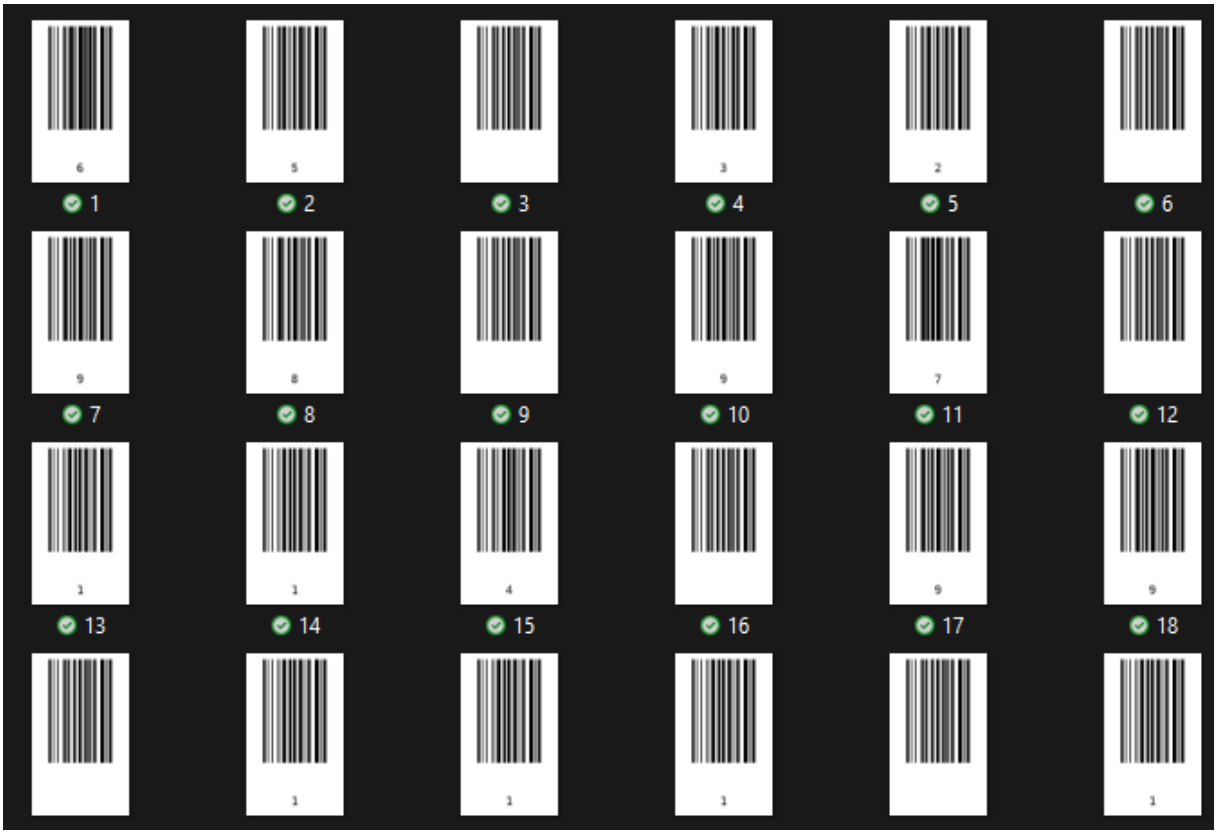
BLUE TEAM LABS ONLINE

CHALLENGE „BARCODE WORLD”

1. Challenge Overview

The challenge involves analyzing a large dataset containing 9,374 images, each featuring a barcode. The goal is to retrieve a hidden flag. Due to the volume of data, manual analysis is unfeasible, necessitating an automated approach using Python.

The content of the folder is:



Exactly 9374 photos, each one of them containing barcode image.

2. Methodology & Tools

To automate the barcode reading process, the following Python libraries were selected:

- Pyzbar (pyzbar.pyzbar): The core library for decoding barcodes.
- Pillow (PIL.Image): Used to load and manipulate images into a format compatible with Pyzbar.
- OS (os): Used for iterating through the file system and handling path joining (os.path.join) to ensure cross-platform compatibility.

3. Solution Development

To automate the extraction process, a Python script was developed. The script utilizes the pyzbar library for barcode decoding and Pillow for image processing.

