2

ТЕЛЕПОРТАЦИЯ С ПОМОЩЬЮ ПЕРЕМЕННЫХ

Ну что, готовы овладеть силой языка Python, чтобы править миром Minecraft? В этой главе мы пробежимся по основам Python, а затем вы сможете применить свои знания и отправить игрока в телепортационный тур!

Описанные в этой главе понятия и приемы относятся не столько к программированию в игре Minecraft, сколько к программированию на языке Python в целом. С помощью полученных знаний вы сможете создавать любые другие Python-программы!

Что такое программа?

Программа — это набор инструкций, следуя которым компьютер выполняет определенную задачу или несколько задач. Вспомните приложение «Секундомер» на мобильном телефоне. Эта программа содержит инструкции, которые описывают, что должно произойти при нажатии кнопок «старт» и «стоп», а также как будет отображаться на экране счетчик времени. Все эти инструкции появились не сами по себе — их написал программист.

Люди по всему миру каждый день пользуются миллионами разных программ. Эсэмэски с телефона отправляет программа, светофором управляет программа, любая компьютерная игра, например Minecraft, — тоже программа.

Прочитав эту книгу, вы научитесь писать коды (те самые инструкции) на одном из языков программирования (Python) и с их помощью создавать собственный мир Minecraft.

Хранение информации в переменных

Начнем знакомство с языком Python с переменных. *Переменные* нужны для хранения данных, которые позже будут использованы в программе. *Данные* — это любая полезная информация: числа, имена, текст, списки значений и т. д. К примеру, в переменной `pickaxes` может храниться число 12:

Pickaxes — кирки

```
>>> pickaxes = 12
```

Хотя Python и позволяет давать переменным русскоязычные имена, делать так в среде программистов не принято. Используйте в названиях переменных латинские буквы и другие символы. Русскоязычный текст возможен лишь внутри строковых значений переменных (см. гл. 4) и в комментариях. Прим. науч. ред.

В переменных могут храниться даже предложения — такие как «Убирайся, крипер!». Значения переменных можно менять и тем самым проворачивать в Minecraft различные трюки. Уже скоро вы будете сами задавать значения переменным и отправлять игрока куда вам вздумается!

Чтобы создать переменную в Python, нужно ввести в окне консоли имя переменной, поставить знак «равно» (=), а после добавить значение этой переменной. Допустим, вы решили отправить игрока в путешествие по биомам. В этом случае ему понадобятся солидные запасы провизии. Назовем переменную `bread` и зададим ей значение 145:

Биом — природная зона в мире Minecraft

Bread — хлеб

```
>>> bread = 145
```

Имя переменной всегда пишется слева от знака «равно», а значение, которое в ней хранится, — справа (рис. 2.1). Эта строка кода означает, что мы *объявили* переменную `bread` и *присвоили* ей значение 145.

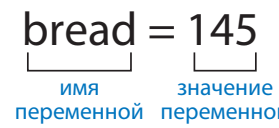

Рис. 2.1. Объявление переменной. Чтобы съесть 145 батонов хлеба, нужно быть очень голодным!

Объявив переменную и присвоив ей значение, вы можете проверить, что в ней хранится. Для этого введите в окне консоли имя переменной и нажмите ENTER:

```
>>> bread
145
```

Вы можете называть переменные как угодно, однако желательно, чтобы имя соответствовало назначению переменной. Так будет проще понять, что происходит в вашей программе. Имена переменных пишутся со строчной буквы. Это не правило языка, а соглашение, принятое Python-программистами. Ему нужно следовать, чтобы другим программистам было удобно читать ваш код.

Значения переменных хранятся во временной памяти компьютера, поэтому при его выключении или завершении программы эти данные исчезают. Попробуйте закрыть IDLE, а затем запустить программу снова. Что произойдет, если вы запросите значение переменной `bread`*?*

Как устроены языки программирования

У каждого языка программирования есть набор правил — *синтаксис*. Эти правила похожи на правила русского языка, которыми мы пользуемся, чтобы составлять предложения. Если вы будете знать синтаксис языка Python, ваши программы будут работать, а если нарушите их, компьютер просто не поймет, что вы от него хотите.

Каждая инструкция в вашем коде похожа на предложение. Только в русском языке конец предложения обозначается точкой, а в Python — переходом на новую строку. Строка с инструкцией называется *командой*.

