Learning Objectives

Learners will be able to...

- **Effectively generate an image using DALL-E**
- **Create variations of DALL-E images using different parameters and techniques**
- **Analyze and apply the concepts of masks and edits in DALL-E image creation.**

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 Edits

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 can generate images that are realistic and editable, and can even create images from scratch, without any training data. Endpoints are often used to retrieve data from a database, create new data, or modify existing data. We have seen that DALL·E models can easily generate images from text. Now we are going to use DALL·E to modify existing data

---

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

We are familiar with the steps to generate a new image.

```
#this code generates a new image  from a prompt
response = openai.Image.create(
  prompt="digital art of throne room of a magical castle.",
  n=1,
  size="256x256"
)
#from the response generated this puts the url in a separate
        variable
image_url = response['data'][0]['url']

#using the url, we use the code below to save it as a file
img_data = requests.get(image_url).content
with open('image_name.png', 'wb') as handler:
    handler.write(img_data)
```

Variation

OpenAI has an image variation endpoint that allows you to generate similar images when given an image as a prompt.

To try it in the last Try It button we created a digital art representation of a castle as `image_name.jpg`. We are going to use that image to generate another similar picture or variation. To avoid rewriting our original file we are going to delete everything up until the comment that says `# Keep CODE ABOVE`. Then we are going to write, the code below to generate a variation.

```
response = openai.Image.create_variation(
  image=open("image_name.png", "rb"),
  n=1,
  size="1024x1024"
)
image_url = response['data'][0]['url']
#using the url, we use the code below to save it as a file
img_data = requests.get(image_url).content
with open('image_name_var.png', 'wb') as handler:
    handler.write(img_data)
```

Variation

Let's try creating an image variation of a bunny and a cat.

1. First we generate the original image of the bunny and cat.

```
#this code generates a new image  from a prompt
response = openai.Image.create(
  prompt="bunny and a cat",
  n=1,
  size="512x512"
)
#from the response generated this putS the url in a separate
        variable
image_url = response['data'][0]['url']
```

2. We save the bunny and cat image.

```
#using the url, we use the code below to save it as a file
img_data = requests.get(image_url).content
with open('bunny.png', 'wb') as handler:
    handler.write(img_data)
```

3. We use the new image to generate a variation of the said image using our new file name we created in the previous step

```
response = openai.Image.create_variation(
  image=open("bunny.png", "rb"),
  n=1,
  size="512x512"
)
image_url2= response['data'][0]['url']
```

4. Now we have to save our new URL in a different file name.

```
img_data = requests.get(image_url2).content
with open('bunny_var.png', 'wb') as handler:
    handler.write(img_data)
```

Prompts and images are filtered based on OpenAI's content policy, returning an error when a prompt or image is flagged.

mask

On top of a variation endpoint OpenAI's DALL·E has an **edit endpoint**. Similar, to our variation API call we are going to need to include a starting image for it to work on, a number`n`, and `size`. Unlike the variation endpoint we are going to add a `mask`, and a `prompt`.

