Глава 2

Принятие решений

Программы, с которыми вы работали до сих пор, выполнялись строго последовательно. Каждое последующее выражение выполнялось после завершения предыдущего, и все программы работали исключительно построчно сверху вниз. И хотя для простых задач, которые мы решали в первой части книги, такого линейного выполнения программ бывает вполне достаточно, для написания более сложных сценариев потребуется чуть больше.

Конструкции принятия решений позволяют программам при необходимости не выполнять определенные строки кода, тем самым производя ветвление. Выполнение программы по-прежнему происходит построчно сверху вниз, но некоторые строки могут быть просто пропущены. Это позволяет выполнять различные операции в зависимости от условий и серьезно повышает вариативность языка Python в отношении решения задач.

2.1. Выражения if

В программах, написанных на языке Python, ветвление осуществляется при помощи ключевого слова *if*. Выражение `if` включает в себя одно или несколько условий, а также тело выражения. При выполнении условного выражения происходит оценка заданного условия, на основании чего принимается решение, будут ли выполняться инструкции в теле выражения. Если результатом условного выражения будет `True` (Истина), то тело выполнится, после чего программа продолжится. Если же в результате проверки условия получится `False` (Ложь), тело будет пропущено, а выполнение программы продолжится с первой строки после тела.

Условия в выражении `if` могут быть достаточно сложными, а результат может принимать значение `True` или `False`. Такие выражения называются булевыми в честь Джорджа Буля (George Boole) (1815–1864) – пионера в области формальной логики.

Выражения `if` часто включают в себя *операторы отношения* (relational operator), сравнивающие значения, переменные или целые сложные вы-

ражения. Операторы отношения, присутствующие в языке Python, перечислены в табл. 2.1.

Таблица 2.1. Операторы отношения в языке Python

Оператор отношения	Значение
<	Меньше
<=	Меньше или равно
>	Больше
>=	Больше или равно
==	Равно
!=	Не равно

Тело выражения `if` может включать в себя одну или несколько инструкций, которые пишутся с отступом от ключевого слова `if`. Блок тела заканчивается, когда отступ снова выравнивается относительно слова `if`. Вы можете сами выбрать, какую величину отступа использовать в своих программах при написании условных выражений. Во всех фрагментах кода в данной книге используются отступы в четыре пробела, но вы можете использовать один или несколько пробелов на свой вкус. Большинство программистов придерживаются одного формата при формировании отступов в теле условных выражений, но в Python такая последовательность в действиях вовсе не обязательна.

В следующем фрагменте кода у пользователя запрашивается число, после чего следуют два условных выражения со своими телами, отделенными от выражения `if` двоеточием.

```
# Запрашиваем значение у пользователя
num = float(input("Введите число: "))
# Сохраняем подходящее значение в переменной result
if num == 0:
    result = "Введен ноль"
if num != 0:
    result = "Введен не ноль"
# Отобразим результат
print(result)
```

2.2. Выражения if-else

В предыдущей программе в переменную `result` записывалось одно строковое значение, если пользователь ввел ноль, и другое, если ввел число, отличное от нуля. Чаще всего условные выражения строятся таким образом, что всегда будет выполняться какое-то одно условие из перечисленных. В таких конструкциях два условия одновременно выполняться не

могут, но при этом одно из них всегда будет выполнено. Подобные условия называются *взаимоисключающими* (mutually exclusive).

Выражение *if-else* состоит из части `if` и соответствующего ему тела, а также части *else* – без условия, но со своим телом. Во время запуска программы происходит проверка условия. Если его результат будет `True`, выполнится тело условного выражения `if`, а тело `else` будет пропущено. В противном случае все будет наоборот: тело выражения `if` будет пропущено, а тело `else` выполнится. Тела обоих выражений одновременно не могут быть выполнены, как и не могут быть пропущены. В таких случаях, когда условия являются взаимоисключающими, можно использовать выражение `if-else` вместо двух последовательных `if`. Применение конструкции `if-else` более предпочтительно, поскольку в этом случае необходимо будет написать только одно условие, при выполнении программы будет выполняться только оно, и к тому же в случае возникновения ошибки или изменения логики программы исправлять вам придется лишь одно условие. Наша предыдущая программа, переписанная под использование выражения `if-else`, будет выглядеть следующим образом:

```
# Запрашивает значение у пользователя
num = float(input("Введите число: "))
# Сохраняем подходящее значение в переменной result
if num == 0:
    result = "Введен ноль"
else:
    result = "Введен не ноль"
# Отобразим результат
print(result)
```

При вводе пользователем нуля условное выражение выдаст результат `True`, и в переменную `result` будет записано нужное выражение. В этом случае выражение `else` полностью пропускается. В обратной ситуации, когда пользователь ввел число, отличное от нуля, результатом условного выражения будет `False`, так что пропущено будет тело выражения `if`, а выполнится тело выражения `else`, и в переменную попадет строка о том, что пользователь ввел ненулевое значение. В обоих случаях Python продолжит выполнение с инструкции `print`, в результате чего ответ будет выведен на экран.