Предположим, вы решили вести учет кирок (pickaxes), блоков железной руды (iron ore) и булыжников (cobblestone). Для этого можно ввести в окне консоли следующее:

```
>>> pickaxes = 12
>>> iron = 30
>>> cobblestone = 25
```

На рис. 2.2 показано, как это будет выглядеть.

Название блоков железной руды для простоты я сократил до одного слова. Обратите внимание, что каждый тип предметов записан отдельной строкой. Благодаря этому Python понимает, что вы хотите отслеживать количество трех разных предметов. Если вы запишете все команды в одну строку, Python запутается и сообщит о синтаксической ошибке:

```
>>> pickaxes = 12 iron = 30 cobblestone = 25
SyntaxError: invalid syntax
```

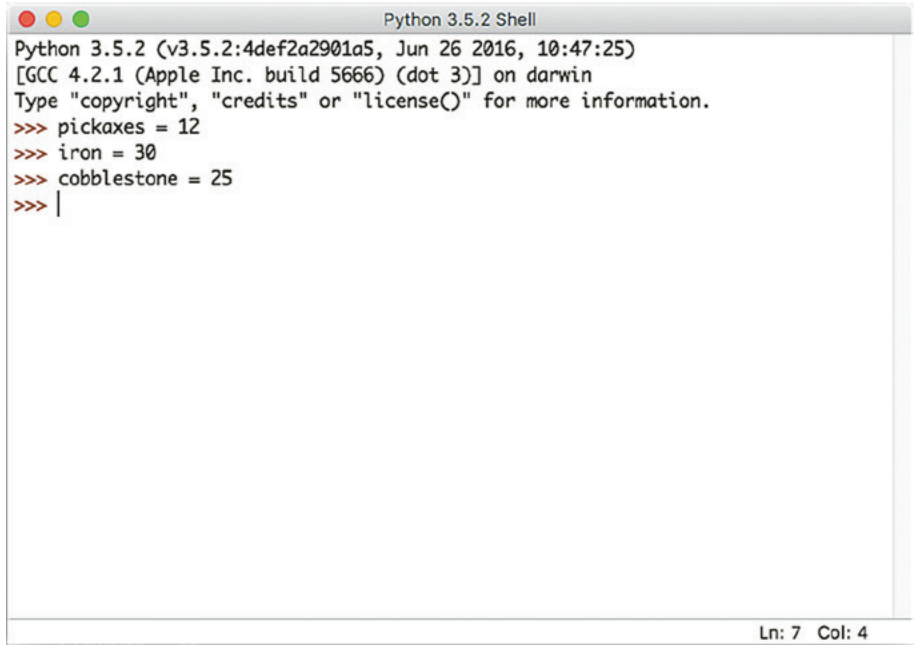

Рис. 2.2. Ввод команд в окне консоли Python

Синтаксическая ошибка (*SyntaxError*) означает, что Python не понимает ваших команд и не может их выполнить. В данном случае программе неясно, где начинается и где заканчивается каждая команда. Такая ошибка называется «неверный синтаксис» (invalid syntax).

Также Python не поймет, что нужно делать, если в начале строки будет стоять пробел:

```
>>>  iron = 30
SyntaxError: unexpected indent
```

Присмотритесь, и вы заметите, что строка начинается с отступа. Синтаксическая ошибка «некорректный отступ» (unexpected indent) означает, что в начале строки есть пробел или пробелы, которых там быть не должно.

Python очень требователен к тому, как вы пишете код. Если при вводе примеров из этой книги вы увидите сообщение о синтаксической ошибке, внимательно его проверьте — скорее всего, в код закралась ошибка.

Синтаксис для переменных

Имена, которые вы будете давать переменным, тоже должны подчиняться синтаксису языка Python. Имейте в виду:

- В имени не должно быть других символов, кроме латинских букв, цифр и нижнего подчеркивания (_).
- Не начинайте имя с цифры. `9bread` не подойдет. Однако в любом другом месте цифра стоять может — например, здесь: `bread9`.
- До и после знака «равно» лучше ставить пробел. Это не обязательно, но так принято делать в среде программистов, чтобы легче читать код.

Переменные очень полезны. Давайте разберемся, как менять их значения, и вы научитесь телепортировать игрока!