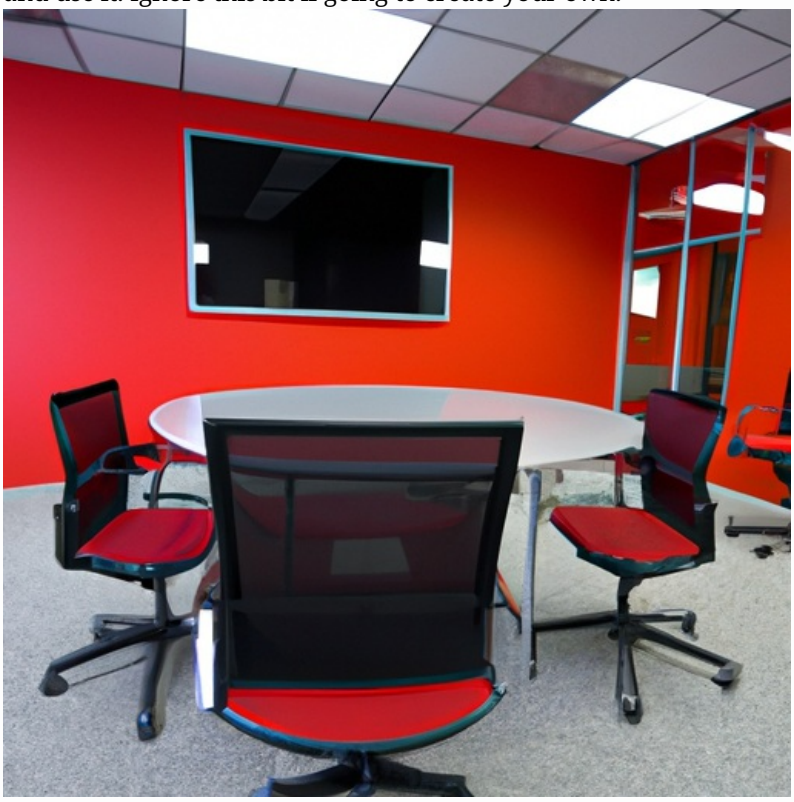
First let us generate the image we will be working with. For the sake of consistency we will be creating another from scratch.

```
#this code generates a new image  from a prompt
response = openai.Image.create(
  prompt="a vibrant modern office with red chairs and a
        television screen.",
  n=1,
  size="512x512"
)
#from the response generated this puts the url in a separate
        variable
image_url = response['data'][0]['url']
#using the url, we use the code below to save it as a file
img_data = requests.get(image_url).content
with open('original.png', 'wb') as handler:
    handler.write(img_data)

print(image_url)
```

The image generated when I first ran the code is provided below. Feel free to use if you do not want to interact with one by yourself. The image is saved under `original-copy.png`. Below is a code provided to help you copy

and use it. Ignore this bit if going to create your own.



```
import shutil

def copy_image(source_path, destination_path):
    try:
        shutil.copy2(source_path, destination_path)
        print("Image copied successfully!")
    except IOError as e:
        print(f"An error occurred while copying the image: {e}")

# Example usage
source_image = "original-copy.png"
destination_image = "original.png"
copy_image(source_image, destination_image)
```

After generating this bit we are going to delete/remove the code to generate an image and simply keep our libraries. We don't need to regenerate an original image.

Now we need to create our **mask**. The purpose of a mask is to define the region of an image that you want to change or apply an effect to, while leaving the rest of the image unchanged.

In image processing, a mask is typically a grayscale image with the same dimensions as the original image. The mask is used to selectively modify the original image by indicating which pixels should be affected by an operation and which pixels should be left unchanged.

