
DESENVOLVIMENTO
DE APIs E
MICROSSERVIÇOS

5

Uso da biblioteca Requests

Lucas Mendes Marques Gonçalves

Resumo

É possível baixar informações das mais diversas na internet. Mas nos interessa integrar essas informações em aplicações. Para isso, temos que automatizar o processo de acesso. É isso que começamos a aprender nesta unidade.

5.1. OMDB

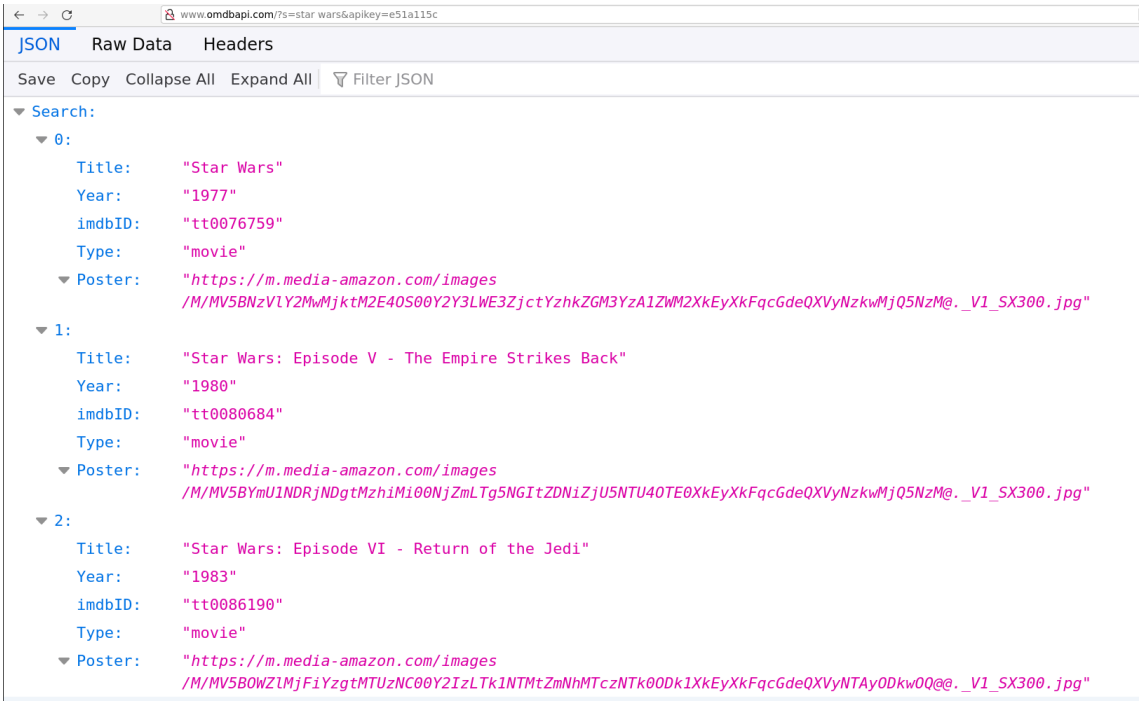
O OMDB será usado como exemplo nesta unidade. Trata-se de um serviço que permite baixar dados de filmes, gratuitamente e de forma automatizada.

Para conseguir acesso ao OMDB, você precisa de uma chave de API (api key). Uma chave de API é uma forma que diversos serviços implementam de controlar o consumo de cada um de seus usuários, para poder impor limites ou fazer cobranças de usuários mais frequentes. No nosso caso, conseguimos fazer uso gratuito da API, em até 1000 pedidos por dia.

Para obter uma chave de acesso, visite http://www.omdbapi.com/apikey.aspx e siga as instruções (há instruções mais detalhadas no vídeo, se precisar).

A API omdb vai nos fornecer basicamente dois serviços. O primeiro é a busca textual, na qual fornecemos uma string e recebemos uma lista (pouco detalhada) de filmes que se adequam à busca, como podemos ver na Figura 5.1.

