notable use of significant whitespace.'''

**Примечание 2.**На первый взгляд может показаться странным, что можно использовать как одинарные, так и двойные кавычки, однакотакой подход позволяет очень легко добавлять в строку нужные кавычки:

s1 = 'Мы можем использовать в одиночных кавычках двойные кавычки "Война и мир"'

s2 = "Мы можем использовать в двойных кавычках одиночные кавычки 'Война и мир'"

print(s1)

print(s2)

**Вопросы и задания:**

1. Что покажет приведенный ниже фрагмент кода?

mystr = 'да'

mystr = mystr + 'нет'

mystr = mystr + 'да'

print(mystr)

Ответ: данетда

1. Что покажет приведенный ниже фрагмент кода?

str1 = '1'

str2 = str1 + '2' + str1

str3 = str2 + '3' + str2

str4 = str3 + '4' + str3

print(str4)

Ответ: 121312141213121

1. Напишите программу, которая выводит текст.

"Python is a great language!", said Fred. "I don't ever remember having this much fun before."

Ответ: print("\"Python is a great language!\"" + ", said Fred. " + "\"I don't ever remember having this much fun before.\"")

1. Напишите программу, которая считывает с клавиатуры название футбольной команды и выводит фразу:

«Футбольная команда [введённая строка] имеет длину [длина введённой строки] символов».

Ответ:

c = input()

print(f"Футбольная команда {c} имеет длину", len(c), "символ

1. Даны названия трех городов. Напишите программу, которая определяет самое короткое и самое длинное название города.

Ответ: a, b, c = str(input()), str(input()), str(input())

print(min(a, b, c, key=len ))

print(max(a, b, c, key=len ))

*2.2.5. Оператор in*

**Операторin** позволяет проверить, что одна строка находится внутри другой.

Рассмотрим следующий код:

s = input()

if 'a' ins:

print('Введенная строка содержит символ а')

else:

print('Введенная строка не содержит символ а')

Такой код проверяет,содержится ли впеременной*s*символ*a*и выводит соответствующий текст.

Мы можем использовать операторinвместе с логическим операторомnot. Например

s = input()

if '.' notins:

print('Введенная строка не содержит символа точки')

С помощью оператораinмы можем упроститьследующий код, проверяющий, что в переменнойsнаходится один из 5 символов a, e, i, o, u:

if s == 'a' or s == 'e' or s == 'i' or s == 'o' or s == 'u':

print('YES')

довида:

iflen(s) == 1 and s in 'aeiou':

print('YES')

*Примечания*

**Примечание.**Если строкаs1содержится в строкеs2, то говорят, что строкаs1является**подстрокой**для строкиs2. Другими словами, операторinопределяет является ли однастрока подстрокой другой.

**Вопросы и задания**:

1. Какие значения может принимать строковая переменнаяs, чтобы в результате выполнения кодабыло выведено слово «YES»?

if s in 'abc123abc':

print('YES')

else:

print('NO')

Ответ:

s = '3ab'

s = 'a'

s = '23'

s = '1'

1. Напишите программу, которая считывает одну строку, после чего выводит «YES», если в введенной строке есть подстрока «синий» и «NO» в противном случае.

Ответ: print('YES' if 'синий' in input() else 'NO')

1. Будем считать email адрес корректным, если в нем есть символ собачки (@) и точки. Напишите программу проверяющую корректность email адреса.

Ответ:

email = input()

print('YES' if all(\_ in email for \_ in ('@', '.')) else 'NO')

***Результат по теме 1***

***Уважаемые студенты!***

*Пожалуйста, сохраните выполненное задание в файле:*

*«И\_ №группы\_ Фамилия\_ Python\_П№11-12»*

Задания необходимо оформить в виде отчета, который состоит из элементов:

1. Титульный лист

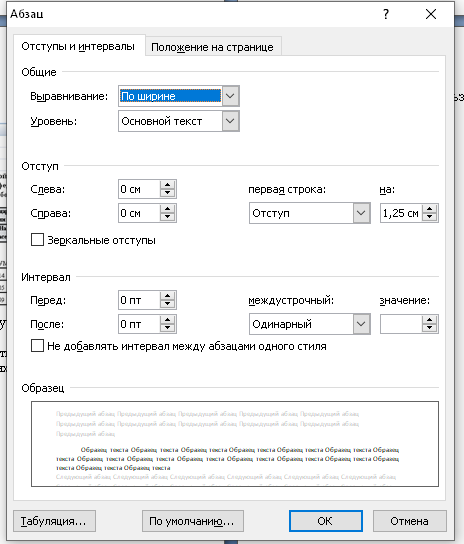
2. Содержание

3. Скриншоты кодов

4. Список использованных источников

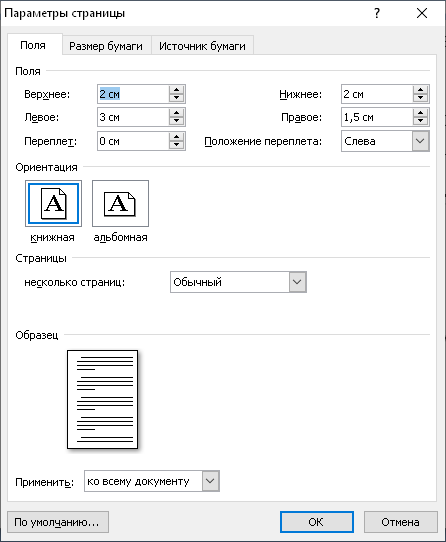
Соблюдайте требования к оформлению:

1. Шрифт- **TimesNewRoman**, размер – **14**



2. **Нумерация страниц** (титульный лист - без номера, содержание – 2 страница)

3. В параметрах страниц должны быть выставлены следующие значения:



4. Содержание создаётся с помощью инструмента **Автоматическое оглавление**. Содержание ОБЯЗАТЕЛЬНО оформить: Шрифт- TimesNewRoman, размер – 14, цвет – чёрный.

5. Каждая подтема должна начинаться с новой страницы, используем **Разрыв страницы**.



Содержание

1. Условный оператор

1.1. Выбор из двух

1.2. Логические операции

1.3. Вложенные и каскадные условия

2. Типы данных

2.1. Числовые типы данных: int, float

2.2. Строковый тип данных

Список использованных источников

1. Гуриков,С.Р.ОсновыалгоритмизацииипрограммированиянаPython: учеб.пособие/С.Р.Гуриков. —Москва:ФОРУМ: ИНФРА-М,2018. —343с.

2. Трофимов, В.В. Алгоритмизация и программирование: учебник для вузов /В.В.Трофимов,Т.А.Павловская;подредакциейВ.В.Трофимова.—Москва: Издательство Юрайт, 2021. — 137 с. — (Высшее образование). — ISBN 978-5-534-07834-3. — Текст : электронный // ЭБС Юрайт [сайт]. — URL:https://urait.ru/bcode/471125.