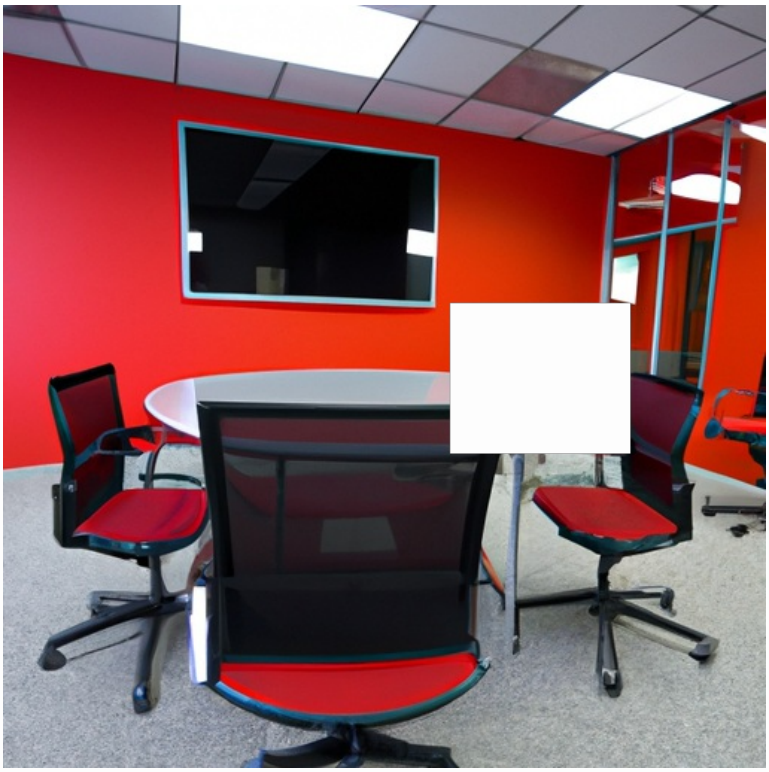
Now we are going to generate code to basically write over our image in order to create the mask. First, we are going to open our original image and convert it to `RGBA`. This step is important otherwise it would generate an error saying the following:`openai.error.InvalidRequestError: Invalid input image - format must be in ['RGBA', 'LA', 'L'], got RGB`. After generating and saving your file feel free to delete the previous code that was used for image generation.

```
## Load the original image
img = Image.open('original.png')
# Convert the image to RGBA format
img = img.convert('RGBA')
```

We are going to create a copy of our original image so we can put a mask over it.

```
# Create a new blank image for the mask
mask = img.copy()
```

Now for the fun part we simply gonna draw a rectangle box over where the edit should be. Here is an example:



the mask image represented the same room but this time with a filled in white square at the center of the image.

We know our picture is 512x512. Depending on your image change the location of the box so it makes sense so an edit can occur. Feel free to keep changing the value of `bbox` until you have a version that suits you.

The bbox variable is a tuple containing four values: (left, top, right, bottom).

- left is the x-coordinate of the left edge of the bounding box
- top is the y-coordinate of the top edge of the bounding box
- right is the x-coordinate of the right edge of the bounding box
- bottom is the y-coordinate of the bottom edge of the bounding box

Finally, we are also going to make sure we save our picture in a file called `masked.png`.

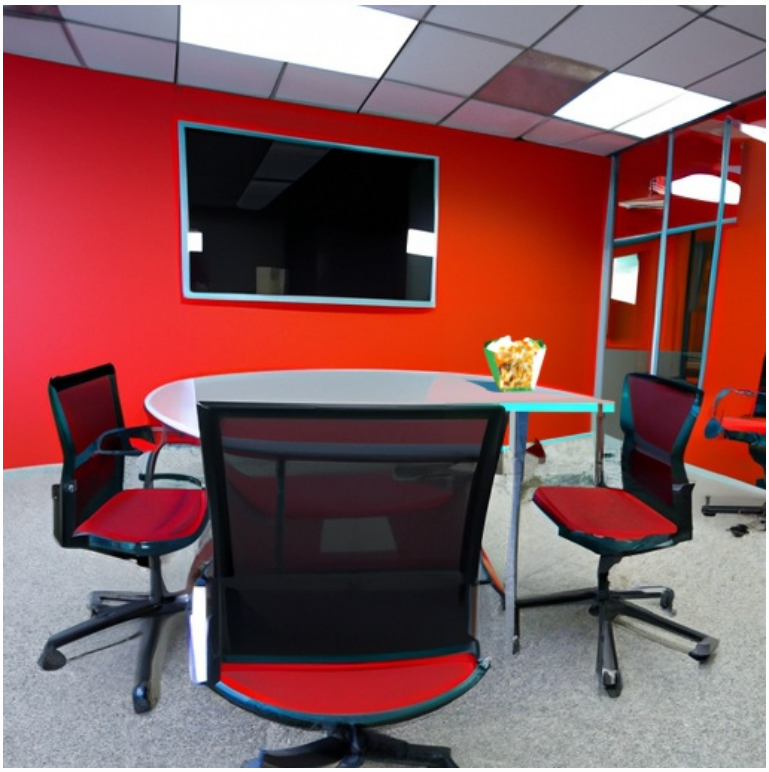
```
# Define the region you want to keep in white
bbox = (300,200, 420, 300)
mask_draw = ImageDraw.Draw(mask)
mask_draw.rectangle(bbox, fill=255)

# Save the mask
mask.save('masked.png')
```

Editing our Image

Now that we have our mask and original picture we can generate our edited picture. We are going to slightly modify our prompt to describe what we want generated.

