Learning Objectives

Learners will be able to...

- **Effectively utilize `openai.Image.create` to generate a response**
- **Successfully generate an image using DALL-E**
- **Analyze and apply the various parameters available in DALL-E API calls**

info

Make Sure You Know

You are familiar with Python.

Limitations

This is a gentle introduction. So there is a little bit of Python programming. The information might not be the most up to date as OpenAI releases new features.

Text to Image

Dall-E

OpenAI's **DALL·E** text to image is a system that can generate images from textual descriptions, using a deep learning-based approach. The system is trained on a large dataset of text-image pairs, and can generate new images that are similar to the ones in the training data. The system can generate images that are realistic and editable, and can even create images from scratch, without any training data.

DALL·E models can easily generate images from text, making it a great choice for those who want to create quick and easy visual content.

> The Images API provides three methods for interacting with images:
> 1. Creating images from scratch based on a text prompt
> 2. Creating edits of an existing image based on a new text prompt
> 3. Creating variations of an existing image

Generating Image

In order to generate an image using the DALL·E we can use the `openai.Image.create` call. Here is an example below:

```
response = openai.Image.create(
  prompt="a dog smoking a cigar in a poker room.",
  n=1,
  size="256x256"
)
```

You can create an original image based on a text prompt using the image generations endpoint. Images can be sized at 256x256, 512x512, or 1024x1024 pixels, with smaller sizes generate faster. You can request 1-10 images at a time by using the n parameter. A 256x256 would generate faster than 1024x1024. For this course, we will tend to just you the 256 or the 512. API endpoints are the points where requests are made and responses are sent back. Let's print the content of the response to see what wasgenerated by the API call.

```
print(response)
```

When we printed the response we see there 3 distinct words `created`, `data` and inside `data` we see a URL. Here we want to focus on `URL`. The URL is the link to the image that the AI generated for us. Data represents all the info generated for the user and creation is pretty much an id. Now that we know what our response looks like we can trim down the response to what we actually want, which is the URL. Remove the `print(response)` line, and add the following.

```
image_url = response['data'][0]['url']
print(image_url)
```

We can open another tab in our browser, then we can copy and paste the URL to see the image that we have generated.

Try giving it a different size from the standard size mentioned above. For example, 255x255 or 256x1024.

It should generate an error, please revert back to a 256x256 or 512x512. We want it to be able to generate faster for future outputs. A Try it button is provided below to make sure we are no longer getting errors. It should just generate an URL here.

Saving our Image

Image Generation

In order to not have to keep a copy and pasting our result in, we are going to save the image inside the URL as file in our Codio box.

The first thing we are going to do is import the `requests` library. Using the request library we can get the image contents. We will use a handler to write in our new image file `image_name.jpg`. This code should go below all the code, we have already.

```
img_data = requests.get(image_url).content
with open('image_name.jpg', 'wb') as handler:
    handler.write(img_data)
```

We will be rewriting our file. When regenerating an image we will need to click on our refresh button to see the changes. Try using the buttons below with a few different prompts.

If feeling uninspired, try the following prompts:
* digital art of a white cat ninja.
* acrylic painting of a mountain.
* a chef shark making shrimp.
* digital art of white ninja cat with headband.

The more descriptive you can be, the closer the image will get to what you are expecting. For example, Here is what it generated for me for that last input.

image_name-copy

Writing descriptive OpenAI DALL·E prompts helps the model to better understand the context of the task, allowing it to create more accurate and creative responses. In other words, the more precise the explanation, the greater the probability of achieving the desired outcome for you or your end user.

N

When generating our prompt our code looks something like this:

```
response = openai.Image.create(
  prompt="a dog smoking a cigar in a poker room.",
  n=1,
  size="256x256"
)
image_url = response['data'][0]['url']
```

We have talked about the prompt and the size. The main thing we are calling and have not talked about yet is the `n`. You can request 1-10 images at a time using the n parameter. Try replacing `n` with 3.

As you can see from the image file we saved in our response, we got only one image. Let's try adding a print statement to visualize what we are getting for the URL.

```
image_url = response['data'][0]['url']
print(image_url)
```

Here we realize that we are going to only get one URL. We are going to change our code to view more of the response data.

```
image_url = response['data']
print(image_url)
```

Here we can see we got 3 URLs generated. What we are going to do is pull the different URL into separate variabls and generate files to save the URL. To make our life easier we should have the following:

```
image_url1 = response['data'][0]['url']
image_url2 = response['data'][1]['url']
image_url3  = response['data'][2]['url']
```

Please note you have have had `image_url = response['data'][0]['url']` from the following page,comment it out for now. if you want to check if the URLS look like what you have in mind,. You can run the following code:

```
print(image_url1[0:15])
print(image_url2[0:15])
print(image_url2[0:15])
```

We are slicing the content of our print statements here because we are generating URLs. The URLs tend to be in a longer format, because of that slicing might make the result more pleasing to the eye since we are just checking.

Saving Multiple Images

Now that we can get the URL for the 3 images we are going to save them in different files. The button below after the Try It! button are setups to open the following file names:
* image_name1.jpg
* image_name2.jpg
* image_name3.jpg

Now remember the technique we previously used to save our image. We are going to use that save the 3 images.

```
img_data = requests.get(image_url1).content
with open('image_name1.jpg', 'wb') as handler:
    handler.write(img_data)
img_data = requests.get(image_url2).content
with open('image_name2.jpg', 'wb') as handler:
    handler.write(img_data)
img_data = requests.get(image_url3).content
with open('image_name3.jpg', 'wb') as handler:
    handler.write(img_data)
```

If you click on every single links and you got a different image then you are on the right track. Since we are using Python let's start by cleaning up our code a bit.

For starters in order to better see the change that our code worked we are going to switch the prompt from ninja cat to ninja birds.

```
prompt="digital art of ninja bird "
```

Then remove everything after from when we mention `img_data = requests.get(image_url1).content`.

Now that we have standard style to name our images it makes it easy for us to use Python function like `enumerate` to loop through and create our files. All we have to do is simply, change the values from 1 to 2 to 3.

```
for i, image_url in enumerate([image_url1, image_url2,
        image_url3], start=1):
    img_data = requests.get(image_url).content
    with open(f'image_name{i}.jpg', 'wb') as handler:
        handler.write(img_data)
```

Just like that you cut more than half the lines of code you needed to store the different files. Using this code it should be pretty easy to replicate and save more than 3 pictures. Remember that `n` has a max value of 10.

Coding Exercise

Create a function named `imageGen` that takes for argument a prompt. The function should create `1` image with a size `256x256`. Save it in the file `my_image.jpg`.