Оглавление

ГЛАВА 1
Подготовительная работа

На сегодняшний день есть два общеизвестных варианта настройки *Python* и *Minecraft* для интегрированной работы.

1. Установить *Python 3.X*, *Minecraft Java Edition*, выбрав версии *Minecraft 1.12.2* или *Minecraft 1.19*, а также скачать сервер *Spigot.*

2. Использовать микрокомпьютер *Raspberry PI* и операционную систему *Raspbian*.

Самый простой — это второй способ, если у вас есть микрокомпьютер *Raspberry PI*. Данный способ хорош ещё и тем, что вместе с операционной системой *Raspbian* идёт игра *Minecraft* совершенно бесплатно. Более подробную информацию вы можете получить с официального сайта https://www.raspberrypi.org/. Этот способ мы не будем рассматривать, потому что микрокомпьютеры *Raspberry PI* не очень распространены в России. В другой книге мы обязательно изучим этот микрокомпьютер.

Первый способ можно использовать не только для *Windows*, но и для *Mac OS X*.

Мы будем использовать первый способ. Вне зависимости от выбора варианта настройки все программы будут работать одинаково на любой из обозначенных версий *Minecraft* и операционных систем, так как не задействуют специфические функции из библиотек *Windows*.

Мы постарались сделать книгу похожей на учебник, чтобы читатель не только познакомился с языком *Python*, но и смог закрепить свои знания и навыки и перенести их на более взрослые и реальные проекты, связанные с программированием.

Теперь перед вами открывается удивительный, волшебный мир магии программирования. Весь *Minecraft* у ваших ног. Мы начинаем путешествие.

Желаем удачи!

1.1. Установка Python 3

Изучение языка программирования *Python* во многих учебниках и курсах начинается с установки редактора кода. В этом плане данный курс не исключение. Первый наш урок будет заключаться в том, чтобы установить редактор кода.

Пошаговая инструкция

1. Перейти на официальный сайт разработчиков языка *Python* https://www.python.org/. Перед нами появляется такой сайт (рис. 1):

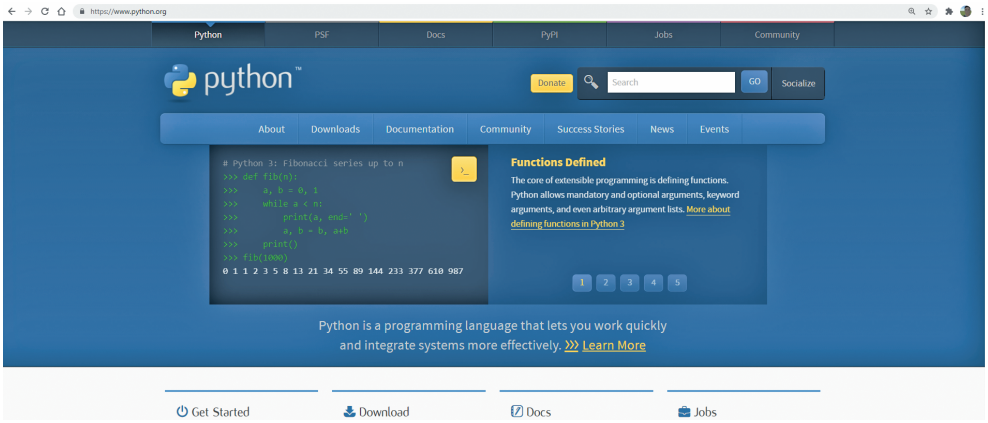


Рис. 1. Официальный сайт *Python*

Допишите программу для остальных функций из шестого столбца таблицы:

$y = 2 * x^2 - 2 * x - 2$,

$y = -2 * x^2 + 2 * x - 2$,

$y = 2 * x^2 + 2 * x - 2$,

$y = 2 * x^2 - 2 * x + 2$,

$y = -2 * x^2 + 2 * x + 2$.

Проведите сравнительный анализ, постарайтесь заметить закономерности.

* * *

И в заключение осталось рассмотреть последний вариант: $a = 0$, $b \neq 0$, $c \neq 0$.

Данный вариант образует уже известную нам линейную функцию.

Создайте программу в файле **kv7.py** для графика функции $y = 2 * x + 2$.

* * *

Мы рассмотрели всевозможные варианты квадратичных функций, провели небольшой анализ и теперь знаем, за что отвечает каждый из коэффициентов. Как вы заметили, все представленные функции — это частный случай общей записи квадратичной функции $y = a * x^2 + b * x + c$.

Для закрепления материала в книге представлено несколько проектных заданий, рекомендуем их пройти.

2.3.1. Продвинутый уровень

Модуль построения квадратичной функции

Из вышесказанного следует, что с помощью общей записи функции мы можем получить все остальные. Напишем программу, в которой вы спроектируете функцию, способную строить графики любой квадратичной функции.

Создадим файл **kvad.py**, в котором опишем функцию (метод) с помощью оператора **def**.

Более подробно о создании программных функций написано в книге «*Python*. Великое программирование в *Minecraft*». Этот метод будет получать данные о координатах игрока, строить систему координат и график заданной функции.

