Build an AI bedtime storyteller with Stable Diffusion

Tell Bedtime Stories who's the hero and where it happens, and the program creates an illustrated tale for you, using the magic of artificial intelligence

MAKER

Sean McManus

Author of *Mission Python*, *Web Design in Easy Steps*, and *Raspberry Pi For Dummies* (with Mike Cook). Get free chapters at Sean's website.

sean.co.uk

You'll Need

- ChatGPT account and API key **platform.openai.com**
- Pimoroni HyperPixel Square (optional) **magpi.cc/hyperpixel4**
- External speakers (optional)

Once upon a time there were two artificial intelligences, who were famed throughout the land. ChatGPT was an inventive storyteller, while Stable Diffusion was an ace illustrator. Together, they made up children's stories, which Raspberry Pi read aloud while showing the pictures. Listeners could put themselves, or their friends, in the stories, and decide where they should take place. Sometimes the stories were strange, and the pictures even more so. But as an art experiment, Bedtime Stories was a revelation. Anyone spying on the code could discover how to use ChatGPT and Stable Diffusion to create text and pictures from Python.

01 Get your DeepAI API key

DeepAI aims to make artificial intelligence (AI) accessible to new developers. The company provides an online text-to-image tool, accessed through an application programming interface (API). You send it a prompt (a request) and it sends you back an image. The API has been around since 2016 and its latest version uses the Stable Diffusion image generation software. Visit **deepai.org** to create your account. You can buy credit for 100 API calls for $5. On your Profile page, change the settings to allow unfiltered content. The filtered API rejected requests that included terms such as 'hand in hand' or 'creepy', resulting in incomplete stories.

02 Install Python modules

We'll need three Python modules for this project. OpenAI's ChatGPT is used to create the story text and the image prompts for DeepAI. An image prompt is a description of the image we'd like the AI software to make for us. The `requests` module is used to access Stable Diffusion through the DeepAI API. The `pyttsx3` module reads the stories aloud. Using speech in this project makes it feel like a friendly robot is reading you a bedtime story, but also means we can dedicate the screen to the story images. Open a terminal window and enter these instructions to install the Python modules and text-to-speech software. You may see some warnings about OpenAI not being on PATH, but you can safely ignore them.

```
pip install pyttsx3 requests openai
sudo apt install espeak mopidy
```

03 Create the ChatGPT prompt

This project has two parts: Bedtime Story Maker (**story_maker.py**), and Bedtime Story Reader (**story_reader.py**). Stories are created and saved by the Maker and loaded and read by the Reader. Take a look at **story_maker.py**. When you run the program, it asks you to enter the character names and types, location, and genre for the story. It then builds them into a string called `story_prompt` that is used to ask ChatGPT to make the story. The story prompt also asks for an image prompt for every paragraph, starting with 'Image Prompt:'. The story prompt tells ChatGPT to suggest a book illustrator for each picture.