2.3. Выражения if-elif-else

Выражение `if-elif-else` используется в случае выбора одной из нескольких альтернатив. Выражение начинается с части `if`, за которой следует часть `elif`, а завершается конструкция блоком `else`. Все эти части выраже-

ния должны содержать свои тела с правильными отступами. Кроме того, все инструкции `if` и `elif` должны включать в себя условия, результатом вычисления которых может быть либо `True`, либо `False`.

При выполнении конструкции `if-elif-else` первой запускается проверка условия в блоке `if`. Если результатом будет `True`, выполнится тело этого блока, после чего будут пропущены все оставшиеся части выражения. В противном случае Python перейдет к анализу следующего условия – из блока `elif`. Если оно верно, будет, как и ожидается, выполнено тело этого блока с последующим пропуском остальных инструкций. Если же нет, Python продолжит проверять все условия до конца выражения. Если ни одно из них не даст истины, будет выполнено тело из блока `else`, после чего выполнение программы продолжится.

Давайте расширим наш предыдущий пример, чтобы выполнялась проверка не только на ноль, но также на положительные и отрицательные значения. И хотя возможно решить эту задачу посредством выражений `if` и/или `if-else`, лучше всего подобные сценарии реализовывать при помощи конструкции `if-elif-else`, поскольку одно из трех условий должно выполняться в любом случае.

```
# Запрашиваем значение у пользователя
num = float(input("Введите число: "))
# Сохраняем подходящее значение в переменной result
if num > 0:
    result = "Введено положительное число"
elif num < 0:
    result = "Введено отрицательное число"
else:
    result = "Введен ноль"
# Отобразим результат
print(result)
```

При вводе пользователем положительного значения первое же условие даст результат `True`, а значит, будет выполнено тело этого блока, после чего программа продолжит выполнение с инструкции `print`.

При вводе отрицательного значения первое условие даст результат `False`, а значит, начнутся проверки следующих условий по очереди. Второе условие окажется выполнено, в результате чего будет запущено тело этого блока, после чего программа продолжит выполнение с инструкции `print`.

Наконец, если пользователь ввел нулевое значение, все блоки `if` и `elif` дадут `False`, а значит, будет выполнено тело блока `else`. После этого будет осуществлен вывод результата на экран.

Мы видим, что в конструкции `if-elif-else` всегда будет выполняться только одно тело в зависимости от прописанных условий.

2.4. Выражения if-elif

Надо отметить, что блок `else` в конструкции `if-elif-else` является необязательным. Его присутствие гарантирует выполнение одного из блоков выражения в любом случае. Если этот блок отсутствует, это значит, что может не выполниться ни один из блоков. Это произойдет, если ни одно из условий в блоках не даст значения `True`. Вне зависимости от того, будет ли выполнено хоть одно тело в выражении `if-elif`, программа продолжит свое выполнение со следующей за последней частью `elif` инструкции.

2.5. Вложенные выражения if

Блоки `if`, `elif` и `else` могут включать в себя любые выражения, включая другие блоки `if`, `if-else`, `if-elif` и `if-elif-else`. Если одно условное выражение появляется в теле другого, оно называется *вложенным* (nested). Следующий фрагмент кода иллюстрирует эту концепцию:

```
# Запрашиваем у пользователя число
num = float(input("Введите число: "))
# Сохраняем нужный нам результат
if num > 0:
    # Определяем, какое прилагательное нам стоит использовать
    adjective = " "
    if num >= 1000000:
        adjective = " очень большое "
    elif num >= 1000:
        adjective = " большое "

    # Сохраняем в переменную положительные числа с правильным прилагательным
    result = "Это" + adjective + "положительное число"
elif num < 0:
    result = "Это отрицательное"
else:
    result = "Это ноль"

# Выводим результат
print(result)
```

Программа начинается с запроса числа у пользователя. Если он ввел число, большее нуля, выполнится тело внешнего блока `if`. В нем мы сначала инициализируем переменную `adjective` пробелом. Затем запускается вложенная конструкция `if-elif`. Если пользователь ввел значение, превышающее или равное миллиону, присвоим переменной `adjective` строку `" очень большое "`. Если введенное значение укладывается в интервал

от 1000 до 999 999, переменная `adjective` получит строковое значение `" большое "`. После этого сохраняем результат в переменную `result`. Блоки `elif` и `else` из внешней условной конструкции выполняться не будут, поскольку уже было выполнено тело блока `if`. Наконец, программа выводит результат на экран.