Изменение значений переменных

Вы можете в любой момент изменить значение переменной. Например, вы встретили в мире Minecraft пять кошек и хотите сохранить это число. Первым делом объявите переменную `cats` и присвойте ей значение 5. В окне консоли это будет выглядеть так:

Cats — кошки

```
>>> cats = 5
>>> cats
5
```

Вдруг вы встречаете еще пять кошек и хотите обновить содержимое переменной. Что будет, если вы присвоите уже существующей переменной значение 10?

```
>>> cats = 10
>>> cats
10
```

Запросив значение переменной, вы увидите на экране вовсе не цифру 5! И теперь, если вы введете переменную `cats` в какую-то команду, ей будет соответствовать число 10.

В переменных можно хранить данные самых разных типов. *Тип данных* говорит компьютеру о том, как именно нужно обрабатывать каждое конкретное значение. Сначала я расскажу о целых числах — этот тип данных встречается чаще всего. А позже познакомлю с дробными числами.

Целые числа

Целые числа бывают положительными и отрицательными. Числа 10, 32, –6, 194689 и –5 целые, а 3,4 и 6,025 — дробные.

Мы имеем дело с целыми числами каждый день, порой не задумываясь об этом. Встречаются они и в игре Minecraft. Направляясь в шахту, чтобы добыть 5 алмазов, игрок может захватить с собой 2 яблока и повстречать по пути 12 коров. Все эти числа — целые.

Предположим, в вашем Minecraft-мире есть 5 свиней и вы решили написать программу, которая каким-то образом будет использовать это число. Объявите целочисленную переменную `pigs` и присвойте ей значение, которое соответствует количеству свиней:

Pigs — свиньи

```
>>> pigs = 5
```

В переменных можно хранить и отрицательные числа. Например, чтобы обозначить температуру −5 градусов, можно объявить такую переменную:

```
>>> temperature = -5
```

А теперь настало время применить знания о переменных и целых числах, чтобы выполнить первую миссию!

МИССИЯ 1. ТЕЛЕПОРТАЦИЯ ИГРОКА

В ходе этой миссии вы научитесь использовать переменные и работать с целыми числами, создав программу телепортации игрока.

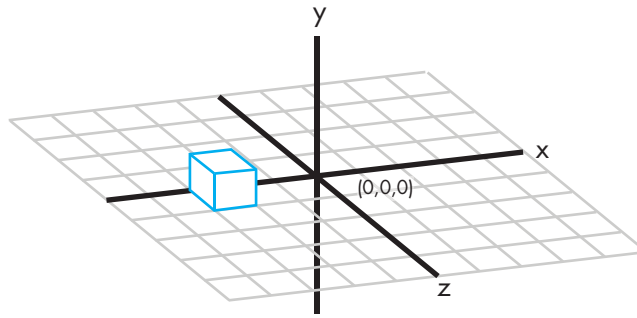

Рис. 2.3. Оси координат в трехмерном пространстве

На рис. 2.3 видно, что позиция игрока в мире Minecraft определяется тремя *координатами*: *x* («икс»), *y* («игрек») и *z* («зэт»). Координата *y* соответствует высоте, *x* и *z* — положению на горизонтальной плоскости.

Если вы используете версию Minecraft для Raspberry Pi, координаты игрока — это три цифры в левом верхнем углу экрана (рис. 2.4). Если же у вас версия Minecraft для Windows или Mac OS, нажмите во время игры клавишу F3, чтобы увидеть координаты — они будут во втором блоке текста слева, после букв XYZ (рис. 2.5).

Рис. 2.4. Координаты игрока в Minecraft для Raspberry Pi

Подвигайте вашего игрока туда-сюда и понаблюдайте, как меняются его координаты. Здорово, правда? Однако перемещения на большие расстояния занимают много времени. Зачем тратить его зря, если игрока можно телепортировать? Просто измените его координаты на любые другие с помощью Python-программы.

Рис. 2.5. Координаты игрока в Minecraft для настольных компьютеров

Выполните следующие шаги:

1. Откройте IDLE и кликните **File → New File** (для некоторых версий — **New Window**). Вы увидите пустое окно программы (рис. 2.6). Если у вас Raspberry Pi или на компьютере установлено несколько версий Python, убедитесь, что открылся именно Python 3, а не Python 2.7.

