yorkshire4 Documentation

Release 0.1

Russell Keith-Magee

Oct 12, 2019

Contents

In 1982, my parents bought a Commodore 64 for our family. At first, I just played games - from a cartridge, or from a tape drive. But I soon understood that this machine could be *programmed*, which meant you could write your own games. But what game should I write?

Luckily, my local library had the answer, in the form of books with titles like "Computer Space Games" and "Write your own Adventure Programs". These books contained full code listings for simple games. Each game was illustrated with vibrant pictures that fired the imagination... which was a good thing, because the games themselves were incredibly simplistic, and usually text based.

But, the promise of a game where you blew up space aliens or slew a dragon was enough to convince you carefully transcribe the code, line by line... until you had a game that you had written all by yourself!!

And thus was set the direction for my entire career.

I'm not the only person who had this experience.. Many of today's tech professionals were inspired by the Usborne computing books they read as children. So much so, that Usborne has provided PDFs of those old programming books of my youth.

Today, I have children of my own. Instead of huddling over a Commodore 64 with it's "38911 BASIC BYTES FREE", they have phones, tablets and laptops, each of which have several orders of magnitude more computing power.

And, I'm a part of a worldwide computer language community - the Python community. Python is a marvellous language - simple and clear enough to be used as a teaching language, but powerful enough to be used to drive high traffic websites like Instagram, for scientific data analysis, for financial trading, and much more.

This project is my attempt to resurrect this period of my youth, and bring to my children (and others) the joy that I had as a 7 year old, writing games - but in the language that I now use as an adult.

CHAPTER 1

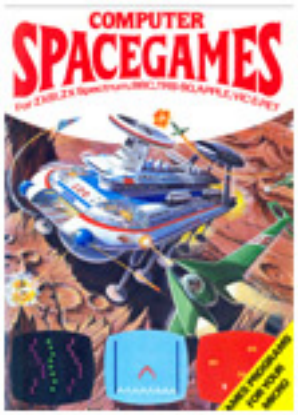
Getting Started

1. Install Python 3
2. Create a virtual environment
3. Start inputting the code!

CHAPTER 2

The Library

2.1 Computer games

- 

Computer Space Games
- 

Computer Battle Games

- 

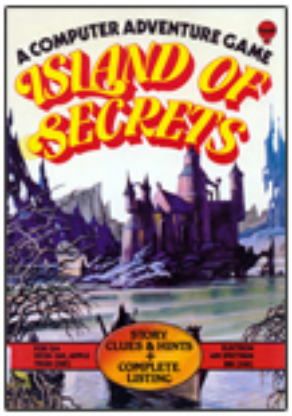
Computer Spy Games
- 

Weird Computer Games
- 

Creepy Computer Games

2.2 Adventure games

- 

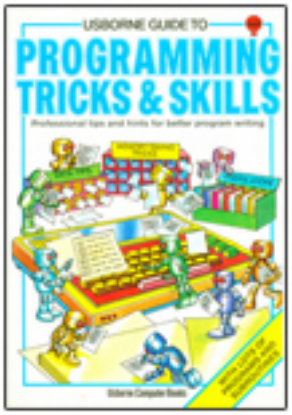
The Mystery of Silver Mountain
- 

Island of Secrets
- 

Write your own Fantasy Games

- *Write your own Adventure Programs*

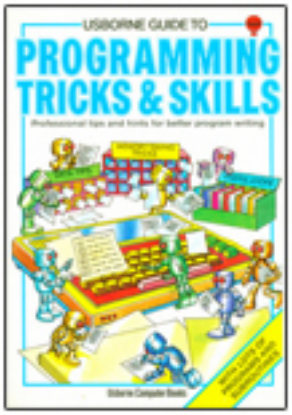
2.3 Introduction to Programming



- *Programming Tricks and Skills*

- *Machine Code for Beginners*

- 

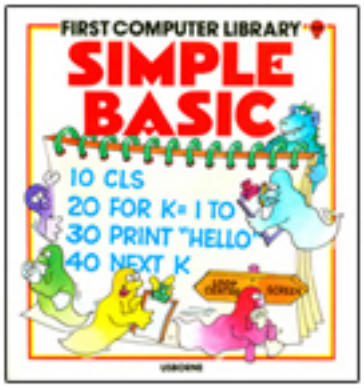
Computer Programming for Beginners
- 

Practical Things to do with a Microcomputer

2.4 First computer library

- 

Computer Fun

- 

Simple Basic

2.4.1 Computer Battle Games

Download the original book

Robot Missile

The year is 2582 and the people of Earth are in the midst of a battle against the Robots. A lethal Robot Missle has just landed and everyone is depending on you to find the secret code which unlocks its defuse mechanism. If you fail, the entire Earth Command Headquarters will be blown up.