Если бы пользователь ввел отрицательное значение или ноль, выполнился бы второй или третий блок внешней конструкции `if-elif-else` соответственно. В результате в переменной `result` оказалось бы другое значение, которое и было бы выведено на экран.

2.6. Булева логика

Булевым называется выражение, возвращающее только `True` или `False`. Это могут быть как сами булевы значения `True` и `False`, так и переменные, хранящие булевы выражения, операторы отношения, а также вызовы функций, возвращающих булево выражение. Булевы выражения также могут включать *булевы операторы* (Boolean operator), комбинирующие булевы значения. В Python предусмотрены три булевых оператора: `not`, `and` и `or`.

Оператор `not` обращает значение булева выражения. Если переменная `x` содержит значение `True`, то выражение `not x` вернет `False`, и наоборот.

Поведение всех булевых выражений может быть сведено в *таблицы истинности* (truth table). Такие таблицы содержат один столбец для булева значения переменной, а во втором показан результат примененного к ней булева выражения. Каждая строка в таблице отражает уникальное сочетание значений `True` и `False` для переменных в выражении. Таблица истинности для выражения, содержащего `n` различных переменных, будет состоять из 2^n строк, в каждой из которых будет показан результат вычисления выражения для конкретных значений переменных. Например, таблица истинности для оператора `not`, примененного к одной переменной `x`, будет включать две строки (2^1), как показано в табл. 2.2.

Таблица 2.2. Таблица истинности для оператора `not`

`x`	`not x`
`False`	`True`
`True`	`False`

Операторы `and` и `or` применяются сразу к двум булевым значениям при определении результата. Булево выражение `x and y` даст `True`, если `x` равен `True` и `y` тоже равен `True`. Если `x` равен `False` или `y` равен `False`, а также если оба значения `x` и `y` равны `False`, выражение `x and y` даст на выходе `False`. Таблица истинности для оператора `and` показана в табл. 2.3. В ней $2^2 = 4$ строки, поскольку оператор `and` применяется к двум переменным.

Таблица 2.3. Таблица истинности для оператора and

x	y	x and y
False	False	False
False	True	False
True	False	False
True	True	True

Булево выражение `x or y` даст `True`, если `x` равен `True` или `y` равен `True`, а также если обе переменные `x` и `y` содержат `True`. Это выражение дает `False`, только если `x` и `y` содержат `False`. Таблица истинности для оператора `or` показана в табл. 2.4.

Таблица 2.4. Таблица истинности для оператора or

x	y	x or y
False	False	False
False	True	True
True	False	True
True	True	True

В следующем фрагменте кода на Python показано использование булева оператора `or` для определения того, входит ли значение, введенное пользователем, в число первых пяти простых чисел. Операторы `and` и `not` могут быть использованы в похожей манере при построении сложных логических цепочек.

```
# Запрашиваем у пользователя целое число
x = int(input("Введите целое число: "))
# Определяем, входит ли значение в число первых пяти простых чисел
if x == 2 or x == 3 or x == 5 or x == 7 or x == 11:
    print("Это одно из первых пяти простых чисел.")
else:
    print("Это не одно из первых пяти простых чисел.")
```

2.7. Упражнения

Представленные здесь упражнения должны быть выполнены с использованием условных конструкций `if`, `if-else`, `if-elif` и `if-elif-else`, включая концепции, которые были изложены в первой главе. При решении некоторых задач вам также может понадобиться встроить блок `if` в другой блок `if`.

Упражнение 35. Чет или нечет?

(*Решено. 13 строк*)

Напишите программу, запрашивающую у пользователя целое число и выводящую на экран информацию о том, является введенное число четным или нечетным.

Упражнение 36. Собачий возраст

(*22 строки*)

Считается, что один год, прожитый собакой, эквивалентен семи человеческим годам. При этом зачастую не учитывается, что собаки становятся абсолютно взрослыми уже к двум годам. Таким образом, многие предпочитают каждый из первых двух лет жизни собаки приравнивать к 10,5 года человеческой жизни, а все последующие – к четырем.