New window — новое окно

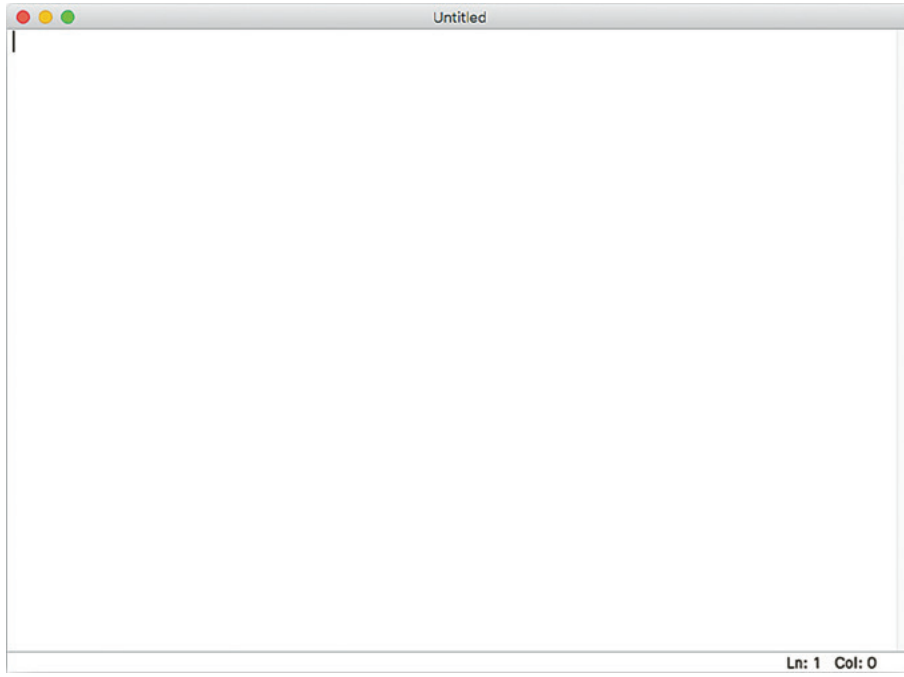

Рис. 2.6. Новое окно программы

2. Кликните **File → Save As**. Откроется диалоговое окно.
3. Зайдите в нем в папку *Minecraft Python*, которую вы создали, выполняя инструкции из главы 1, и добавьте в нее папку под названием *variables*.
4. Откройте папку *variables*, назовите сохраняемый файл *teleport.py* и кликните **Save**.

Variables — переменные

Teleport — телепортация

Теперь переключитесь на окно программы IDLE и введите туда следующие две строчки кода:

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
```

Эти команды нужны для подключения к игре Minecraft, и они должны быть в каждой Python-программе, которая с ней взаимодействует. Затем объявите три целочисленные переменные с именами `x`, `y` и `z`.

```
x = 10
y = 110
z = 12
```

Эти переменные соответствуют координатам, в которые вы собираетесь телепортировать игрока. Пока присвоим им значения 10, 110 и 12.

Далее введите следующую строку кода — она и будет телепортировать игрока:

```
mc.player.setTilePos(x, y, z)
```

Выражение `setTilePos()` — это *функция*, то есть фрагмент ранее созданного программистами кода. Функция `setTilePos(x, y, z)` дает игре команду изменить координаты игрока на значения, указанные в скобках, то есть на значения только что созданных вами переменных. Эти значения в скобках называются *аргументами*. Таким образом, с помощью аргументов вы передаете значения переменных `x`, `y` и `z` в функцию, чтобы после *вызова* она могла ими воспользоваться.

Set tile pos (pos — *сокр.* от position) — задать позицию блока

Если у вас версия игры для Raspberry Pi, не используйте в качестве координат `x` и `z` числа больше 127 и меньше −127. Мир Minecraft Pi невелик, и указание координат, находящихся за его пределами, приведет к сбою игры.

В листинге 2.1 дан полный код для телепортации игрока.

Листинг — текст компьютерной программы. *Прим. науч. ред.*

teleport.py

```
❶ # Подключаемся к Minecraft
from mcpi.minecraft import Minecraft
mc = Minecraft.create()