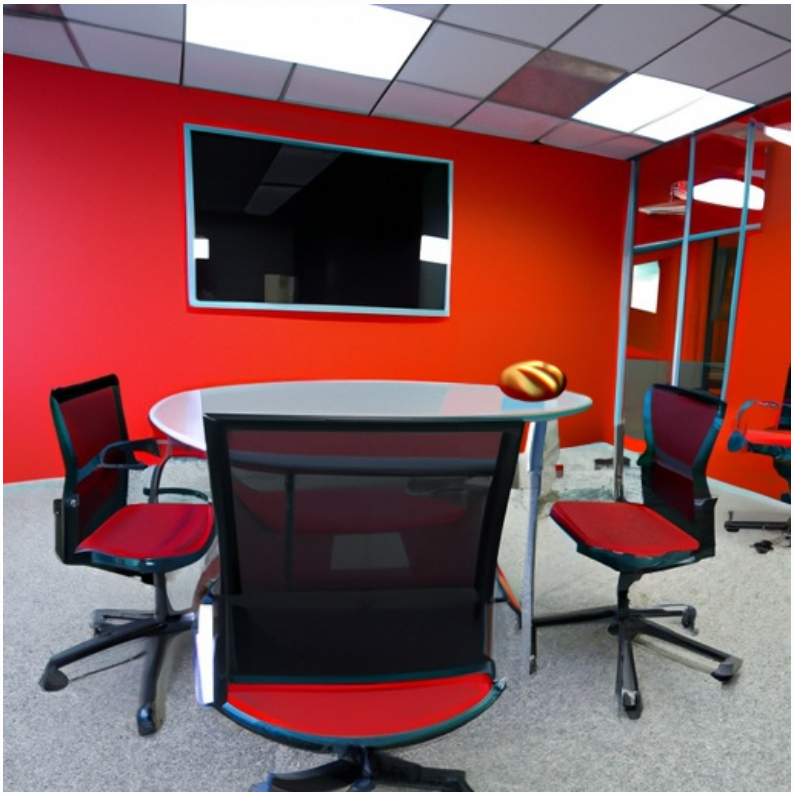
```
response=openai.Image.create_edit(
  image=open("original.png", "rb"),
  mask=open("masked.png", "rb"),
  prompt="a vibrant modern office with red chairs and a
        television screen.a snack is on the table. ",
  n=1,
  size="512x512"
)
image_url = response['data'][0]['url']
#using the url, we use the code below to save it as a file
img_data = requests.get(image_url).content
with open('edited.png', 'wb') as handler:
    handler.write(img_data)
```



this image shows the same image as the original and mask.However where the white box was we have a bucket of fries on the table.

We can also change the prompt to display other things such as:

```
response=openai.Image.create_edit(
  image=open("original.png", "rb"),
  mask=open("masked.png", "rb"),
  prompt="a vibrant modern office with red chairs and a
        television screen. A bagel is on the table. ",
  n=1,
  size="512x512")
```



The mask parameter is optional. If not provided our image still needs to have transparency, which will be used as the mask. Feel free to move the box associated with your mask, in order to make different edits with your picture.

Coding Exercise

Write a function `variationMaker()` that takes in a file name like `image.png`. The functions should create 1 variation of the image in `512x512` and save it under a file name called `varTest.png`.

Feel free to test it out before submission. A try it button will be provided, additionaly so will `varTest.png`file.