Напишите программу, которая будет переводить человеческий возраст в собачий с учетом указанной выше логики. Убедитесь, что программа корректно работает при пересчете возраста собаки меньше и больше двух лет. Также программа должна выводить сообщение об ошибке, если пользователь ввел отрицательное число.

Упражнение 37. Гласные и согласные

(*Решено. 16 строк*)

Разработайте программу, запрашивающую у пользователя букву латинского алфавита. Если введенная буква входит в следующий список (`a`, `e`, `i`, `o` или `u`), необходимо вывести сообщение о том, что эта буква гласная. Если была введена буква `y`, программа должна написать, что эта буква может быть как гласной, так и согласной. Во всех других случаях должно выводиться сообщение о том, что введена согласная буква.

Упражнение 38. Угадайте фигуру

(*Решено. 31 строка*)

Напишите программу, определяющую вид фигуры по количеству ее сторон. Запросите у пользователя количество сторон и выведите сообщение с указанием вида фигуры. Программа должна корректно обрабатывать и выводить названия для фигур с количеством сторон от трех до десяти включительно. Если введенное пользователем значение находится за границами этого диапазона, уведомите его об этом.

Упражнение 39. Сколько дней в месяце?

(*Решено. 18 строк*)

Количество дней в месяце варьируется от 28 до 31. Очередная ваша программа должна запрашивать у пользователя название месяца и отображать количество дней в нем. Поскольку годы мы не учитываем, для февраля можно вывести сообщение о том, что этот месяц может состоять как из 28, так и из 29 дней, чтобы учесть фактор високосного года.

Упражнение 40. Громкость звука

(*30 строк*)

В табл. 2.5 представлен уровень громкости в децибелах для некоторых распространенных источников шума.

Таблица 2.5. Уровни громкости различных источников

Источник звука	Уровень громкости (дБ)
Отбойный молоток	130 дБ
Газовая газонокосилка	106 дБ
Будильник	70 дБ
Тихая комната	40 дБ

Создайте программу, в которой пользователь будет вводить уровень шума в децибелах. Если введенное им значение будет в точности совпадать с одним из значений в приведенной таблице, необходимо вывести, чему соответствует указанный уровень громкости. Если значение попадет между уровнями в таблице, нужно сообщить, между какими именно. Также программа должна выдавать корректные сообщения, в случае если введенное пользователем значение окажется ниже минимального или больше максимального.

Упражнение 41. Классификация треугольников

(*Решено. 21 строка*)

Все треугольники могут быть отнесены к тому или иному классу (равнобедренные, равносторонние и разносторонние) на основании длин их сторон. Равносторонний треугольник характеризуется одинаковой длиной всех трех сторон, равнобедренный – двух сторон из трех, а у разностороннего треугольника все стороны разной длины.

Напишите программу, которая будет запрашивать у пользователя длины всех трех сторон треугольника и выдавать сообщение о том, к какому типу следует его относить.

Упражнение 42. Узнать частоту по ноте

(*Решено. 39 строк*)

В табл. 2.6 перечислены частоты звуков, относящихся к одной октаве, начиная с до.

Таблица 2.6. Частоты нот одной октавы

Нота	Частота (Гц)
C4	261,63
D4	293,66
E4	329,63
F4	349,23
G4	392,00
A4	440,00
B4	493,88

Пусть ваша программа запрашивает у пользователя обозначение ноты и показывает ее частоту согласно приведенной таблице. После этого вы можете доработать свою программу таким образом, чтобы она поддерживала все октавы, начиная от субконтроктавы (C0) до пятой октавы (C8). И хотя можно это реализовать путем добавления бесконечного количества блоков `if`, это будет довольно громоздким, недостаточно элегантным и просто неприемлемым решением данной задачи. Вместо этого при расчетах лучше использовать отношения между одними и теми же нотами в соседствующих октавах. К примеру, частота любой ноты октавы n будет составлять ровно половину от частоты той же ноты октавы $n + 1$. Используя это соотношение, вы без труда сможете добавить в свою программу учет всех нот любой октавы без присутствия бесчисленных условных блоков.

Подсказка. Пользователь должен вводить ноту вместе с номером нотации октавы. Начните с разделения буквы, обозначающей ноту, и цифры, соответствующей номеру октавы. Затем определите частоту введенной ноты по представленной выше таблице и разделите ее на 2^{4-x}, где x – номер октавы в научной нотации, введенный пользователем. Это позволит умножить или разделить на два число из таблицы нужное количество раз.