# Присваиваем переменным x, y и z значения координат
x = 10
y = 110
z = 12

# Меняем позицию игрока
mc.player.setTilePos(x, y, z)
```

Листинг 2.1. Код для телепортации игрока

Чтобы в коде было проще разобраться, я добавил в него комментарии ❶. *Комментарии* — полезная конструкция языка программирования. Они позволяют пояснить, что делает каждая часть кода. Python, выполняя программу, эти пояснения игнорирует. Иными словами, когда вы запустите свой код, Python пропустит эти строки, будто их нет. Однострочный комментарий начинается с символа «решетка» (#).

В комментариях листинга 2.1 я описал, что происходит в каждой части программы *teleport.py*. Советую вам тоже завести привычку писать комментарии к своим программам, чтобы потом было проще вспомнить, как они устроены.

Рис. 2.7. Полный текст кода в окне программы IDLE

На рис. 2.7 показан полный текст программы в окне программы IDLE. Давайте же запустим эту программу! Сделайте следующее:

1. Откройте Minecraft, кликнув по его иконке на рабочем столе.

Start game — начать игру

Create a new world — создать новый мир

2. Если у вас Raspberry Pi, кликните **Start Game**, а затем **Create a New World**. Если же у вас версия Minecraft для настольных компьютеров, откройте игровой мир, как написано в разделе «Запуск Spigot и создание профиля игры» (на с. 26 для Windows и на с. 39 для Mac OS).

3. Когда мир будет создан, нажмите клавишу ESC (или TAB, если у вас Raspberry Pi), чтобы появился курсор мыши. Теперь вы можете выйти из игры, переместив курсор за пределы окна Minecraft, и вернуться в игру, сделав двойной клик по окну Minecraft. На рис. 2.8 показаны окно Minecraft и окно программы IDLE на экране моего компьютера.

Рис. 2.8. Окно Minecraft и окно IDLE на экране моего компьютера

4. Кликните внутри окна программы с кодом программы *teleport.py*.

5. Кликните **Run → Run Module** или нажмите F5. Если вы не сохранили программу, IDLE попросит вас это сделать — кликните в появившемся диалоговом окне **OK.** Если вы кликнете **Cancel**, программа не запустится.

В среде IDLE на Raspberry Pi диалоговое окно с запросом на сохранение программы может открыться позади окна Minecraft, так что его не будет видно. Если вам кажется, что программа IDLE зависла, возможно, проблема в этом. Тогда сверните окно Minecraft, кликните OK в диалоговом окне IDLE, а затем снова разверните окно Minecraft.

Отлично! Теперь программа должна запуститься, и через несколько секунд ваш игрок телепортируется в новое место с координатами (10, 110, 12). У меня он оказался над болотом (рис. 2.9). Поскольку мир Minecraft на вашем компьютере отличается от моего, ваш игрок окажется в другой обстановке.

БОНУСНОЕ ЗАДАНИЕ: ФИГАРО ЗДЕСЬ, ФИГАРО ТАМ

Ну что, разобрались, как работает телепортация? Присвойте переменным `x`, `y` и `z` другие значения и посмотрите, где окажется игрок. Попробуйте ввести отрицательные числа!

Рис. 2.9. Я телепортировал игрока из дома в место с координатами (10, 110, 12), и он оказался над болотом. Караул!

Вещественные числа

Не все числа целые. Существуют и дробные числа. В десятичных дробях дробную часть от целой принято отделять запятой. Например, половину

яблока можно записать так: 0,5. В языке Python дробные числа записываются через точку, а называются *вещественными*. И это еще один тип данных наряду с целыми числами. Кстати, целое число вполне можно записать в виде дробного (например, так: 3.0), но дробное число с цифрами после точки нельзя записать в виде целого числа. Вещественные числа используют вместо целых, когда нужна бóльшая точность.

Возможно, вы уже заметили, что в значениях координат игрока на рис. 2.4 и 2.5 есть цифры после точки, а значит, это вещественные числа!

В Python вы можете присвоить переменной значение в виде вещественного числа — точно так же как вы делали это с целыми числами. Например, чтобы задать переменной `x` значение 1,34, нужно ввести:

```
>>> x = 1.34
```

А чтобы сделать это вещественное число отрицательным, просто поставьте перед ним знак «минус»:

```
>>> x = -1.34
```

В следующей миссии вы обретете полную власть над телепортацией, потому что научитесь перемещать игрока в позицию с точными координатами.

МИССИЯ 2. ПЕРЕМЕЩЕНИЕ В ТОЧНОСТИ ТУДА, КУДА НАДО

Вы уже умеете задавать позицию игрока с помощью целых чисел, однако его координаты можно указать куда более точно, если воспользоваться вещественными числами. В ходе этой миссии вы измените код из миссии 1 и телепортируете игрока в место с вещественными значениями координат. Для этого сделайте следующее:

1. Откройте в IDLE программу *teleport.py* (см. с. 61), кликнув **File → Open** и выбрав соответствующий файл в папке *variables*.
2. Сохраните файл под именем *teleportPrecise.py* (туда же, в папку *variables*).
3. В коде программы *teleportPrecise.py* присвойте переменным `x`, `y` и `z` вещественные значения, введя вместо 10, 110 и 12 числа 57.0, 103.0 и 31.0.
4. Поменяйте последнюю строчку на `mc.player.setPos(x, y, z)`, убрав из команды слово `Tile`.

Teleport precise — точная телепортация

Tile — блок

5. Сохраните код.

6. Откройте Minecraft и запустите программу.

В законченном виде ваш код должен выглядеть так:

teleportPrecise.py

```
# Подключаемся к Minecraft
from mcpi.minecraft import Minecraft
mc = Minecraft.create()

# Присваиваем переменным x, y и z значения координат
x = 57.0
y = 103.0
z = 31.0

# Меняем позицию игрока
mc.player.setPos(x, y, z)
```

Set pos (pos — *сокр.* от position) — задать позицию

Чем отличается команда `mc.player.setPos(x, y, z)` из нового кода от команды `mc.player.setTilePos(x, y, z)` из листинга 2.1? Известно, что пространство Minecraft состоит из блоков. Функция `setTilePos()` с помощью целых чисел указывает игре координаты блока, в который нужно телепортировать игрока. Функция же `setPos()` делает немного иное — при помощи вещественных чисел она задает не только координаты блока, но и точную позицию игрока внутри этого блока. Запустив программу у себя, я переместил игрока на вершину башни (рис. 2.10).

Рис. 2.10. Присвоив переменным точные вещественные значения координат, я телепортировал игрока на вершину башни

БОНУСНОЕ ЗАДАНИЕ: ТОЧНАЯ ТЕЛЕПОРТАЦИЯ

Присвойте переменным `x`, `y` и `z` какие-нибудь положительные и отрицательные вещественные значения и запустите программу. Затем измените цифры после точек и снова запустите. Что произойдет?

Замедление телепортации с помощью модуля time

Python выполняет команды кода почти мгновенно. Однако вы можете замедлить процесс, научив вашу программу делать паузу в несколько секунд.

Time — время

Воспользуйтесь модулем `time`, который содержит готовые функции для работы со временем. Для этого добавьте в самый верх программы следующую строку:

```
import time
```

Sleep — уснуть

Важно помнить, что модуль `time` должен быть импортирован (с помощью команды `import`) раньше всех его функций. Например, функции `sleep()`, которая позволяет приостановить выполнение программы на указанное число секунд. В противном случае Python не сможет найти функцию `sleep()` и, не зная, что делать дальше, остановит вашу программу. Именно поэтому команды импорта всех необходимых вам модулей лучше помещать в начале кода. Я, например, в первых двух строках кода всегда пишу команды для подключения к игре, а третьей строкой импортирую модуль `time`.

Вот пример использования вызова одной из функций модуля `time` — функции `sleep()`:

```
time.sleep(5)
```

Эта команда приостановит выполнение программы на пять секунд. В качестве аргумента функции `sleep()` вы можете использовать любые числа — как целые, так и вещественные. Например:

```
time.sleep(0.2)
```

Когда запущенная программа дойдет до этой строчки, она приостановит свою работу на 0,2 секунды.

Итак, теперь вы можете управлять временем, а значит, настал черед следующей миссии!

МИССИЯ 3. ТЕЛЕПОРТАЦИОННЫЙ ТУР

Прелесть телепортации состоит в том, что вы можете переместить игрока куда угодно! Попробуйте в этой миссии применить все свои знания, полученные ранее, чтобы отправить игрока в телепортационный тур по миру Minecraft!

Вам предстоит изменить код из миссии 1 так, чтобы ваш игрок смог побывать в разных областях игрового мира. Он должен сначала телепортироваться в одно место, а затем, после паузы, — в другое.

1. Откройте в IDLE программу *teleport.py* (с. 61). Для этого кликните **File → Open** и выберите соответствующий файл в папке *variables*.

Tour — путешествие

2. Сохраните программу в той же папке под новым именем *tour.py*.

Import time — импортировать (загрузить) модуль time

3. Сразу после строк кода для подключения к Minecraft добавьте команду `import time`.
4. В конце кода добавьте команду `time.sleep(10)`.
5. Скопируйте строки, в которых объявляются переменные `x`, `y` и `z`, а также строку с функцией `setTilePos()` и вставьте их в конец программы, чтобы они повторялись дважды.
6. Поменяйте все значения `x`, `y` и `z` на любые целые числа. Вы можете узнать координаты любого места в мире Minecraft, перейдя туда и посмотрев нужные значения (ранее в этой главе мы выяснили, как это сделать).
7. Сохраните программу.
8. Откройте Minecraft и запустите программу.

Ваш код должен выглядеть так:

tour.py