Key Implementation Detail: Unlike standard file system iteration, which may order files alphanumerically (e.g., 1.png, 10.png, 2.png), the script forces a numerical iteration sequence. This ensures the decoded fragments are concatenated in the correct order to reconstruct the original message.

Script Logic:

1. Load the image using Pillow.
2. Decode the barcode using pyzbar.
3. Extract the data field.
4. Convert the raw data into a readable string.

Pyzbar Decode method returns a list of recognized barcodes on the image. Example would be:

```
print(decoded_list[1])

# Decoded(data=b'1923055034006', type='EAN13', rect=Rect(left=161, top=217, width=175, height=32), polygon=[Point(x=161, y=229), Point(x=239, y=249), Point(x=330, y=248), Point(x=332, y=242), Point(x=335, y=226), Point(x=336, y=220), Point(x=336, y=218), Point(x=248, y=217), Point(x=165, y=217), Point(x=164, y=219), Point(x=162, y=225)], quality=37, orientation='UP')
```

Data field is encoded to bytes format so decode('UTF-8') method will be used in code.

Os.path.join method will be used to join photos library system location and filename of the current analysed image. So example would be:

```
./photos_library + /filename.png
```

Os.path.join ensures secure path merging in a system-specific format.

PIL.Image method will only be used to change image.png file to pillow format acceptable by pyzbar library.

Code:

```
import os
import pyzbar.pyzbar as pyzbar
from PIL import Image

# Directory containing the barcode images
barcodes_folder = './Barcode_World'

# List for storing decoded fragments of the hidden message
hidden_message = []

# Iterate through files named 1.png ... 9374.png (sorted numerically)
for i in range(1, 9375):

    # Construct full path to the image file
    image_path = os.path.join(barcodes_folder, f'{i}.png')

    # Skip if the file does not exist
    if not os.path.exists(image_path):
        continue

    try:
        # Load the image using Pillow
        image_pillow_object = Image.open(image_path)

        # Attempt to decode any barcodes present in the image
        data_reading = pyzbar.decode(image_pillow_object)

        # If decoding returned data, extract the text content
        if data_reading:
            hidden_text = data_reading[0].data.decode('utf-8')

            # Debug output
            print(hidden_text)

            # Append decoded fragment to the message list
            hidden_message.append(hidden_text)

    except Exception as error:
        # Log any issues encountered during processing
        print(f"Error decoding {image_path}: {error}")
        continue

# Combine all decoded fragments into the final message
print("".join(hidden_message))
```