Your computer knows what the code letter is. You must type in your guess and it will tell you whether the code letter is earlier or later in the alphabet. You have four chances to find the correct letter before the missile blows up.

The code

```
# Import `random` so we can generate random numbers.
# The `string` module gives us access to string-related utilities,
# like a list of lowercase characters.
import random
import string


print('ROBOT MISSILE')
print()
```

(continues on next page)

(continued from previous page)

```
print('Type the correct code letter (A-Z) to defuse the missile.')
print('You have 4 chances.')
print()

# Choose, at random, a lowercase ascii character
code = random.choice(string.ascii_lowercase)

# Loop until we get a match, or we've tried 4 times
guesses = 0
success = False

while not success and guesses < 4:
    # Read a character from the user, and convert it to lower case
    guess = input('What is your guess? ').lower()
    guesses += 1

    # Check to see if the guess is correct. If it is, flag the success;
    # otherwise, print an error message hinting towards the final answer.
    if guess == code:
        success = True
    elif guess < code:
        print('The code is later than %s in the alphabet' % guess)
    elif guess > code:
        print('The code is earlier than %s in the alphabet' % guess)

# The loop has exited. Let the user know if they succeeded or not.
if success:
    print('TICK... FIZZ... CLICK...')
    print('You did it!')
else:
    print('BOOOOOOOOMMM...')
    print('You blew it!')
    print()
    print("The correct code was '%s'." % code)
```

Adding to the program

You can make the computer print an extra message for a correct guess on the last go. Change line 37 to read:

```
print('You did it%s!' % (' (just)' if guesses == 4 else ''))
```

This inserts some text between "it" and the exclamation mark; the text that is inserted is dependent on the value of *guesses* at the end of the loop.

Puzzle corner

See if you can work out how to change the program to give you more or less changes of guessing the code letter.

The Vital Message

The code

```
# Import 'random' so we can generate random numbers.
# The 'string' module gives us access to string-related utilities
# The 'time' module helps us to wait for some time
import random
import string
import time


print('VITAL MESSAGE')

# Variable to store the difficulty i.e length of string
difficulty = 0

# Loop until the right difficulty level is enterred
while difficulty < 4 or difficulty > 10:
    print('How difficult do you want the game to be? (4-10)')
    x = int(input())
    difficulty = x

# Create the secret message
message = ''.join(
    random.choice(string.ascii_letters)
    for i in range(0, difficulty)
)

print('Send this message:')
print(message)

# Display the secret message in accordance with the difficulty level
time.sleep(0.5 * difficulty)

code = input('Enter the secret code that need to be transferred')

if code == message:
    print('MESSAGE CORRECT')
    print('The war is over!')
else:
    print('YOU GOT IT WRONG')
    print('You should have sent:', message)
```

Shootout

The code

```

```

Desert Tank Battle

The last major stronghold of Robot forces outside the U.R.S* is hidden in ancient castle ruins in the middle of the desert. A fleets of desert hovertanks has been sent to destroy it and you are the commander. Your tank controls the five remaining missiles.

You must assess carefully the direction and elevation before you launch each one. Your computer will ask you for a direction angle between -90° (extreme left) and +90° (extreme right) and an elevation angle between 0° (along the ground) and 90° (straight up). The elevation determine the distance the missile will travel.

Is your aim good enough to destroy the robot stronghold?