Упражнение 43. Узнать ноту по частоте

(*Решено. 42 строки*)

В предыдущем упражнении мы определяли частоту ноты по ее названию и номеру октавы. Теперь выполним обратную операцию. Запроси-

те у пользователя частоту звука. Если введенное значение укладывается в диапазон ±1 Гц от значения в таблице, выведите на экран название соответствующей ноты. В противном случае сообщите пользователю, что ноты, соответствующей введенной частоте, не существует. В данном упражнении можно ограничиться только нотами, приведенными в таблице. Нет необходимости брать в расчет другие октавы.

Упражнение 44. Портреты на банкнотах

(*31 строка*)

Во многих странах существует традиция помещать портреты своих бывших политических лидеров или других выдающихся личностей на банкноты. В табл. 2.7 приведены номиналы банкнот США с изображенными на них людьми.

Таблица 2.7. Банкноты США

Портрет	Номинал банкноты
Джордж Вашингтон	$1
Томас Джефферсон	$2
Авраам Линкольн	$5
Александр Гамильтон	$10
Эндрю Джексон	$20
Улисс Грант	$50
Бенджамин Франклин	$100

Напишите программу, которая будет запрашивать у пользователя номинал банкноты и отображать на экране имя деятеля, портрет которого размещен на соответствующем денежном знаке. Если банкноты введенного номинала не существует, должно выводиться сообщение об ошибке.

Примечание. Хотя банкноты номиналом два доллара очень редко можно встретить в США, официально они существуют и могут быть использованы при любых расчетах. Также в начале прошлого века в Америке были выпущены в обращение банкноты номиналом $500, $1000, $5000 и $10 000, хотя с 1945 года они не печатались, а в 1969 году и вовсе были выведены из обращения. Так что их мы не будем рассматривать в данном упражнении.

Упражнение 45. Даты праздников

(*18 строк*)

В Канаде есть три национальных праздника, отмечающихся в одни и те же даты каждый год. Они приведены в табл. 2.8.

Таблица 2.8. Канадские праздники

Праздник	Дата
Новый год	1 января
День Канады	1 июля
Рождество	25 декабря

Напишите программу, которая будет запрашивать у пользователя день и месяц. Если введенные данные в точности указывают на один из перечисленных в таблице праздников, необходимо вывести его название. В противном случае сообщить, что на заданную дату праздники не приходятся.

Примечание. В Канаде есть два дополнительных национальных праздника: Страстная пятница и Праздник трудящихся, – которые выпадают на разные даты каждый год. Есть свои праздники и в отдельных провинциях – некоторые с фиксированными датами, другие – с плавающими. Мы не будем рассматривать такие праздники в этом упражнении.

Упражнение 46. Какого цвета клетка?

(*22 строки*)

Клетки на шахматной доске идентифицируются буквой и цифрой. Буква определяет положение клетки по горизонтали, а цифра – по вертикали, как показано на рис. 2.1.

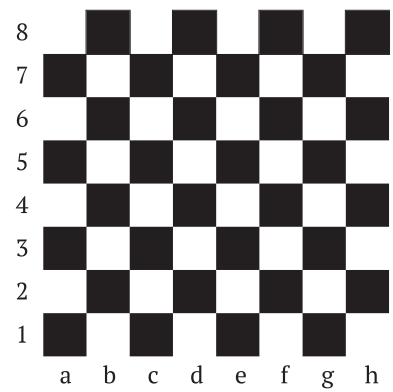

Рис. 2.1 ❖ Поля на шахматной доске

Ваша программа должна запрашивать у пользователя координаты клетки. Используйте условное выражение для определения того, с какой клет-

ки – белой или черной – начинается столбец. Затем при помощи обычной арифметики необходимо определить цвет конкретной клетки. Например, если пользователь ввел `a1`, программа должна определить, что клетка с этими координатами черная. Если `d5` – белая. Проверку на ошибочность ввода координат клетки выполнять не нужно.

Упражнение 47. Определение времени года

(*Решено. 43 строки*)

Год делится на четыре сезона: зима, весна, лето и осень. Хотя даты смены сезонов каждый год могут меняться из-за особенностей календаря, мы в данном упражнении примем допущения, перечисленные в табл. 2.9.

Таблица 2.9. Даты смены сезонов

Сезон	Первый день
Весна	20 марта
Лето	21 июня
Осень	22 сентября
Зима	21 декабря

Разработайте программу, запрашивающую у пользователя день и месяц – сначала месяц в текстовом варианте, затем номер дня. На выходе программа должна выдать название сезона, которому принадлежит выбранная дата.