```
# Подключаемся к Minecraft
from mcpi.minecraft import Minecraft
mc = Minecraft.create()
import time

# Присваиваем переменным x, y и z значения координат
x = # Впишите значение
y = # Впишите значение
z = # Впишите значение

# Меняем позицию игрока
mc.player.setTilePos(x, y, z)
```

```
# Ждем 10 секунд
time.sleep(10)

# Присваиваем переменным x, y и z значения координат
x = # Впишите значение
y = # Впишите значение
z = # Впишите значение

# Меняем позицию игрока
mc.player.setTilePos(x, y, z)
```

Рис. 2.11. Я задал такие координаты, чтобы мой игрок телепортировался сначала в дом, а затем в пустыню

Только вместо комментариев напротив переменных должны стоять конкретные числа.

После запуска программы игрок должен оказаться в месте с первым набором координат, а через 10 секунд — в месте со вторым набором координат (рис. 2.11).

Затем измените код так, чтобы при каждой следующей телепортации менялось значение лишь одной переменной: `x`, `y` или `z`. Необязательно всегда изменять значение всех переменных! А еще попробуйте использовать вместо целых чисел вещественные.

БОНУСНОЕ ЗАДАНИЕ: ЕЩЕ БОЛЬШЕ ТЕЛЕПОРТАЦИИ

Скопируйте и вставьте строчки кода из программы *tour.py* столько раз, сколько раз вы хотите телепортировать игрока. Замените 10 внутри функции `time.sleep(10)` на другое значение. Можете даже использовать разные числа, чтобы игрок проводил в каждом месте разное количество времени.

Отладка

Все мы совершаем ошибки, и зачастую даже лучшие программисты не могут с ходу написать правильный код. Поэтому умение создавать работающие программы — лишь один из навыков, необходимых хорошему разработчику. Другое важное умение — поиск и устранение ошибок в коде. Этот процесс называется *отладкой*. Далее мы рассмотрим приемы, которые помогут вам отлаживать свои программы.

Ошибки в программе могут либо остановить ее работу, либо заставить работать некорректно. Если программа не запускается, Python выдаст *сообщение об ошибке* (рис. 2.12).

На рис. 2.12 видно, что я ввел в окне консоли Python некий код и получил сообщение об ошибке. В этом сообщении содержится немало информации, но если вы обратите внимание на последнюю строчку — *`NameError: name 'x' is not defined`*, — то заметите, что с переменной `x` что-то не так. Собственно говоря, в коде отсутствует объявление этой переменной. Чтобы исправить эту ошибку, нужно добавить в код соответствующую команду:

Name error: name 'x' is not defined — код ошибки: имя 'x' не определено