The code

```
# Import:
# * the `random` module so we can generate random numbers,
# * the `math` module so we can perform some basic trigonometry
import random
import math


# Normal terminals are defined a lot like typewriters. They are "line based" -
# they wait for the user to type a full line, then type Enter, and the entire
# line of input is processed.
#
# This is a utility method to get a whole number input from the keyboard which
# must be within a certain range. For example, a number between 1 and 10.
# For example:
#
#     my_guess = get_number('Please enter a guess between 1 and 10', 1, 10)
#
def get_number(prompt, min_value, max_value):
    # we don't know what value the user will type in yet, so start off
    # with "nothing" in the value
    the_number = None
    # get ready to keep asking the user for the number until they give us
    # an actual number that's within the range we need
    while the_number is None:
        # we can use the `input` function to get the user to type something
        # in and press [ENTER], and store it in the variable `user_input`,
        # so this line asks the user for the number with the `prompt` text
        user_input = input(prompt + ' ')
        try:
            # we have the user's input, so now we need to try to convert
            # it to a whole number
            the_number = int(user_input)
            # it's a number, so now we need to check the range
            if the_number > max_value or the_number < min_value:
                # the number is outside the allowed range
                # show the user a message to help them get it right next time
                if the_number > max_value:
                    print('That is more than {} - please try again.'.format(max_
↪value))
                elif the_number < min_value:
                    print('That is less than {} - please try again.'.format(min_
↪value))
                # discard the number we got because it's invalid
                the_number = None
        except (TypeError, ValueError):
            # whatever the user typed in, it wasn't a number
            # show the user a message to help them get it right next time
            print('That is not a number - please try again.')
```

(continues on next page)

(continued from previous page)

```
    # we have a number that's within the range we need - hand the result back
    return the_number


# Here we set up a bunch of things ready to play the game:
# Firstly - where is the enemy stronghold?
# This selects a whole number between -90 and 90 (inclusive) for the
# direction of the enemy stronghold
stronghold_direction = random.randint(-90, 90)
# This selects a fractional number between 0.0 and 1.0 for the
# distance of the enemy stronghold
stronghold_distance = random.random()

# Next, how many missiles does the player get to try to hit the enemy stronghold?
available_missiles = 5
# How many missiles have they used so far? They haven't started yet, so it's zero
used_missiles = 0
# Is the stronghold destroyed? They haven't started yet, so that is false
stronghold_is_destroyed = False

# Now we can start the actual game.
print()
print('Desert Tank Battle')
print()

# continue playing so long as the player has missiles left and the stronghold
# is not yet destroyed
while used_missiles < available_missiles and not stronghold_is_destroyed:
    # Get the guesses for direction and elevation - we store them in
    # the variables `aim_direction` and `aim_elevation_in_degrees`
    aim_direction = get_number('Direction (-90 to 90) ?', -90, 90)
    aim_elevation_in_degrees = get_number('Elevation (0 to 90) ?', 0, 90)

    # increase the count of used missiles
    used_missiles += 1

    # calculate the distance the missile travelled (the answer
    # will be between 0 and 1).
    # First, though, we need to convert the angle the player gave us
    # from degrees to radians so that we can use it in the `sin`
    # function to work out the distance. If you want to know more about
    # radians and degrees for measruring angles, you can read articles here:
    #   https://en.wikipedia.org/wiki/Degree_(angle)
    #   https://en.wikipedia.org/wiki/Radian
    aim_elevation_in_radians = math.radians(aim_elevation_in_degrees)
    # Now we can use the converted angle to work out the distance
    missile_distance = math.sin(aim_elevation_in_radians)

    # check how far away the missile landed from the actual target
    # first we check how many degrees off target the aim was - we use
    # the `abs` function to get the "absolute" difference between the
    # directions, disregarding whether it was left or right
    direction_difference = abs(stronghold_direction - aim_direction)
    # secondly we check how far off the distance was from the target distance
    # again, we use the `abs` function to get the "absolute" difference
    # between the distances, disregarding whether it was to short or too far
    distance_difference = abs(stronghold_distance - missile_distance)
```

(continues on next page)