Упражнение 48. Знаки зодиака

(*47 строк*)

В гороскопах, наполняющих газеты и журналы, астрологи пытаются положение солнца в момент рождения человека как-то связать с его судьбой. Всего насчитывается 12 знаков зодиака, и все они приведены в табл. 2.10.

Таблица 2.10. Знаки зодиака

Знак зодиака	Даты	Знак зодиака	Даты
Козерог	22 декабря – 19 января	Рак	21 июня – 22 июля
Водолей	20 января – 18 февраля	Лев	23 июля – 22 августа
Рыбы	19 февраля – 20 марта	Дева	23 августа – 22 сентября
Овен	21 марта – 19 апреля	Весы	23 сентября – 22 октября
Телец	20 апреля – 20 мая	Скорпион	23 октября – 21 ноября
Близнецы	21 мая – 20 июня	Стрелец	22 ноября – 21 декабря

Напишите программу, запрашивающую у пользователя дату его рождения и выводящую на экран соответствующий знак зодиака.

Упражнение 49. Китайский гороскоп

(*Решено. 40 строк*)

Китайский гороскоп делит время на 12-летние циклы, и каждому году соответствует конкретное животное. Один из таких циклов приведен в табл. 2.11. После окончания одного цикла начинается другой, то есть 2012 год снова символизирует дракона.

Таблица 2.11. Китайский гороскоп

Год	Животное	Год	Животное
2000	Дракон	2006	Собака
2001	Змея	2007	Свинья
2002	Лошадь	2008	Крыса
2003	Коза	2009	Бык
2004	Обезьяна	2010	Тигр
2005	Петух	2011	Кролик

Напишите программу, которая будет запрашивать у пользователя год рождения и выводить ассоциированное с ним название животного по китайскому гороскопу. При этом программа не должна ограничиваться только годами из приведенной таблицы, а должна корректно обрабатывать все годы нашей эры.

Упражнение 50. Шкала Рихтера

(*30 строк*)

В табл. 2.12 приведены диапазоны магнитуд землетрясений по шкале Рихтера с описаниями.

Таблица 2.12. Шкала Рихтера

Магнитуда	Описание землетрясения
Меньше 2,0	Минимальное
Больше или равно 2,0 и меньше 3,0	Очень слабое
Больше или равно 3,0 и меньше 4,0	Слабое
Больше или равно 4,0 и меньше 5,0	Промежуточное
Больше или равно 5,0 и меньше 6,0	Умеренное
Больше или равно 6,0 и меньше 7,0	Сильное
Больше или равно 7,0 и меньше 8,0	Очень сильное
Больше или равно 8,0 и меньше 10,0	Огромное
10.0 и больше	Разрушительное

Ваша программа должна запрашивать у пользователя магнитуду землетрясения по шкале Рихтера и выводить на экран описание уровня, соот-

ветствующего введенному значению. Например, если пользователь введет значение 5,5, нужно вывести сообщение о том, что этой магнитуде соответствует умеренный уровень землетрясения.

Упражнение 51. Корни квадратичной функции

(*24 строки*)

Общий вид квадратичной функции одной переменной имеет следующий вид: $f(x) = ax^2 + bx + c$, где a, b и c – константы и a не равна нулю. Корни этой функции могут быть извлечены путем нахождения таких значений переменной x, для которых будет соблюдаться равенство $ax^2 + bx + c = 0$. Эти значения могут быть вычислены с помощью формулы для корней квадратного уравнения, показанной ниже. Квадратичная функция может иметь от нуля до двух действительных корней.

$$\text{root} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Часть выражения под квадратным корнем называется дискриминантом. Если дискриминант отрицательный, квадратное уравнение не будет иметь действительных корней. В случае равенства дискриминанта нулю у квадратного уравнения будет ровно один действительный корень. Иначе корней будет два, и выражение необходимо будет вычислить дважды: один раз со знаком плюс, второй со знаком минус для числителя.

Напишите программу, вычисляющую действительные корни квадратичной функции. Сначала вы должны запросить у пользователя значения a, b и c. После этого должно быть выведено на экран количество действительных корней функции и их значения.

Упражнение 52. Буквенные оценки – в числовые

(*Решено. 52 строки*)

В разных странах успеваемость студентов в университетах ведется поразному: где-то в качестве оценок используются буквы, где-то цифры. Соответствие между ними приведено в табл. 2.13.