The output is:

```
65 32 98 97 114 99 111 100 101 32 40 97 108 115 111 32 115 112 101 108 108 101 100 32
98 97 114 32 99 111 100 101 41 32 105 115 32 97 32 109 101 116 104 111 100 32 111 102
32 114 101 112 114 101 115 101 110 116 105 110 103 32 100 97 116 97 32 105 110 32 97
32 118 105 115 117 97 108 44 32 109 97 99 104 105 110 101 45 114 101 97 100 97 98 108
101 32 102 111 114 109 46 32 73 110 105 116 105 97 108 108 121 44 32 98 97 114 99 111
100 101 115 32 114 101 112 114 101 115 101 110 116 101 100 32 100 97 116 97 32 98
121 32 118 97 114 121 105 110 103 32 116 104 101 32 119 105 100 116 104 115 32 97
110 100 32 115 112 97 99 105 110 103 115 32 111 102 32 112 97 114 97 108 108 101 108
32 108 105 110 101 115 46 32 84 104 101 115 101 32 98 97 114 99 111 100 101 115 44 32
110 111 119 32 99 111 109 109 111 110 108 121 32 114 101 102 101 114 114 101 100 32
116 111 32 97 115 32 108 105 110 101 97 114 32 111 114 32 111 110 101 45 100 105 109
101 110 115 105 111 110 97 108 32 40 49 68 41 44 32 99 97 110 32 98 101 32 115 99 97
110 110 101 100 32 98 121 32 115 112 101 99 105 97 108 32 111 112 116 105 99 97 108
32 115 99 97 110 110 101 114 115 44 32 99 97 108 108 101 100 32 98 97 114 99 111 100
101 32 114 101 97 100 101 114 115 46 32 76 97 116 101 114 44 32 116 119 111 45 100
105 109 101 110 115 105 111 110 97 108 32 40 50 68 41 32 118 97 114 105 97 110 116
115 32 119 101 114 101 32 100 101 118 101 108 111 112 101 100 44 32 117 115 105 110
103 32 114 101 99 116 97 110 103 108 101 115 44 32 100 111 116 115 44 32 104 101 120
97 103 111 110 115 32 97 110 100 32 111 116 104 101 114 32 103 101 111 109 101 116
114 105 99 32 112 97 116 116 101 114 110 115 44 32 99 97 108 108 101 100 32 109 97
116 114 105 120 32 99 111 100 101 115 32 111 114 32 50 68 32 98 97 114 99 111 100 101
115 44 32 97 108 116 104 111 117 103 104 32 116 104 101 121 32 100 111 32 110 111
116 32 117 115 101 32 98 97 114 115 32 97 115 32 115 117 99 104 46 32 50 68 32 98 97
114 99 111 100 101 115 32 99 97 110 32 98 101 32 114 101 97 100 32 111 114 32 100 101
99 111 110 115 116 114 117 99 116 101 100 32 117 115 105 110 103 32 97 112 112 108
105 99 97 116 105 111 110 32 115 111 102 116 119 97 114 101 32 111 110 32 109 111 98
105 108 101 32 100 101 118 105 99 101 115 32 119 105 116 104 32 105 110 98 117 105
108 116 32 99 97 109 101 114 97 115 44 32 115 117 99 104 32 97 115 32 115 109 97 114
116 112 104 111 110 101 115 46 32 84 104 101 32 98 97 114 99 111 100 101 32 119 97
115 32 105 110 118 101 110 116 101 100 32 98 121 32 78 111 114 109 97 110 32 74 111
115 101 112 104 32 87 111 111 100 108 97 110 100 32 97 110 100 32 66 101 114 110 97
114 100 32 83 105 108 118 101 114 32 97 110 100 32 112 97 116 101 110 116 101 100 32
105 110 32 116 104 101 32 85 83 32 105 110 32 49 57 53 49 32 40 85 83 32 80 97 116 101
110 116 32 50 44 54 49 50 44 57 57 52 41 46 32 84 104 101 32 105 110 118 101 110 116
105 111 110 32 119 97 115 32 98 97 115 101 100 32 111 110 32 77 111 114 115 101 32 99
111 100 101 91 49 93 32 116 104 97 116 32 119 97 115 32 101 120 116 101 110 100 101
100 32 116 111 32 116 104 105 110 32 97 110 100 32 116 104 105 99 107 32 98 97 114
115 46 32 72 111 119 101 118 101 114 44 32 105 116 32 116 111 111 107 32 111 118 101
114 32 116 119 101 110 116 121 32 121 101 97 114 115 32 98 101 102 111 114 101 32
116 104 105 115 32 105 110 118 101 110 116 105 111 110 32 98 101 99 97 109 101 32 99
111 109 109 101 114 99 105 97 108 108 121 32 115 117 99 99 101 115 115 102 117 108
46 32 65 110 32 101 97 114 108 121 32 117 115 101 32 111 102 32 111 110 101 32 116
121 112 101 32 111 102 32 98 97 114 99 111 100 101 32 105 110 32 97 110 32 105 110
100 117 115 116 114 105 97 108 32 99 111 110 116 101 120 116 32 119 97 115 32 115
112 111 110 115 111 114 101 100 32 98 121 32 116 104 101 32 65 115 115 111 99 105 97
116 105 111 110 32 111 102 32 65 109 101 114 105 99 97 110 32 82 97 105 108 114 111
97 100 115 32 105 110 32 116 104 101 32 108 97 116 101 32 49 57 54 48 115 46 32 68 101
118 101 108 111 112 101 100 32 98 121 32 71 101 110 101 114 97 108 32 84 101 108 101
```