(continued from previous page)

```
    # if the direction was with 2 degrees of the actual direction, and the
    # distance was within 0.05 of the actual distance, the target was hit
    good_direction = direction_difference <= 2
    good_distance = distance_difference <= 0.05

    if good_direction and good_distance:
        # record that the stronghold has been destroyed!
        stronghold_is_destroyed = True
    else:
        # the player missed the target - create a message to help them
        # aim better the next time around:
        message = 'The missile '

        # give information about the direction, if that was wrong:
        if not good_direction:
            if aim_direction < stronghold_direction:
                message += 'landed to the left'
            else:
                message += 'landed to the right'

        # if the direction and distance were both wrong, we need to
        # put an 'and' in to make the sentence make sense:
        if not good_direction and not good_distance:
            message += ' and '

        # give information about the distance, if that was wrong:
        if not good_distance:
            if missile_distance < stronghold_distance:
                message += 'did not travel far enough'
            else:
                message += 'travelled too far'

        # finish off the message nicely with a period for good punctuation:
        message += '.'

        # show the created message to help the player out:
        print(message)

if stronghold_is_destroyed:
    # the player hit the target - show a congratulatory message
    print()
    print('*KABOOOMMM*')
    print("You've done it")
    print()
else:
    # the player missed the target with all their missiles
    print()
    print('Disaster - you failed.')
    print('Retreat in disgrace.')
    print()
```

Puzzle corner

Can you work out how to add the possibility of the robots seeing you and shooting at you before your five goes are up?

Robot Invaders

The code

Secret Weapon

The code

Escape!

The Robots have caught you, taken your weapons and locked you up. Suddenly you remember you still have your sonar wristwatch, which can be tuned to produre sounds of any frequency. If you can only find the resonant frequency of your Robot guards, they should vibrate so much they fall apart.

You must be careful not to use frequencies that are too low or the building will vibrate and collapse on top of you. If you go too high, you will get such a headache you will have to give up.

Can you escape the horrors of the Robot prison? (Look carefully at the program for a clue to the range of frequencies to try.)

The code

```
# Import:
# * random so we can generate random numbers,
# * sys to manipulate the terminal
import sys, random

# Includes some yorkshire4 terminal utilities, too
from yorkshire4.terminal import clear_screen


clear_screen()

# This chooses a number between 1 and 100 for the frequency of the Robots.
frequency = random.randint(1,100)

# These are used if you go too low or too high.
low = False
high = False

won = False
lost = False

# This is the beginning of a loop hich allows you to have 5 turns.
for turn in range(5):
    guess = int(input('What is your guess? '))
```

(continues on next page)

(continued from previous page)

```
    # Checks if your guess is within 5 of the frequency.
    # If it is, it sets the won variable to True.
    if abs(frequency - guess) < 5:
        won = True
        break
    # Checks if your guess is so low it is less than the frequency by more than 40.
    # If it is, check that a low guess hasn't already been made, if it hasn't, print␣
↪a warning.
    # If it isn't, the program tells you that you have lost.
    elif frequency - guess > 40:
        if not low:
            print('Too low... the building rumbles, careful...')
            low = True
        else:
            print('The building collapsed!')
            lost = True
            break
    # Checks if your guess is so high it is more than the frequency by more than 40.
    # If it is, check that a high guess hasn't already been made, if it hasn't, print␣
↪a warning.
    # If it isn't, the program tells you that you have lost.
    elif guess - frequency > 40:
        if not high:
            print('Too high... ouch!')
            high = True
        else:
            print('Your head aches - you have to give up.')
            lost = True
            break
    else:
        print('No visible effect.')

# Checks what happened in the loop, if the game was won, print YOU'VE DONE IT.
# If not, check that there weren't too many low or high guesses before printing a␣
↪message.
if won:
    print('YOU\'VE DONE IT!')
elif not lost:
    print('You took too long!')
    print('The frequency was %d.' % frequency)
```

Puzzle corner

The three Robrt guards each have their own resonant frequency. You can't escape until you have found all three. How could you change the prograw to do this?

How to make the game harder

Change the number in the first "if" statement from 5 to a lower number. This means you have te get closer to the frequency to win. You can also increase the possible range of the frequency variable by changing the 100 to a higher number.

Pirate Dogfight

The code

```
```

Supersonic Bomber

The code

```
```

Iceberg

The code

```
```

The Wall

The code

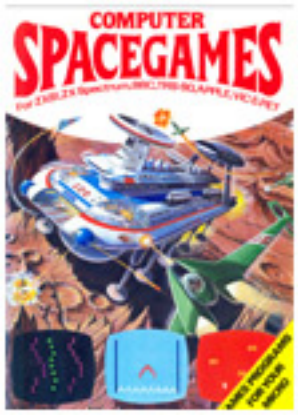
```
```

Missile!