Первоначально добавим два основных элемента: координаты игрока и систему координат.

```
import mcpi.minecraft as minecraft
import mcpi.block as block
import math

craft = minecraft.Minecraft.create()

def kv(a, b, c):
    cor = craft.player.getTilePos()

    x = cor.x
    y = cor.y
    z = cor.z

    for j in range(100):
        craft.setBlock(x-50+j, y, z, 35, 15)
        craft.setBlock(x, y-50+j, z, 35, 15)
```

Метод назван **kv** и имеет параметры **a**, **b** и **c**, которые будут отвечать за значение коэффициентов квадратичной функции.

Так как квадратичное уравнение имеет вид $y = a * x^2 + b * x + c$, а погрешность построения в игре составляет 1 блок, то наши графики очень «разряжены». Чтобы этого избежать, воспользуемся масштабным коэффициентом в 0,1 для ***x***.

Добавим код для создания графика произвольной функции.

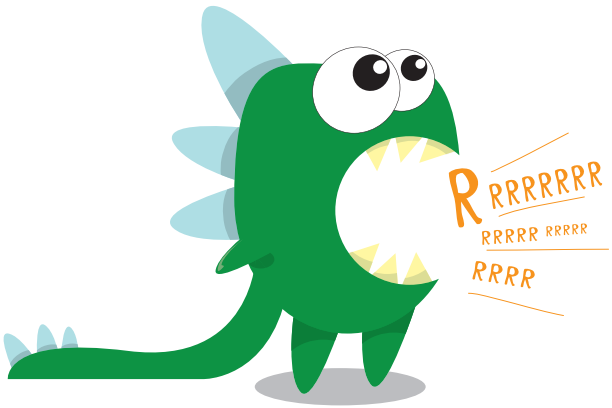
```
for i in range(-50, 55):
    y1 = a*((0.1*i)**2)+b*0.1*i+c
    craft.setBlock(x+i, y+y1, z, 35)
```

Наш модуль готов. Теперь протестируем его на примере создания графика произвольной квадратичной функции. Создадим файл **graf1.py** и запишем следующий код.

```
from kvad import *

# Создадим график функции y = a*x^2+b*x+c с применением
# kv(a, b, c)

kv(2, -2, 5)
```



В первой строчке мы импортируем все функции модуля **kvad.py**. Далее поясняем для всех пользователей нашей программы, что необходимо сделать. После этого вызываем функцию **kv ()** с тремя произвольными значениями чисел. Результат работы программы представлен на рис. 50.

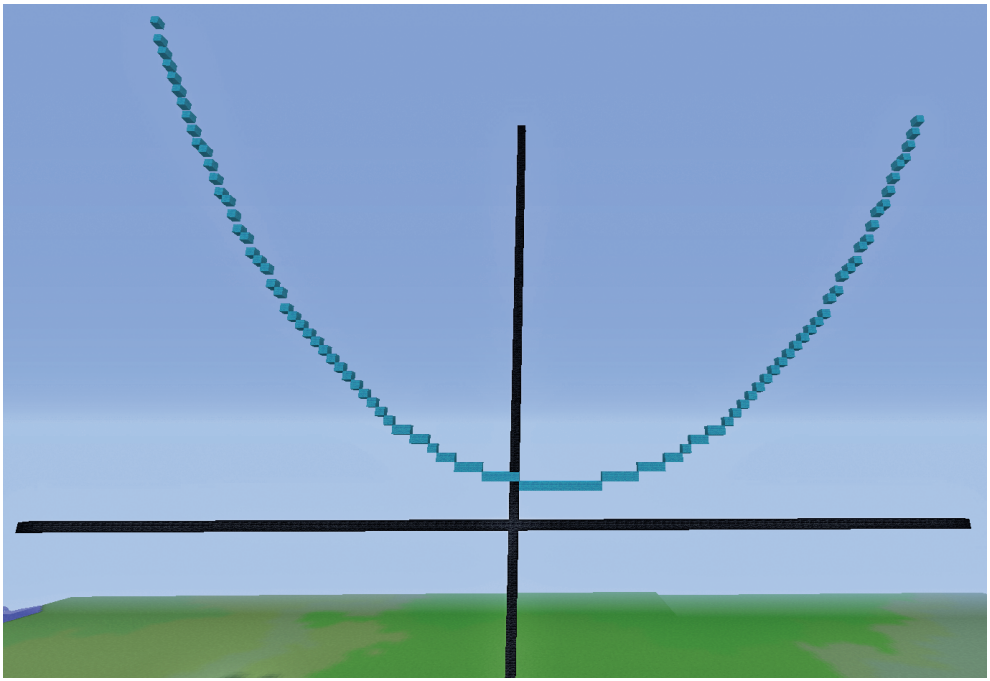


Рис. 50. Построение графика произвольной квадратичной функции с помощью своего модуля

Усложним наш модуль и программу так, чтобы можно было создавать множество графиков, но при этом и задавать им разные цвета для отличия.

Для этого в модуле **kvad.py** для функции **kv()** добавим ещё один параметр — **color**.

Об авторах

Корягин Андрей Владимирович — выпускник физико-математического факультета Пензенского государственного педагогического университета им. В. Г. Белинского. Методист в области образовательной робототехники, обучения детей языкам программирования и 3D-моделирования. Автор научных статей в области применения информационных технологий в образовании.

Корягина Алиса Витальевна — выпускница кафедры математического моделирования математического факультета Воронежского государственного университета. Ведёт научную работу в области теории фракталов и динамического хаоса. Специалист в области математического моделирования фрактальных структур в среде *Apophysis*. Разработчик программной ветви *Apophysis AV "Phoenix Edition"*.