# Learning Objectives

---

### Learners will be able to...

- **Generate a response using `openai.Completion.create`**
- **Use API calls to help with Python code**
- **Run different prompts with OpenAI**
- **Build a movie recommendation program powered by OpenAI's GPT-3**

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.

# Program Using OpenAI

GPT-3 API is a powerful set of tools for natural language processing that can help developers create innovative applications. Now that we are more or less familiar with GPT-3's API we are going to create our first program using the API. This will serve two goals:
1. It will help to get some practice using Python and the OpenAI API
2. Practice our prompt generation

## Moflix

We are going to create a movie recommendation program powered by OpenAI's GPT-3. The program will be named `Moreflix`. Your Python file will be located in the top left , this is where you will put in your code. Lastly, you will be provided with the standard box(located in the bottom left) that we can use as our GPT playground.

For starters, let's try generating what we want. Use the playground to ask for the following prompt and then run our Try it! button.

```
Give me 5 movie recommendations
```

## UX

I know that typing the prompt into our playground is an easy solution, but why don't we take it a step further and create a program specifically designed for movie recommendations? This program will be tailored to give users a smoother movie searching experience, requiring less typing than GPT-3. We'll also focus on UX design to give our users an even better experience. This program will make it easier to find the perfect movie recommendation without having to put in the extra effort on our user.

**UX** stands for User Experience and refers to the overall experience a user has when interacting with a product or service. It encompasses aspects such as the design, usability, and functionality of an interface, and takes into account the user's needs, motivations, and expectations. A good UX should be enjoyable, intuitive, and provide a positive user experience.

# Part 1. Box Creation

---

We are going to create a Python function that asks the user for some information. We will use the information given by the user to create our program.

We will be interacting with user input we are going to use the terminal to interact with our program. The Try It button below will run your code inside the terminal .

First, let's create a function that takes a list of movies and put it in a nice box. As we don't have our movie list yet, we are going to create a sample list. Copy and paste the following inside our moflix.py

```
sample=["One life","Two","Potato Pie and Life"]
```

We are going to need to create a function that takes an argument which is our movie list. That function should return the movie list in a nice box.


Our box should change based on the number of movies and the length of the items that need to fit in the box. We are going to create our function called `InBox` that first tries to get the length of the biggest element in our list.

```
def InBox(x):
  #this gives you the longest string
  biggest = max(x, key = len)
  #THIS gets you the length of the biggest string
  biggest=len(biggest)
```

Now for the top of our box we are going to start and end with `+` . Then for the content in the middle we will fill it with `-`. This technique will be used for both our top and bottom.

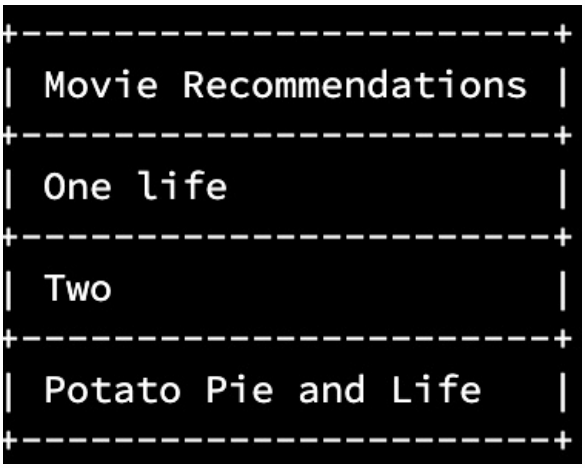
```
print("+" + "-" * (biggest + 2) + "+")
```

We want to focus on the middle part now. We will use a for loop to iterate through all our items. We'll start off with a `|` and then add our movie title. We'll then add additional whitespaces `" "` to match the length of the longest element.

```
for i in x:
    print("| " + i + (" "*(biggest-len(i))) + " |")
```

Then we will go outside our for loop and add the same code we used to create the top layer, in order to create our bottom layer.

In order to make our design look nicer, Let's try making our box look different.




Guidance

```
# print element of a list in nice box for the user
def InBox(x):
  x=(["Movie Recommendations"]+x)
  #this gives you the longest string
  biggest = max(x, key = len)
  #THIS GETS the len of the biggest string
  biggest=len(biggest)
  #creating our box
  print("+" + "-" * (biggest + 2) + "+")
  for i in x:
    print("| " + i + (" "*(biggest-len(i))) + " |")
    print("+" + "-" * (biggest + 2) + "+")
```

# Part 2. Prompt Creation

Now that we have our box we need to be able to fill it with a list of movies. The next step we are going to take is creating our prompt. We are going to create a function called Moreflix that will prompt the user for:
1. The number of recommendations they want
2. The genre they want
3. A similar movie they have in mind.