The code

```
```

2.4.2 Computer Space Games



Download the original book

Starship Takeoff

You are a starship captain. You have crashed your ship on a strange planet and must take off again quickly in the alien ship you have captured. The ship's computer tells you the gravity on the planet. You must guess the force required for a successful take off. If you guess too low, the ship will not lift off the ground. If you guess too high, the ship's fail-safe mechanism comes into operation to prevent it from being burnt up. If you are still on the planet after ten tries, the aliens will capture you.

The code

```
# Import the 'random' module, so you can generate random numbers
import random


print('Starship Take-off')
print('-----------------')
print()

# Select 2 random numbers. `random.randrange(X)` picks a number
# between 0 and X, *including* 0, but *not* including X. Adding
# 1 means we get a number between 1 and X, *inclusive*.
gravity = random.randrange(20) + 1
weight = random.randrange(40) + 1

# Compute the required launch force.
required = gravity * weight

# Print the gravity for the planet
print('Gravity =', gravity)

# Start a loop that will run until either there have been
# 10 attempts, or the user entered the correct force.
success = False
c = 0
while c < 10 and not success:
    # On each loop, increment the count of attempts,
    # and prompt the user for a value
    c = c + 1
    force = int(input("Force:"))

    # Compare the value entered with the required value.
    # If they match, set the success flag; otherwise,
    # print an error
    if force > required:
        print('Too high!')
    elif force < required:
        print('Too low!')
    else:
        success = True

    # As long as this isn't the last loop,
    # we can try again
    if c < 10:
```

(continues on next page)

(continued from previous page)

```
        print('Try again.')

print()

# We've either been successful, or we've done 10 loops.
# Tell the user if they've been successful.
if success:
    print('Good take off!!')
else:
    print('You failed - the aliens got you')
```

How to make the game harder

You can change to the program to give you less than 10 goes. Do this by altering the number on lines 24 and 42 (They must be the same).

Puzzle corner

You could change the range of possible forces. See if you can work out how.

Intergalactic Games

There's fierce competition among the world's TV companies for exclusive coverage of the First Intergalactic Games. Everything depends on which company wins the race to put satellite into orbit at the right height. You are the engineer in charge of the launch for New Century TV. The crucial decisions about the angle and speed of the launching rocket rests on your shoulders. Can you do it?

The code

```
# import randint, so we can get a random integer
# import atan and pi, so we can calculate the correct angle
# import sqrt, so we can calculate the correct speed
from random import randint
from math import atan, pi
from math import sqrt


# Set up the number of guesses we're going to get
guesses = 8

# Create a flag to keep track if we've won
won_game = False

print("INTERGALACTIC GAMES")

# Chooses the height to which you must launch your satellite,
# puts the value in height, and prints it.
height = randint(1, 100)
print("You must launch a satellite to a height of", height)
```

(continues on next page)

(continued from previous page)

```
# As long as we have guesses left and haven't won yet, we'll keep trying
while guesses:
    guesses -= 1

    # Asks you for an angle and puts it in angle
    angle = int(input("Enter an angle (0-90): "))

    # Asks you for a speed and puts it in speed
    speed = int(input("Enter a speed (0-40000): "))

    # Subtracts the correct angle from your guess to determine
    # how close you were and stores the value in angle_diff
    angle_diff = angle - (atan(height / 3) * 180 / pi)

    # Subtracts the correct speed from your guess to determine
    # how close you were and stores the value in speed_diff
    speed_diff = speed - (3000 * sqrt(height + 1 / height))

    # Checks if you were close enough to win
    if abs(angle_diff) < 2 and abs(speed_diff) < 100:
        won_game = True
        break

    # Prints appropriate messages to help you with your next guess
    if angle_diff < -2:
        print("Too shallow.")
    elif angle_diff > 2:
        print("Too steep.")

    if speed_diff < -100:
        print("Too slow.")
    elif speed_diff > 100:
        print("Too fast.")
    print()

if won_game:
    print("You've done it!")
    print("NCTV WINS--thanks to you!")
else:
    print("You've failed.")
    print("You're fired!")
```

Adding to the program

These changes will make the computer give you bonus points depending on how quickly you make a successful launch.

After line 41, insert:

```
if guesses:
    bonus = 1000 * guesses
    print("You've earned a bonus of", bonus, "credits")
```

Puzzle corner

Can you change the program so that, if you win, it automatically goes back for another game, adding the bonus you've already earned to any new bonus? See how long you can play before and NCTV fires you.