```
>>> x = 10
```

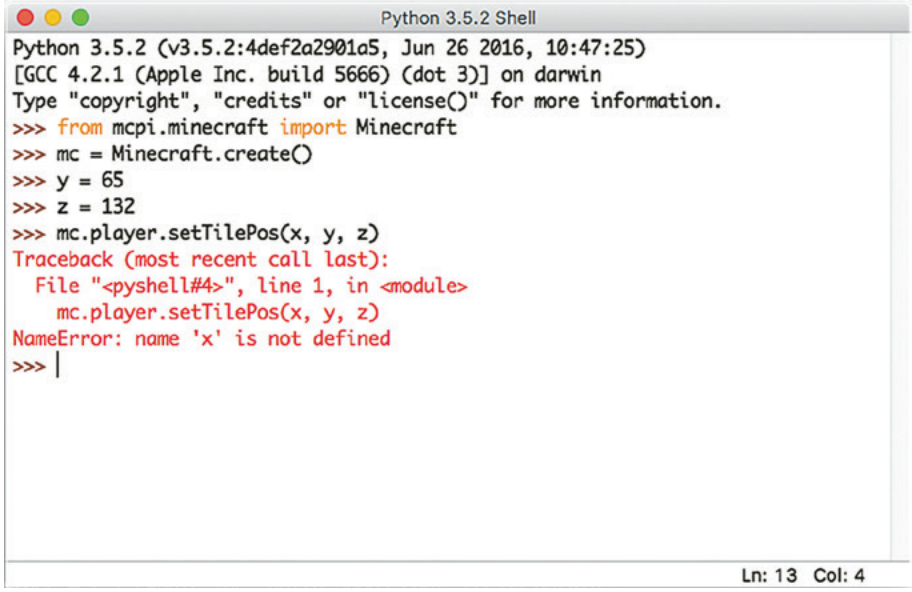

Рис. 2.12. Python показывает, что ошибка произошла из-за нарушения синтаксиса

Она устранит ошибку, и сообщение исчезнет. Но это совсем не значит, что теперь в программе все работает идеально.

Встречаются ошибки, которые не прерывают работу программы и не приводят к выводу сообщений, однако конечный результат говорит о том, что в программе что-то не так. Например, если в коде для телепортации игрока пропустить строку с телепортирующей функцией, такой как `setTilePos()`, программа спокойно запустится и завершится без единого сообщения об ошибке, но положение игрока не изменится. То есть программа получится совершенно бесполезной!

Чаще всего ошибки возникают из-за опечаток. Если ввести команду так, что компьютер не сможет ее понять, программа работать не будет. Пишите код внимательно и помните, что для Python важно, какие буквы вы используете — строчные или прописные!

МИССИЯ 4. ИСПРАВЬТЕ НЕРАБОТАЮЩИЙ ТЕЛЕПОРТАТОР

В ходе этой миссии вам предстоит отладить две программы. Первая программа (листинг 2.2) напоминает программу *teleport.py* со с. 61, однако в этой версии есть несколько ошибок. Откройте в окне программы IDLE новый файл, введите код листинга 2.2 и сохраните под именем *teleportBug1.py*.

Bug — ошибка в программе, баг

teleportBug1.py

```
from mcpi.minceraft import Minecraft
# mc = Minecraft.create()

x = 10
  y = 11
z = 12
```

Листинг 2.2. Первая неработающая программа телепортации

Чтобы отладить программу, выполните следующее:

1. Запустите *teleportBug1.py*.
2. Получив сообщение об ошибке, изучите его последнюю строку — она подскажет, в чем заключается проблема.
3. Исправьте ошибку и снова запустите программу.
4. Продолжайте исправлять ошибки до тех пор, пока программа не телепортирует игрока в место с заданными координатами.

Проверьте, есть ли в вашем коде функция телепортации `setTilePos()`*!*

Теперь отладьте еще одну программу. Код листинга 2.3 запускается, но по какой-то причине игрок перемещается в место, которое не соответствует значениям координат *x*, *y* и *z*. Введите этот код в новый файл IDLE и сохраните как *teleportBug2.py*.

teleportBug2.py

```
from mcpi.minecraft import Minecraft
mc = Minecraft.create()

x = 10
y = 110
z = -12

mc.player.setPos(x, z, y)
```

Листинг 2.3. Вторая неработающая программа телепортации

В отличие от кода *teleportBug1.py*, при запуске этой программы вы не получите сообщения об ошибке. Чтобы ее найти и исправить,

внимательно изучайте код, пока не обнаружите неверно записанную команду. Эта программа должна телепортировать игрока в место с координатами 10, 110, –12. Запустив ее, проверьте координаты места, куда на самом деле перенесся игрок, — это поможет найти ошибку.

Отладив обе программы и добившись их правильной работы, добавьте в них комментарии, описывающие, в чем именно заключались ошибки. Это послужит напоминанием и поможет в будущем избежать подобных проблем.

Что вы узнали

Поздравляю! Вы написали свою первую Python-программу, которая перемещает игрока при помощи переменных и функций, узнали о двух типах данных (целые и вещественные числа), попробовали управлять временем и отладили неработающие программы. Также вы познакомились с двумя телепортирующими функциями Minecraft Python API: `setPos()` и `setTilePos()`.

В главе 3 вы научитесь мгновенно возводить постройки при помощи математических операций и функций установки блоков.

издательство
МАНН, ИВАНОВ И ФЕРБЕР
Максимально полезные книги