Таблица 2.13. Оценка успеваемости

Буквенная оценка	Числовая оценка	Буквенная оценка	Числовая оценка
A+	4,0	C+	2,3
A	4,0	C	2,0
A–	3,7	C–	1,7
B+	3,3	D+	1,3
B	3,0	D	1,0
B–	2,7	F	0

Напишите программу, которая будет принимать на вход буквенную оценку и выводить на экран соответствующую оценку в числовом выражении. Убедитесь в том, что программа генерирует понятное сообщение об ошибке при неверном вводе.

Упражнение 53. Числовые оценки – в буквенные

(*47 строк*)

В предыдущем упражнении мы переводили буквенные оценки студентов в числовые. Сейчас перевернем ситуацию и попробуем определить буквенный номинал оценки по его числовому эквиваленту. Убедитесь в том, что ваша программа будет обрабатывать числовые значения между указанными в табл. 2.13. В этом случае оценки должны быть округлены до ближайшей буквы. Программа должна выдавать оценку A+, если введенное пользователем значение будет 4,0 и выше.

Упражнение 54. Оценка работы

(*Решено. 30 строк*)

Представьте, что в компании проводится аттестация сотрудников в конце каждого года. Шкала рейтинга начинается на отметке 0,0, и чем лучше оценка, тем выше руководство оценивает сотрудника, а значит, тем больше будет его прибавка к зарплате. Рейтинг, присваиваемый сотрудникам, может составлять значения 0,0, 0,4 или 0,6 и выше. Значения между 0,0 и 0,4, а также между 0,4 и 0,6 никогда не используются. Значения, ассоциированные с каждым рейтингом, показаны в табл. 2.14. Прибавка к зарплате сотрудника рассчитывается как рейтинг, умноженный на $2400,00.

Таблица 2.14. Таблица рейтингов

Рейтинг	Значение
0,0	Низкий уровень
0,4	Удовлетворительный уровень
0,6 и выше	Высокий уровень

Напишите программу, которая будет запрашивать у пользователя рейтинг сотрудника и выводить соответствующее значение из приведенной таблицы. Также необходимо показать сумму прибавки сотрудника. При вводе некорректного значения рейтинга программа должна об этом сообщать.

Упражнение 55. Длины волн видимой части спектра

(*38 строк*)

Длины волн видимой части спектра колеблются от 380 до 750 нанометров (нм). И хотя сам спектр является непрерывным, его принято делить на шесть цветов, как показано в табл. 2.15.

Таблица 2.15. Длины волн по цветам

Цвет	Длина волны (нм)
Фиолетовый	Больше или равно 380 и меньше 450
Синий	Больше или равно 450 и меньше 495
Зеленый	Больше или равно 495 и меньше 570
Желтый	Больше или равно 570 и меньше 590
Оранжевый	Больше или равно 590 и меньше 620
Красный	Больше или равно 620 и меньше или равно 750

Запросите у пользователя длину волны и выведите на экран соответствующий ей цвет. Если введенное пользователем значение длины волны окажется за пределами видимой части спектра, сообщите об этом.

Упражнение 56. Определение частоты

(*31 строка*)

Электромагнитные волны можно классифицировать по частоте на семь категорий, как показано в табл. 2.16.

Таблица 2.16. Частоты электромагнитных волн

Категория	Диапазон частот (Гц)
Радиоволны	Менее 3×10^{9}
Микроволны	От 3×10^{9} до 3×10^{12}
Инфракрасное излучение	От 3×10^{12} до $4{,}3\times10^{14}$
Видимое излучение	От $4{,}3\times10^{14}$ до $7{,}5\times10^{14}$
Ультрафиолетовое излучение	От $7{,}5\times10^{14}$ до 3×10^{17}
Рентгеновское излучение	От 3×10^{17} до 3×10^{19}
Гамма-излучение	Более 3×10^{19}

Напишите программу, которая будет запрашивать у пользователя значение частоты волны и отображать название соответствующего излучения.

Упражнение 57. Счет за телефон

(*44 строки*)

Тарифный план мобильной связи включает в себя 50 минут разговоров и 50 смс-сообщений за $15,00 в месяц. Каждая дополнительная минута стоит $0,25, а каждое дополнительное сообщение – $0,15. Все счета за телефон включают налог на поддержку кол-центров 911 в размере $0,44, и общая сумма, включающая сумму отчислений кол-центрам, облагается налогом в размере 5 %.