Evil Alien

Somewhere beneath you, in deepest, blackest space, works Elron, the Evil Alien. You've managed to deactivate all but his short-range weapons, but he can still make his ship invisible. You know he's somewhere within the three-dimensional grid you can see on your ship screen, but where?

You have four space bombs. Each one can be exploded at a position in the grid specified by three numbers between 0 and 9. Can you blast the Evil Elron out of space before he creeps up and captures you?

The code

```
# Import:
# * the `random` module so we can generate random numbers,
from random import randint

print("EVIL ALIEN")

# set the size of the grid
size = 10

# set the size of the goes
goes = 4

# Create a flag to keep track if we've won
won_game = False

# Elron's position is fixed by these lines,
# which select 3 numbers between 0 and the size of the grid
x = randint(0, size)
y = randint(0, size)
d = randint(0, size)

# start of a loop which tells the computer to repeat the
# next lines G times
for i in range(goes):

    # this section asks you for your 3 numbers and stores them in x1, y1 and d1
    x1 = int(input("X position (0 to 9): "))
    y1 = int(input("Y position (0 to 9): "))
    d1 = int(input("D position (0 to 9): "))

    # checks if you were right and jumps to 300 if you were
    if x== x1 and y == y1 and d == d1:
        won_game = True
        break

    # your guesses are compared with Elron's position and a report printed
    print('SHOT WAS ')
    if y1 > y:
        print('NORTH')
```

(continues on next page)

(continued from previous page)

```
    if y1 < y:
        print('SOUTH')
    if x1 > x:
        print('EAST')
    if x1 < x:
        print('WEST')

    print()

    if d1 > x:
        print('TOO FAR')
    if d1 < x:
        print('NOT FAR ENOUGH')
    # end of loop. returns for another go until 'goes' remain.

# prints if you've used up all your goes
if not won_game:
    print('YOUR TIME HAS RUN OUT!!')
else:
    print('*BOOM* YOU GOT HIM!')
```

How to make the game harder

This program has been written so that you can easily change the difficulty by changing the size of the grid. To do this, put a different value for *s* in line 10.

If you increase the grid size you will need more space bombs to give you a fair chance of blasting Elron. Do this by changing the value of *goes* in line 14.

Puzzle corner

Can you work out how to change the program so that the computer asks you for a difficulty number which it can out into *size* instead of *size* being fixed? (Tip: limit the value of *size* to between 6 and 30 and use *int(size//3)* for the value of *goes* in line 14.)

Beat the Bug Eyes

You're trapped! Everywhere you turn you catch a glimpse of the steely cold light of a space bug's eyes before it slithers down behind a rock again. Slowly the bugs edge towards you, hemming you in, waiting for a chance to bind you in their sticky web-like extrusions. Luckily you have your proton blaster with you.

The bug eyes pop up in four different places on your screen and these correspond to keys 1 to 4. Press the correct key while the bug's eyes are on the screen and you will blast it. There are 10 bugs in all - the more you blast, the greater your chance of escape.

The code

```
# Import:
# * the `random` module so we can generate random numbers,
# * `time` so we can sleep
```

(continues on next page)

(continued from previous page)

```
import random
import time

# Includes some yorkshire4 terminal utilities, too
from yorkshire4.terminal import hide_cursor, show_cursor, clear_screen


# Start by hiding the cursor
hide_cursor()

score = 0

# Do 10 iterations of showing bug eyes
for t in range(10):
    clear_screen()

    # random.random() returns a floating point number in the range 0 -> 1
    # Add 0.5 to get a number between 0.5 and 1.5. That's how long we'll
    # sleep before displaying some eyes.
    time.sleep(random.random() + 0.5)

    # Pick a random zone in which to display the eyes.
    zone = random.randrange(4) + 1

    # The zone gives us the x,y offset
    if zone == 1:
        x_offset = 0
        y_offset = 0
    elif zone == 2:
        x_offset = 38
        y_offset = 0
    elif zone == 3:
        x_offset = 0
        y_offset = 12
    elif zone == 4:
        x_offset = 38
        y_offset = 12

    # Combine the offsets and a random number to postion the cursor
    # and print eyes. We can just print newlines to set the y (vertical)
    # position, then print spaces to set the x (horizontal) position.
    for y in range(random.randrange(12) + y_offset):
        print()

    print(' ' * (random.randrange(38) + x_offset) + '* *')

    # Wait up to a second for the player to respond.
    try:
        selected = int(getch(timeout=1))
    except TypeError:
        # If the user doesn't press a character, or they press something
        # other than a number, the call to int() will raise a ValueError.
        # Regardless, we know they didn't pick a zone; set the zone to None
        selected = None

    if selected == zone:
        score = score + 1
```