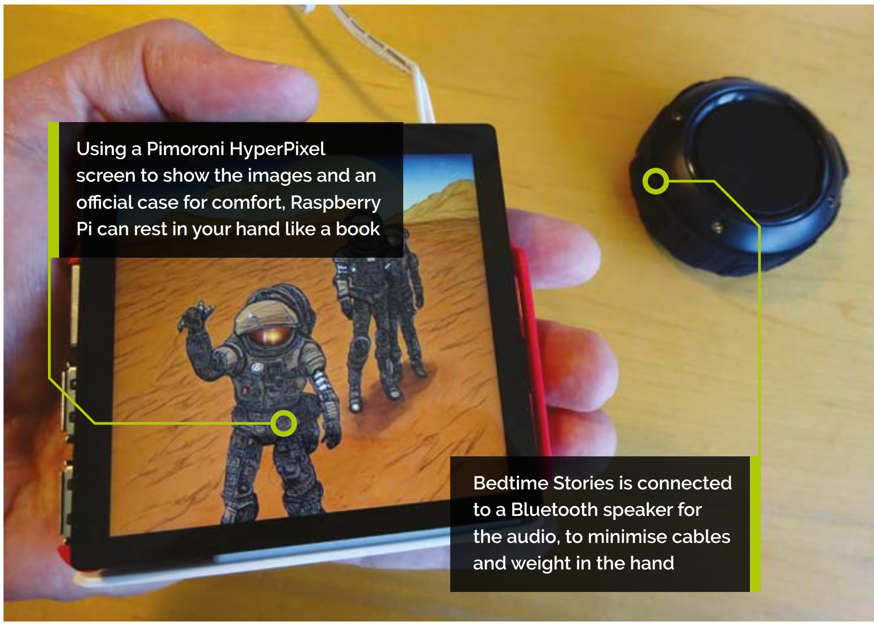
04 Using the ChatGPT API

Lines 33 to 39 send ChatGPT the story prompt. Add your own ChatGPT API key. We've told ChatGPT it is a children's author. Hints like this help to guide it to the best responses, based

> Saving the stories means you can play them on demand, without waiting for them to generate

on the vast amount of online content it's been exposed to. The response comes in as a single piece of text, which should alternate between story paragraphs and image prompts. Lines 42 to 46 split that response into a list of image prompts and story paragraphs. Sometimes ChatGPT fails to return any image prompts, so the `while` loop keeps asking ChatGPT for stories until it also sends some image prompts.

▲ ChatGPT came up with the idea for this female ghost with long, flowing hair and ethereal robes. Stable Diffusion made it real


Using a Pimoroni HyperPixel screen to show the images and an official case for comfort, Raspberry Pi can rest in your hand like a book
Bedtime Stories is connected to a Bluetooth speaker for the audio, to minimise cables and weight in the hand

05 Saving the story and image prompts

The story paragraphs and image prompts are saved to text files using the `write_list()` function. It accepts a list and a file name and saves each list item using the file name with a number on the end (e.g. **story-0.txt**, **story-1.txt**). They're saved in a folder named with the time, from the year to the second, such as 'Story 2023-04-10-09-34-52'. Saving the stories means you can play them on demand, without waiting for them to generate, as often as you like. It also enables you to study what worked well, and what didn't. ChatGPT wrote the first version of this function for us, by the way. It's good at basic Python programming, as long as you know how to guide it. We used the online front-end to ChatGPT for that at **chat.openai.com**.

06 Generate Stable Diffusion images

Next, the program goes through the list of image prompts and sends each one to DeepAI's API. You'll need to add your own DeepAI API key in line 59. To make the images more consistent, the program adds something to each prompt to clarify who the characters are and where the story

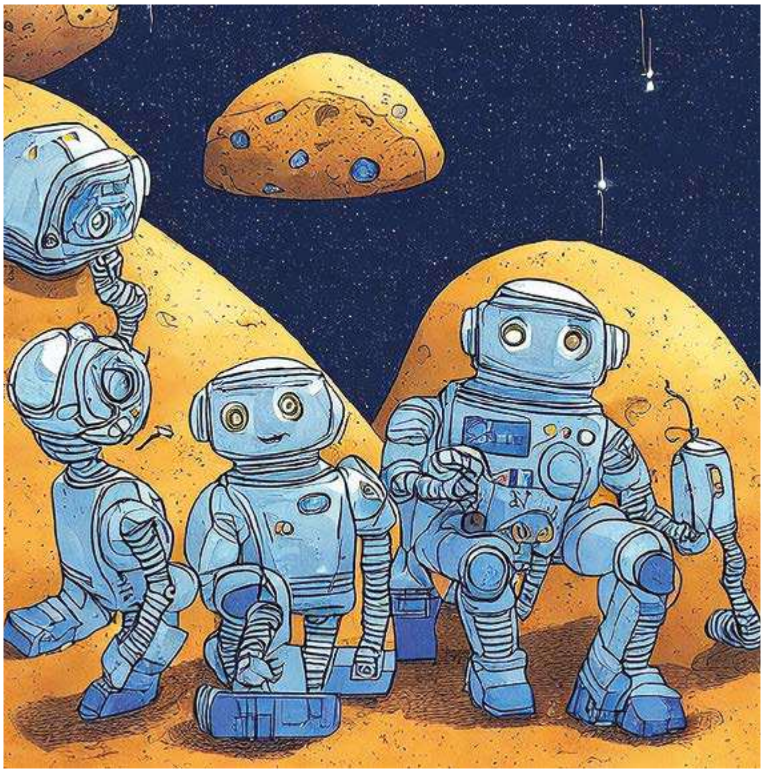
Top Tip

Watch your wallet

The DeepAI API requests are inexpensive, but if you make a lot of stories, it adds up fast.