112 104 111 110 101 32 97 110 100 32 69 108 101 99 116 114 111 110 105 99 115 32 40
71 84 69 41 32 97 110 100 32 99 97 108 108 101 100 32 75 97 114 84 114 97 107 32 65 67
73 32 40 65 117 116 111 109 97 116 105 99 32 67 97 114 32 73 100 101 110 116 105 102
105 99 97 116 105 111 110 41 44 32 116 104 105 115 32 115 99 104 101 109 101 32 105
110 118 111 108 118 101 100 32 112 108 97 99 105 110 103 32 99 111 108 111 114 101
100 32 115 116 114 105 112 101 115 32 105 110 32 118 97 114 105 111 117 115 32 99
111 109 98 105 110 97 116 105 111 110 115 32 111 110 32 115 116 101 101 108 32 112
108 97 116 101 115 32 119 104 105 99 104 32 119 101 114 101 32 97 102 102 105 120
101 100 32 116 111 32 116 104 101 32 115 105 100 101 115 32 111 102 32 114 97 105
108 114 111 97 100 32 114 111 108 108 105 110 103 32 115 116 111 99 107 46 32 84 119
111 32 112 108 97 116 101 115 32 119 101 114 101 32 117 115 101 100 32 112 101 114
32 99 97 114 44 32 111 110 101 32 111 110 32 101 97 99 104 32 115 105 100 101 44 32
119 105 116 104 32 116 104 101 32 97 114 114 97 110 103 101 109 101 110 116 32 111
102 32 116 104 101 32 99 111 108 111 114 101 100 32 115 116 114 105 112 101 115 32
101 110 99 111 100 105 110 103 32 105 110 102 111 114 109 97 116 105 111 110 32 115
117 99 104 32 97 115 32 111 119 110 101 114 115 104 105 112 44 32 116 121 112 101 32
111 102 32 101 113 117 105 112 109 101 110 116 44 32 97 110 100 32 105 100 101 110
116 105 102 105 99 97 116 105 111 110 32 110 117 109 98 101 114 46 91 50 93 32 84 104
101 32 112 108 97 116 101 115 32 119 101 114 101 32 114 101 97 100 32 98 121 32 97 32
116 114 97 99 107 115 105 100 101 32 115 99 97 110 110 101 114 44 32 108 111 99 97
116 101 100 32 102 111 114 32 105 110 115 116 97 110 99 101 44 32 97 116 32 116 104
101 32 101 110 116 114 97 110 99 101 32 116 111 32 97 32 99 108 97 115 115 105 102
105 99 97 116 105 111 110 32 121 97 114 100 44 32 119 104 105 108 101 32 116 104 101
32 99 97 114 32 119 97 115 32 109 111 118 105 110 103 32 112 97 115 116 46 91 51 93 32
84 104 105 115 32 105 115 32 116 104 101 32 102 108 97 103 32 45 32 66 52 114 99 48
100 51 95 72 49 53 55 48 114 89 46 32 84 104 101 32 112 114 111 106 101 99 116 32 119
97 115 32 97 98 97 110 100 111 110 101 100 32 97 102 116 101 114 32 97 98 111 117 116
32 116 101 110 32 121 101 97 114 115 32 98 101 99 97 117 115 101 32 116 104 101 32
115 121 115 116 101 109 32 112 114 111 118 101 100 32 117 110 114 101 108 105 97 98
108 101 32 97 102 116 101 114 32 108 111 110 103 45 116 101 114 109 32 117 115 101
46 91 50 93 32 66 97 114 99 111 100 101 115 32 98 101 99 97 109 101 32 99 111 109 109
101 114 99 105 97 108 108 121 32 115 117 99 99 101 115 115 102 117 108 32 119 104
101 110 32 116 104 101 121 32 119 101 114 101 32 117 115 101 100 32 116 111 32 97
117 116 111 109 97 116 101 32 115 117 112 101 114 109 97 114 107 101 116 32 99 104
101 99 107 111 117 116 32 115 121 115 116 101 109 115 44 32 97 32 116 97 115 107 32
102 111 114 32 119 104 105 99 104 32 116 104 101 121 32 104 97 118 101 32 98 101 99
111 109 101 32 97 108 109 111 115 116 32 117 110 105 118 101 114 115 97 108 46 32 84
104 101 105 114 32 117 115 101 32 104 97 115 32 115 112 114 101 97 100 32 116 111 32
109 97 110 121 32 111 116 104 101 114 32 116 97 115 107 115 32 116 104 97 116 32 97
114 101 32 103 101 110 101 114 105 99 97 108 108 121 32 114 101 102 101 114 114 101
100 32 116 111 32 97 115 32 97 117 116 111 109 97 116 105 99 32 105 100 101 110 116
105 102 105 99 97 116 105 111 110 32 97 110 100 32 100 97 116 97 32 99 97 112 116 117
114 101 32 40 65 73 68 67 41 46 32 84 104 101 32 118 101 114 121 32 102 105 114 115
116 32 115 99 97 110 110 105 110 103 32 111 102 32 116 104 101 32 110 111 119 45 117
98 105 113 117 105 116 111 117 115 32 85 110 105 118 101 114 115 97 108 32 80 114
111 100 117 99 116 32 67 111 100 101 32 40 85 80 67 41 32 98 97 114 99 111 100 101 32
119 97 115 32 111 110 32 97 32 112 97 99 107 32 111 102 32 87 114 105 103 108 101 121
32 67 111 109 112 97 110 121 32 99 104 101 119 105 110 103 32 103 117 109 32 105 110
32 74 117 110 101 32 49 57 55 52 46 91 52 93 32 81 82 32 99 111 100 101 115 44 32 97 32
115 112 101 99 105 102 105 99 32 116 121 112 101 32 111 102 32 50 68 32 98 97 114 99
111 100 101 44 32 104 97 118 101 32 114 101 99 101 110 116 108 121 32 98 101 99 111
109 101 32 118 101 114 121 32 112 111 112 117 108 97 114 46 91 53 93 32 79 116 104

```
101 114 32 115 121 115 116 101 109 115 32 104 97 118 101 32 109 97 100 101 32 105
110 114 111 97 100 115 32 105 110 32 116 104 101 32 65 73 68 67 32 109 97 114 107 101
116 44 32 98 117 116 32 116 104 101 32 115 105 109 112 108 105 99 105 116 121 44 32
117 110 105 118 101 114 115 97 108 105 116 121 32 97 110 100 32 108 111 119 32 99
111 115 116 32 111 102 32 98 97 114 99 111 100 101 115 32 104 97 115 32 108 105 109
105 116 101 100 32 116 104 101 32 114 111 108 101 32 111 102 32 116 104 101 115 101
32 111 116 104 101 114 32 115 121 115 116 101 109 115 44 32 112 97 114 116 105 99
117 108 97 114 108 121 32 98 101 102 111 114 101 32 116 101 99 104 110 111 108 111
103 105 101 115 32 115 117 99 104 32 97 115 32 114 97 100 105 111 45 102 114 101 113
117 101 110 99 121 32 105 100 101 110 116 105 102 105 99 97 116 105 111 110 32 40 82
70 73 68 41 32 98 101 99 97 109 101 32 97 118 97 105 108 97 98 108 101 32 97 102 116
101 114 32 49 57 57 53 46
```