(continues on next page)

(continued from previous page)

```
# Clear the screen, and re-show the cursor.
clear_screen()
show_cursor()

# Last of all, print the user's score.
print('You blasted %s bugs' % score)
```

How to change the speed

You can speed the game up by changing the length of time that you sleep on line 90, or the timeout on line 119.

How to use more of the screen

This program was written to fit on an 80x25 screen - the standard size for a Unix terminal. Can you work out what you need to change to handle a larger or smaller screen?

Puzzle corner

Can you change the program to make the bugs appear in more than four places on the screen? Can you add more bugs too?

Moonlander

The code

```
```

Monsters of Galacticon

The code

```
```

Alien Snipers

The code

```
```

Asteroid Belt

The code

```
```

Trip into the Future

The code

```
```

Death Valley

The code

```
```

Space Mines

The code

```
```

Space Rescue

The code

```
```

Touchdown

The code

```
```

2.4.3 Computer Spy Games

Download the original book

Spy Eyes

The code

```
```

Searchlight

The code

```
```

Robospy

The code

```
```

Spy Q Test

The code

```
```

Secret Message Maker

The code

```
```

Rendezvous

The code

```
```

Morse Coder

The code

```
```

2.4.4 Creepy Computer Games

Download the original book

Computer Nightmare

The code

```
# Import 'random' so we can generate random numbers.
import random

# Initialy the score is 300
score = 300

# List of comments for making the game a more fun!
comments = [
    '** MICROS RULE! **',
```

(continues on next page)

(continued from previous page)

```
    '*PEOPLE ARE SUPID*',
    '+A ROBOT FOR PRESIDENT!',
    '!COMPUTERS ARE GREAT!',
    'I AM BETTER THAN YOU!',
]

# Print a heading
print('  Number   Score')
print('~~~~~~~~~~~~~~~~~')

# Loop while the score is not above 500 i.e you have won,
# or less than 0 means you have lost the game
while score > 0 and score < 500:
    # Number that will appear on screen
    number = random.randint(1, 9)
    print('     %4d     %4d' % (number, score))

    if random.random() > 0.5:
        print(comments[int(score/100)])
    else:
        if score < 60:
            print('THERE IS NO HOPE')
        elif score > 440:
            print('URK! HELP!!')

    answer = int(input('Enter your guess:'))
    # If you have guessed right increase the score
    if answer == number:
        score = score + number * 2
    else:
        score = score - 10

if score < 0:
    print('YOU ARE NOW MY SLAVE')
else:
    print('OK... you win... (this time)')
```