Alert!
Strange images

Stable Diffusion may produce images that are unsettling for young children, such as distorted faces.



The AI image generation software does particularly well with sci-fi images

takes place. We've also added a negative prompt to provide some hints that can help improve image quality. We asked for images that are 720×720, to match the size of the HyperPixel display you are using. Valid values are between 128 and 1536, but you might get odd results below 256 and above about 700. DeepAI returns your image as a URL, so the program downloads the file and saves it for you. It also displays the URL, so you can download the image (you've paid for!) manually if something goes wrong. You can find a simple version of the code to generate and download an image at **magpi.cc/seandeepai**.

Top Tip

Flip the images

If your display is upside down, you can flip your images by inserting this before line 29 in Story Reader:

```
image_to_show
= pygame.
transform.
flip(image_
to_show, True,
True)
```

07 Do your quality control

Use the Run button in Thonny to run **story_maker.py** a few times to make some stories. When it finishes, the story folder should contain story files (e.g. **story-0.txt**, **story-1.txt**); image prompt files (**prompt-0.txt**, **prompt-1.txt**); and an image file for each prompt (**1.jpg**, **2.jpg**, etc.). The Reader reads you the story paragraph aloud while it shows you the related image, so there should be the same number of image and story files. Sometimes there's an extra story file because ChatGPT adds a title, and the Reader can handle that. Occasionally, ChatGPT doesn't send enough image prompts, so there aren't enough pictures. In that case, you can still read the story and enjoy the illustrations yourself, but the Reader won't work. Find a valid story folder you want to play and copy its folder name.

08 Set up Bedtime Story Reader

Now you've made some stories, let's use the Bedtime Story Reader (**story_reader.py**) to enjoy them. Add the name of the folder where your story is stored in line 3 of the program.

09 Discovering the story files

After it's set up the Pygame window for displaying images, and the voice for reading them, the program indexes the story and image files. Their file names go into the `story_files` and `image_files` lists, which are then sorted into the correct order. If there are more story files than image files, the `story_has_title` variable is set to True.

10 Displaying the picture

The program goes through the `story_files` list and displays the associated image for each story paragraph. If the story has a title, the folder will be one image file short because there is no image prompt for the title. In that case, the first image needs to be shown for both the title and the first paragraph. Line 24 ensures this happens.

It's common for characters in AI-generated images to have odd-shaped heads, distorted faces, and an unconventional number of limbs. It all adds to the surreal effect

story_maker.py

Language: **Python**

DOWNLOAD THE FULL CODE:

magpi.cc/bedtimestories

```python
# Story Maker - by Sean McManus - www.sean.co.uk
import os, openai, requests, datetime, random, time

def write_list(list_to_write, filename):
    for number, paragraph in enumerate(list_to_write):
        filepath = os.path.join(directory, f"{filename}-{number}.txt")
        with open(filepath, "w") as file:
            file.write(paragraph)

date_time_str = datetime.datetime.now().strftime("%Y-%m-%d-%H-%M-%S")
directory = f"Story {date_time_str}"
os.mkdir(directory)
image_prompts, story_paragraphs = [], []
character1_name = input("What is your main character's name? ")
character1_type = input("What kind of a character is that? ")
character2_name = input("What is your second character's name? ")
character2_type = input("And what kind of a character is that? ")
venue = input("Where does the story take place? (e.g. in a castle, on Mars) ")
genre = input("What is your story genre? ")
story_prompt = f"Please write me a short {genre}\
story. In this story, {character1_name} is a\
{character1_type} and {character2_name} is a\
{character2_type}. The story takes place {venue}.\
For each paragraph, write me an image prompt for\
an AI image generator. Each image prompt must\
start in a new paragraph and have the words\
'Image Prompt:' at the start. Choose a book\
illustrator and put something in the image\
prompts to say the images should be made in the\
style of that artist."

while len(image_prompts) == 0:
    print("Making ChatGPT request")
    openai.api_key = "YOUR API KEY" ### PUT YOUR API KEY HERE
    ChatGPT_output = openai.ChatCompletion.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a children's author."},
            {"role": "user", "content": story_prompt}
          ] )
     new_story = ChatGPT_output.choices[0].message["content"]
    print(new_story)
    for paragraph in new_story.split("\n\n"):
        if paragraph.startswith("Image Prompt"):
            image_prompts.append(paragraph)
        else:
            story_paragraphs.append(paragraph)
write_list(story_paragraphs, "story")
write_list(image_prompts, "prompt")

for number, image_prompt in enumerate(image_prompts):
    image_prompt += f"{character1_name} is {character1_type} and {character2_name} is {character2_type}. They are {venue}."
    print(f"Generating image {number}")
    r = requests.post(
    "https://api.deepai.org/api/text2img",
    data={'text': image_prompt,
          'negative_prompt': "poorly drawn face, mutation, deformed, distorted face, extra limbs, bad anatomy",
          'width': "720", 'height': "720",
          'grid_size': "1"},
    headers={'api-key': 'YOUR API KEY'} ### PUT YOUR API KEY HERE
    )
    image_url = (r.json()["output_url"])
    print(f"Image {number} is at {image_url}. Saving now...\n\n")
    filename = f"{number}.jpg"
    filepath = os.path.join(directory, filename)
    img_data = requests.get(image_url).content
    with open(filepath, 'wb') as handler:
        handler.write(img_data)
print(f"Your files are in {directory}.")
```

story_reader.py

> Language: **Python**

DOWNLOAD THE FULL CODE:

magpi.cc/bedtimestories

```
001.  # Story Reader - by Sean McManus - www.sean.co.uk
002.  # Put your story folder below
003.  path = "Story 2023-04-10-12-35-04"
004.  import os, pygame, pyttsx3
005.  win_width, win_height = 720, 720
006.  pygame.init()
007.  windowSurface = pygame.display.set_mode((
      win_width, win_height))
008.  pygame.mouse.set_visible(False)
009.  voice = pyttsx3.init()
010.  voice.setProperty('rate', 170)
011.
012.  story_files, image_files = [], []
013.  for file in os.listdir(path):
014.      if file.lower().endswith('.jpg'):
015.          image_files.append(file)
016.      elif file.lower().startswith('story'):
017.          story_files.append(file)
018.  story_has_title = len(story_files) > len (
      image_files)
019.  story_files = sorted(story_files)
020.  image_files = sorted(image_files)
021.
022.  for number, story in enumerate(story_files):
023.      if story_has_title:
024.          image_path = os.path.join(
      path, image_files[max(0, number - 1)])
025.      else:
026.          image_path = os.path.join(
      path, image_files[number])
027.      image_to_show = pygame.image.load(image_path)
028.      image_to_show = pygame.transform.scale(
      image_to_show, (win_width, win_height))
029.      windowSurface.blit(image_to_show, (0,0))
030.      pygame.display.update()
031.      story_path = os.path.join(path, story)
032.      with open(story_path, "r") as file:
033.          story = file.readlines()
034.      voice.say(story)
035.      voice.runAndWait()
036.  pygame.quit()
```

Here, Stable Diffusion has created a bright, colourful image that matches the look of a preschool book

It turns the list indices 0,1,2,3… into 0,0,1,2… and ensures we don't run out of images. If we had four story paragraphs (numbered 0 to 3) and three images (0 to 2), the last image and the last paragraph would coincide as they should. Images are scaled to fit the display. Although images are requested at 720×720, they come through at 768×768 because image sizes are accurate to the nearest multiple of 64.

11 Read out the story paragraph

Lines 32 to 35 open the text file for the story paragraph, read it in to the variable `story` and then use the text-to-speech module to read it aloud. For tips on customising the voice, see Raspberry Radio in issue 122 (**magpi.cc/122**).

12 Run the code

You can run Bedtime Story Reader using any display and in the desktop. Here we are using the HyperPixel Square display. To make the images fill the display, we boot to the command line and run the program using:

```
python story_reader.py
```

There is lots you can do to build on this project. You could add a menu for selecting stories, transitions between images or background music. The code shows you how to use AI to make text and images for any project, opening up a world of adventures.