4. Analysis of the Output

The raw output from the barcodes resulted in a sequence of decimal ASCII values.

Converting these values (using code below) to text revealed a lengthy paragraph regarding the history of barcodes.

```
print(hidden_message)

full_string_of_numbers = "".join(hidden_message)

ascii_values = full_string_of_numbers.split(' ')

decoded_chars = []
for value in ascii_values:
  if value:
    decoded_chars.append(chr(int(val)))

final_message = "".join(decoded_chars)
print(final_message)
```

The final output is:

> A barcode (also spelled bar code) is a method of representing data in a visual, machine-readable form. Initially, barcodes represented data by varying the widths and spacings of parallel lines. These barcodes, now commonly referred to as linear or one-dimensional (1D), can be scanned by special optical scanners, called barcode readers. Later, two-dimensional (2D) variants were developed, using rectangles, dots, hexagons and other geometric patterns, called matrix codes or 2D barcodes, although they do not use bars as such. 2D barcodes can be read or deconstructed using application software on mobile devices with inbuilt cameras, such as smartphones. The barcode was invented by Norman Joseph Woodland and Bernard Silver and patented in the US in 1951 (US Patent 2,612,994). The invention was based on Morse code[1] that was extended to thin and thick bars. However, it took over twenty years before this invention became commercially successful. An early use of one type of barcode in an industrial context was sponsored by the Association of American Railroads in the late 1960s. Developed by General Telephone and Electronics (GTE) and called KarTrak ACI (Automatic Car Identification), this scheme involved placing colored stripes in various combinations on steel plates which were affixed to the sides of railroad rolling stock. Two plates were used per car, one on each side, with the arrangement of the colored stripes encoding information such as ownership, type of equipment, and identification number.[2] The plates were read by a trackside scanner, located for instance, at the entrance to a classification yard, while the car was moving past.[3] This is the flag - B4rc0d3_H1570rY. The project was abandoned after about ten years because the system proved unreliable after long-term use.[2] Barcodes became commercially successful when they were used to automate supermarket checkout systems, a task for which they have become almost universal. Their use has spread to many other tasks that are generically referred to as automatic identification and data capture (AIDC). The very first scanning of the now-ubiquitous Universal Product Code (UPC) barcode was on a pack of Wrigley Company chewing gum in June 1974.[4] QR codes, a specific type of 2D barcode, have recently become very popular.[5] Other systems have made inroads in the AIDC market, but the simplicity, universality and low cost of barcodes has limited the role of these other systems, particularly before technologies such as radio-frequency identification (RFID) became available after 1995.