The Number Wizard

The code

```
```

Ghost Guzzler

The code

```
```

Spiderwoman

The code

```
```

Gravedigger

The code

```
```

Mad House

The code

```
```

Ghost Maze

The code

```
```

Seance

The code

```
```

2.4.5 Weird Computer Games

Download the original book

Tower of Terror

The code

```
```

Skulls of The Pyramid

The code

```
```

Monster Wrestling

The code

```
```

Jaws

The code

```
```

Flying Witches

The code

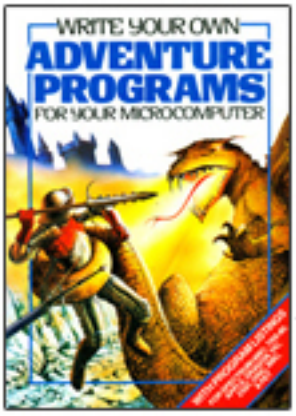
```
```

Micropuzzle

The code

```
```

2.4.6 Write Your Own Adventure Programs

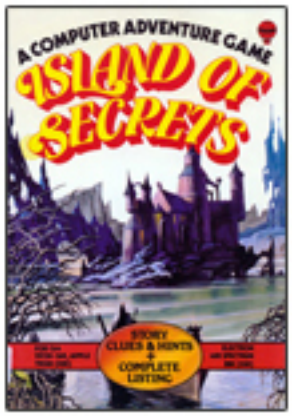


Download the original book

2.4.7 Write Your Own Fantasy Games

Download the original book

2.4.8 Island of Secrets



Download the original book

2.4.9 The Mystery of Silver Mountain

Download the original book

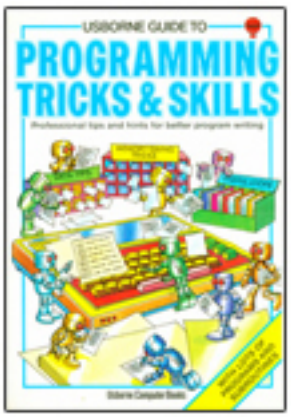
2.4.10 Computer Programming for Beginners

Download the original book

2.4.11 Machine Code for Beginners

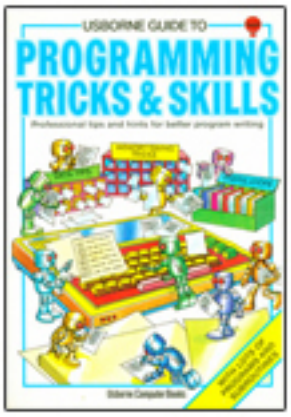
Download the original book

2.4.12 Practical Things to do with a Microcomputer



Download the original book

2.4.13 Programming Tricks and Skills

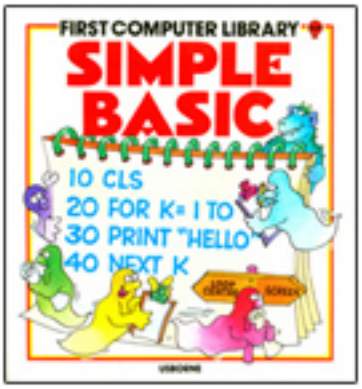


Download the original book

2.4.14 Computer Fun

Download the original book

2.4.15 Simple BASIC



Download the original book

CHAPTER 3

Becoming a Yorkshireman (or woman!)

> If you want to go fast, go alone. If you want to go far, go together.
>
> —African proverb

This is a big project, and I need plenty of help!

3.1 Code of Conduct

3.1.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

3.1.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks

- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

3.1.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

3.1.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

3.1.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at russell@keith-magee.com. All complaints will be reviewed and investigated and will result in a response that is deemed necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

3.1.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of- conduct.html.

3.2 Your development environment

All code for the Yorkshire4 project is contained in our Github repository, so your first step is to fork this repository, and clone a copy to your local computer.

1. Install Python 3
2. Create and active a virtual environment
3. Install sphinx:

   ```
   pip install sphinx
   ```

4. In the *docs* directory, run *make*
5. Open *docs/_build/index.hml* and check the results!

Then, work out how you can best contribute!

3.3 Contributing code

Do you know how to code? There's a lot of code that needs to be converted. Each of the listings needs to be converted from BASIC to Python.

3.3.1 Guidelines for converting BASIC to Python

When converting code, keep in mind that this is an educational activity, so your best coding habits should be on display. Resist trying to be overly "clever". Pick really clear variable names. Generally display the best coding practice you can.

It's not *just* about the code, though. If you look at the original books, there's two tracks on any code example - the code itself, and a bunch of annotations describing what the code is doing. These comments are just as important!

Lastly, keep in mind that this code was written over 35 years ago - so it's not exactly going to be Pythonic as-is. The original code is the starting point, not the end. You should be updating the code to be idiomatic, modern code, not just a literal transcription.

3.4 Contributing words

Are you a tech writer or copy editor? The annotations and flavour text explaining each code example need to be transcribed and updated.

3.5 Contributing art

Are you an artist? The original art from the books is under copyright, and cannot be used as-is. However, the art was an important part of stimulating my youthful enthusiasm - so it would be amazing to have new art to accompany the old text!

CHAPTER 4

Colophon

4.1 Why Yorkshire?

Monty Python's "Four Yorkshiremen" sketch is perhaps the best embodiment of the spirit of this project:

That is - a bunch of (ahem… old) people, sitting around, pining for "the good old days", lamenting how kids these days don't know how good they've got it.