Программы, в которых нечего вычислять, довольно редки. Поэтому научиться программировать с помощью чисел никогда не бывает плохой идеей. Еще более ценным навыком, которому мы собираемся научиться, является обработка пользовательских данных. С ее помощью вы можете создавать интерактивные и гораздо более гибкие приложения. Итак, давайте начнем!

**Считывание чисел из пользовательского ввода**

Поскольку вы знакомы с input()функцией в Python, вряд ли для вас ново, что любые данные, передаваемые этой функции, обрабатываются как **строка**. Но как мы должны обращаться с числовыми значениями? Как правило, они явно преобразуются в соответствующие числовые типы:

integer = int(input())

floating\_point = float(input())

Обратите внимание на текущие рекомендации: крайне важно не называть переменные встроенными типами (скажем, **float** или **int**). Кроме того, мы должны учитывать ошибки пользователя: если пользователь вводит неточный ввод, скажем, строку "twoвместо числа2, ValueErrorпоявится a . На данный момент мы не будем заострять на этом внимание; но не волнуйтесь, более подробная информация об ошибках доступна в специальном разделе. Теперь рассмотрим более подробный и практичный пример обработки числовых входных данных.

**Бесплатные воздушные мили**

Представьте, что у вас есть кредитная карта с бонусной программой бесплатных авиамилей (или, может быть, она у вас уже есть). Как пользователь, вы должны ввести сумму денег, которую вы тратите в среднем с этой карты в месяц. Предположим, что бонусная программа дает вам 2 бесплатных мили за каждый потраченный доллар. Вот простая программа, позволяющая определить, когда вы можете путешествовать куда-либо бесплатно:

# the average amount of money per month

money = int(input("How much money do you spend per month: "))

# the number of miles per unit of money

n\_miles = 2

# earned miles

miles\_per\_month = money \* n\_miles

# the distance between London and Paris

distance = 215

# how many months do you need to get

# a free trip from London to Paris and back

print(distance \* 2 / miles\_per\_month)

Эта программа рассчитает, сколько месяцев потребуется, чтобы проехать выбранное расстояние и обратно.

Хотя рекомендуется писать сообщения для пользователей в input() функции, избегайте их в наших образовательных задачах по программированию, иначе ваш код может не пройти наши тесты.

**Расширенные формы заданий**

Всякий раз, когда вы используете знак равенства =, вы фактически присваиваете переменным некоторое значение. По этой причине = обычно упоминается как **оператор присваивания**. Между тем, есть другие операторы присваивания, которые вы можете использовать в Python. Их также называют **составными операторами присваивания**, поскольку они выполняют арифметическую операцию и присваивание за один шаг. Взгляните на фрагмент кода ниже:

# simple assignment

number = 10

number = number + 1 # 11

Этот код эквивалентен следующему:

# compound assignment

number = 10

number += 1 # 11

Из примера ясно видно, что второй фрагмент кода более лаконичен (поскольку он не повторяет имя переменной).

Естественно, аналогичные формы назначения существуют и для остальных арифметических операций: -=, \*=, /=, //=, %=, \*\*=. При наличии возможности используйте их для экономии времени и усилий.

Далее следует одно из возможных применений составного присваивания.

**Переменная счетчика**

В программировании есть концепция, называемая *циклом*. Он используется для повторения некоторого блока кода определенное количество раз. Довольно часто рядом с ними есть специальные переменные, называемые **счетчиками**. **Счетчик**, как следует из названия, что-то подсчитывает: сколько раз выполняется условие, сколько элементов в последовательности и т. Д. Следовательно, счетчики должны быть целыми числами. Теперь мы переходим к сути: вы можете использовать операторы +=и -=увеличивать или уменьшать счетчик соответственно.

Рассмотрим этот пример, в котором пользователь определяет значение, на которое увеличивается счетчик:

counter = 1

step = int(input()) # let it be 3

counter += step

print(counter) # it should be 4

В случае, если вам нужны только неотрицательные целые числа от пользователя (в конце концов, мы увеличиваем счетчик!), Вы можете предотвратить неправильные вводимые данные с помощью abs()функции. Это встроенная функция Python, которая возвращает абсолютное значение числа (то есть значение независимо от его знака). Давайте немного скорректируем нашу последнюю программу:

counter = 1

step = abs(int(input())) # user types -3

counter += step

print(counter) # it's still 4

Как вы можете видеть, благодаря abs()функции мы получили положительное число.

На данный момент все в порядке, что вы мало что знаете об упомянутых деталях **ошибок**, **циклов** и **встроенных функций** в Python. Мы догоним и убедимся, что вы знаете эти темы всесторонне. Продолжайте учиться!

**Краткие сведения**

Таким образом, мы пролили некоторый свет на новые подробности о целочисленной арифметике и обработке числовых входных данных в Python. Не стесняйтесь использовать их в своих будущих проектах. В этом разделе мы обсудили:

* как считывать числа из пользовательского ввода;
* как присваивать числа переменным и использовать арифметические операторы для присвоения результата вычисления;
* что такое счетчики и когда они используются.