And it contains string:

> This is the flag - B4rc0d3_H1570rY.

5. Challenge Logic Reconstruction

The raw output obtained from the solution script was not a direct plaintext stream but a sequence of isolated digits and spaces.

- The sequence 6, 5 corresponds to the decimal value 65, which is the ASCII code for 'A'.
- The sequence 3, 2 corresponds to 32, which is the ASCII code for Space.

Based on this pattern, it can be deduced that the challenge creator did not encode the text string directly. Instead, the following logic was applied:

1. **Text Input:** The source text was taken as input.
2. **ASCII Conversion:** Each character was converted to its ASCII integer representation (ord('A') → 65).
3. **Digit Splitting:** The integer string was iterated character by character (splitting 65 into 6 and 5).
4. **Separator Injection:** A space character was inserted after every full ASCII code to serve as a delimiter.
5. **Image Generation:** A unique Code128 barcode was generated for each individual digit and space.
6. **Sequential Naming:** The enumerate() function was likely used to save these thousands of barcodes sequentially (e.g., 1.png, 2.png), which mandated the numerical sorting method used in the solution script.

Reconstructed Generator Script: The following Python snippet demonstrates how the dataset was likely generated:

```
import barcode
import os

text = """A barcode (also spelled bar code) is a method of
representing data in a visual, machine-readable form. Initially,
barcodes represented data by varying the widths and spacings of
parallel lines. These barcodes, now commonly referred to as linear or
one-dimensional (1D), can be scanned by special optical scanners,
called barcode readers. Later, two-dimensional (2D) variants were
developed, using rectangles, dots, hexagons and other geometric
patterns, called matrix codes or 2D barcodes, although they do not
use bars as such. 2D barcodes can be read or deconstructed using
application software on mobile devices with inbuilt cameras, such as
smartphones. The barcode was invented by Norman Joseph Woodland and
Bernard Silver and patented in the US in 1951 (US Patent 2,612,994).
The invention was based on Morse code[1] that was extended to thin
and thick bars. However, it took over twenty years before this
invention became commercially successful. An early use of one type of
barcode in an industrial context was sponsored by the Association of
American Railroads in the late 1960s. Developed by General Telephone
and Electronics (GTE) and called KarTrak ACI (Automatic Car
Identification), this scheme involved placing colored stripes in
various combinations on steel plates which were affixed to the sides
of railroad rolling stock. Two plates were used per car, one on each
side, with the arrangement of the colored stripes encoding
information such as ownership, type of equipment, and identification
number.[2] The plates were read by a trackside scanner, located for
instance, at the entrance to a classification yard, while the car was
moving past.[3] This is the flag - B4rc0d3_H1570rY. The project was
abandoned after about ten years because the system proved unreliable
after long-term use.[2] Barcodes became commercially successful when
they were used to automate supermarket checkout systems, a task for
which they have become almost universal. Their use has spread to many
other tasks that are generically referred to as automatic
identification and data capture (AIDC). The very first scanning of
the now-ubiquitous Universal Product Code (UPC) barcode was on a pack
of Wrigley Company chewing gum in June 1974.[4] QR codes, a specific
type of 2D barcode, have recently become very popular.[5] Other
systems have made inroads in the AIDC market, but the simplicity,
universality and low cost of barcodes has limited the role of these
other systems, particularly before technologies such as radio-
frequency identification (RFID) became available after 1995."""

list_to_encode = []

output_folder = r'./Barcode_World2'

for character in text:
    for character_2 in str(ord(character)):
```

```
        list_to_encode.append(character_2)
    list_to_encode.append(str(' '))


code_class = barcode.get_barcode_class('code128')
for i, character in enumerate(list_to_encode):
    my_code = code_class(character,
writer=barcode.writer.ImageWriter())
    file_path = os.path.join(output_folder, f'{i+1}')
    my_code.save(file_path)
```