We write the code below to take care of that. The function will prompt users and return a prompt for us to feed to our OpenAI. We want the user to be able to type 0 if they don't have an answer in mind.

```
def Moreflix():
  # ask the user for different inputs
  print("Enter 0, if you don't have an answer in mind")
  number_recs=int(input('How many movie do you want recommended:
        '))
  genre=input('What genre are you looking for: ')
  similar= input('What is a similar movie: ')
```

From there we have all the tools needed for us to write our prompt. First of all if they add `0` to the number of recs we will default it to `5` for them.

```
if number_recs==0:
  number_recs=5
```

For the others if `0` is included we will generate a prompt simply asking for the number of recommendation. But first we are about to create our first prompt. A quick reminder,here are three basic guidelines to creating better results:
* **Show and tell**
* **Provide quality data**
* **Check your settings**
We are going to focus on show and tell where. For show and tell we want to focus on 3 things:
1. Giving the model clear instructions (tell)
2. Giving the model an example (show)
3. Giving the model clear instructions and an example (show and tell)

In order for the prompt to generate more clear answers, try the following prompt in the playground in the bottom left next to your terminal. In a python array form, give me 3 movie recommendation`

Now that we have a sample prompt and have an idea of what the response would look like, we can start coding. We are going to take care of edge cases where the users does not provide all the information. Base on the information that the user provides we will, create different prompts. For example here is an example where the user does not provide a genre or a similar movie. Copy and paste the code to the `moflix` file on your left.

```
if genre=="0" and similar=="0":
  return("In a python array form, give me" + str(number_recs)
        +"movie recommendation")
```

Now we can take care of the additional scenarios

```
if genre=="0" :
  return ("In a python array form, give me " + str(number_recs)
        +" movie recommendation, similar to " + similar)
if similar=="0":
  return ("In a python array form, give me "+genre + " "+
        str(number_recs) +" movie recommendation")
return ("In a python array form, give me "+genre +"
        "+str(number_recs)+" movie recommendation similar to " +
        similar)
```

Now run the following to make sure all the work is going well. To make our code cleaner, feel free to remove the print statements when done with it.
`print(Moreflix())`

To increase our accuracy, of getting the AI to do what we want after our prompt we will start it off with providing it space to do what we want
`Prompt=Moreflix() + "rec="`

# Part 3. Presenting Info

Now that we have our prompt let's take care of integrating our OpenAI API key as we have done before . Ideally you will move this to the top of our file.

```
import os
import openai
import secret
openai.api_key=secret.api_key
```

Let's create a quick function that takes a prompt and returns our movie results . This function should go between in our `InBox` and `Moflix` functions. We are going to call this function `Res` short for response. This function should work as all the previous prompts we generated using our API.

```
def Res(x):
  response = openai.Completion.create(
    model="text-davinci-002",
    prompt=x,
    top_p=1,
    max_tokens=100)
  return(response['choices'][0]['text'].strip())
```

We can add the following at the end of our code to check how `Res` behaves.

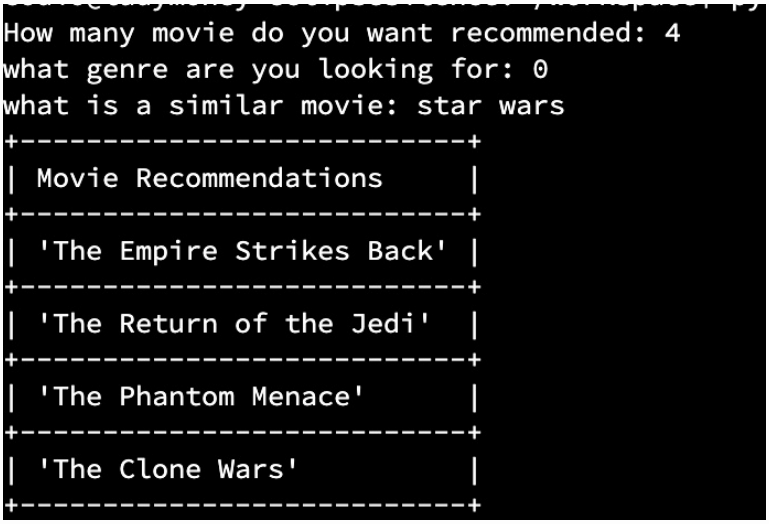
```
Prompts=Moreflix() + "rec ="
print(Res(prompts))
```

After checking that everything works as attended we can change our print to a value assignment. Set the `Res` function call to the variable `movies`. `movies=Res(prompts)`.

We are at the finish line. We know that the response generated by GPT-3 was a string. Even though, it was a string we want to convert it to an actual list. After that we can call InBox from our list.

```
# Converting string to list
import ast
movie_list=ast.literal_eval(movies)
InBox(movie_list)
```

Your program should be able to generate something like the image below.




rec