Напишите программу, которая будет запрашивать у пользователя количество израсходованных за месяц минут разговора и смс-сообщений

и отображать базовую сумму тарификации, сумму за дополнительные минуты и сообщения, сумму отчислений кол-центрам 911, налог, а также итоговую сумму к оплате. При этом дополнительные звонки и сообщения необходимо выводить на экран только в случае их расходования. Убедитесь в том, что все суммы отображаются в формате с двумя знаками после запятой.

Упражнение 58. Високосный год?

(*Решено. 22 строки*)

В большинстве случаев год насчитывает 365 дней. Но на самом деле нашей планете требуется чуть больше времени, чтобы полностью пройти по своей орбите вокруг Солнца. В результате для компенсации этой разницы был введен дополнительный день в феврале для особых годов, называемых високосными. Определить, високосный год или нет, можно, следуя такому алгоритму:

- если год делится на 400 без остатка, он високосный;
- если год (из оставшихся) делится на 100 без остатка, он НЕ високосный;
- если год (из оставшихся) делится на 4 без остатка, он високосный;
- все остальные года не являются високосными.

Напишите программу, запрашивающую год у пользователя и выводящую сообщение о том, високосный ли он.

Упражнение 59. Следующий день

(*50 строк*)

Разработайте программу, принимающую на вход дату и выводящую на экран дату, следующую за ней. Например, если пользователь введет дату, соответствующую 18 ноября 2019 года, на экран должен быть выведен следующий день, то есть 19 ноября 2019 года. Если входная дата будет представлять 30 ноября, то на выходе мы должны получить 1 декабря. И наконец, если ввести последний день года – 31 декабря 2019-го, пользователь должен увидеть на экране дату 1 января 2020-го. Дату пользователь должен вводить в три этапа: год, месяц и день. Убедитесь, что ваша программа корректно обрабатывает високосные годы.

Упражнение 60. На какой день недели выпадает 1 января?

(*32 строки*)

Следующая формула может быть использована для определения дня недели, соответствующего 1 января заданного года:

$$\text{day\_of\_the\_week} = (\text{year} + \text{floor}((\text{year} - 1)/4) - \text{floor}((\text{year} - 1)/100) + \text{floor}((\text{year} - 1)/400))\%7.$$

В результате мы получим целое число, представляющее день недели от воскресенья (0) до субботы (6).

Используйте эту формулу для написания программы, запрашивающей у пользователя год и выводящей на экран день недели, на который в заданном году приходится 1 января. При этом на экран вы должны вывести не числовой эквивалент дня недели, а его полное название.

Упражнение 61. Действительный номерной знак машины?

(*Решено. 28 строк*)

Допустим, в нашей стране старый формат номерных знаков автомобилей состоял из трех заглавных букв, следом за которыми шли три цифры. После того как все возможные номера были использованы, формат был изменен на четыре цифры, предшествующие трем заглавным буквам.

Напишите программу, запрашивающую у пользователя номерной знак машины и оповещающую о том, для какого формата подходит данная последовательность символов: для старого или нового. Если введенная последовательность не соответствует ни одному из двух форматов, укажите это в сообщении.

Упражнение 62. Играем в рулетку

(*Решено. 45 строк*)

На игровой рулетке в казино 38 выемок для шарика: 18 красных, столько же черных и две зеленые, пронумерованные 0 и 00 соответственно. Красные числа на рулетке: 1, 3, 5, 7, 9, 12, 14, 16, 18, 19, 21, 23, 25, 27, 30 32, 34 и 36. Остальные номера в диапазоне положительных чисел до 36 – черные.

Игрок может сделать самые разные ставки. Мы в данном упражнении будем рассматривать лишь следующие виды ставок:

- одно число (от одного до 36, а также 0 или 00);
- красное или черное;
- четное или нечетное (заметьте, что номера 0 и 00 **не** являются ни четными, ни нечетными);
- от 1 до 18 против от 19 до 36.

Напишите программу, имитирующую игру на рулетке при помощи генератора случайных чисел в Python. После запуска рулетки выведите на экран выпавший номер и все сыгравшие ставки. Например, при выпадении номера 13 на экране должна появиться следующая последовательность строк:

```
Выпавший номер: 13…
Выигравшая ставка: 13
Выигравшая ставка: черное
Выигравшая ставка: нечетное
Выигравшая ставка: от 1 до 18
```

Если на рулетке выпал номер 0, вывод должен быть следующим:

```
Выпавший номер: 0…
Выигравшая ставка: 0
```

Для номера 00 сообщения будут иметь следующий вид:

```
Выпавший номер: 00…
Выигравшая ставка: 00
```