Figura 5.1. Busca textual

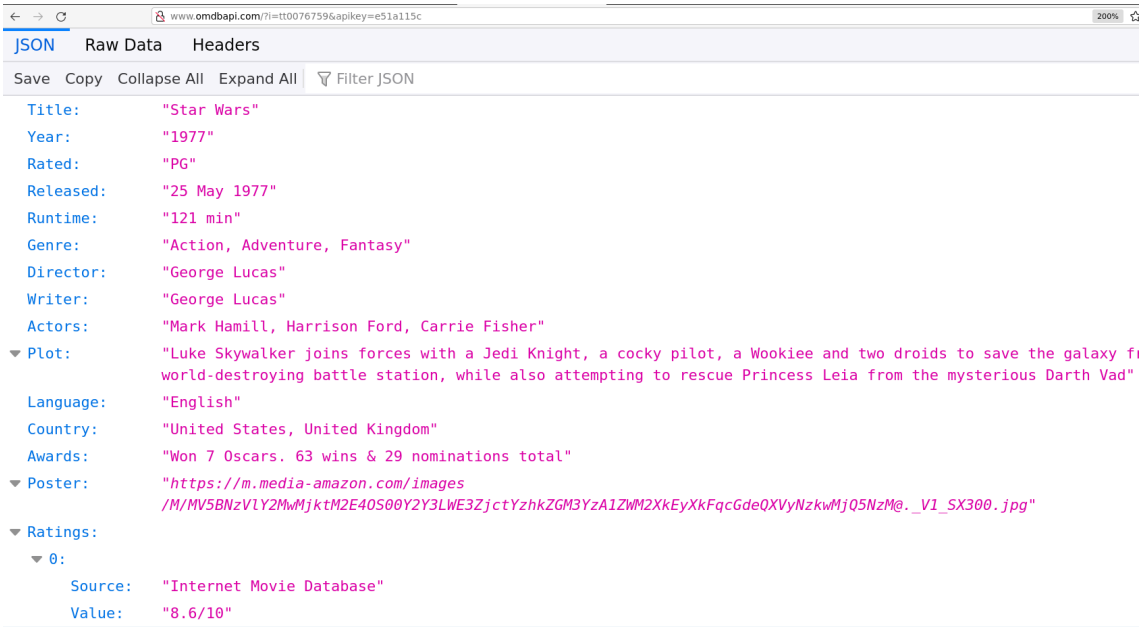


Fonte: do autor, 2021.

Preste especial atenção à URL, onde definimos duas variáveis: s (representando a string que queremos procurar) e apikey (representando a chave de acesso - substitua pela sua!).

O segundo serviço que a API nos fornece é o detalhamento de um filme. Repare que na Figura 5.1, cada filme tem uma imdbID. Com essa ID, podemos requisitar mais detalhes sobre um filme. Veja tal acesso abaixo (Figura 5.2).

Figura 5.2. Filme detalhado



Fonte: do autor, 2021.

Novamente, temos duas variáveis definidas na query string da URL: i, representando a id do filme que queremos detalhar, e apikey.

5.2. Requests

As figuras que já vimos mostram acesso ao OMDB usando o firefox. Mas precisamos aprender a automatizar esse acesso, fazer ele de dentro de nossos códigos python. Para isso, faremos uso da biblioteca requests.

Para instalar essa biblioteca, use o comando **pip install --user requests**, ou, se isso não funcionar, **pip3 install --user requests**, esses comandos devem ser rodados no terminal ou cmd de seu sistema operacional. Se tiver problemas, a solução provável se encontra no fim desse texto.

Codificação 5.1. Biblioteca requests

```
import requests

def busca_por_id(film_id):

   url = f"http://www.omdbapi.com/?apikey={api_key}&i={film_id}"

   pedido = requests.get(url)

   dicionario_do_pedido = pedido.json()

   return dicionario_do_pedido
```

Fonte: do autor, 2021

Uma vez instalada a biblioteca requests, como podemos ver, seu uso é muito simples.

O comando `pedido = requests.get(url)` conecta à URL e baixa o conteúdo relevante. O comando `dicionario_do_pedido = pedido.json()` verifica o conteúdo recebido. Se for um arquivo no formato JSON, ele é processado e retornado ao usuário na variável `dicionario_do_pedido` mas, se o arquivo recebido tiver qualquer outro formato, recebemos um erro. É de se esperar que tais erros ocorram somente enquanto você aprende o funcionamento da webAPI em questão, descobrindo quais URLs retornam arquivos JSON válidos

O resto da função é python usual. Destaque para `url = f"http://www.omdbapi.com/?apikey={api_key}&i={film_id}"` onde montamos uma string para representar a URL desejada, usando uma notação que nos permite interpolar variáveis na nossa string.

5.3. Exercícios

Na videoaula, montamos uma sequência de acessos ao site OMDB. Seguem os exercícios, e logo após, suas respectivas respostas. Note que as respostas não são óbvias, você deve explorar a API do OMDB e sua documentação para conseguir descobrir como baixar cada informação.

Codificação 5.2. Exercícios

```
import requests
from pprint import pprint

'''
A primeira coisa a fazer é ir ao site http://www.omdbapi.com/
e clicar no link API key.

Cadastre-se, abra o e-mail e valide sua chave. Depois, você
poderá acessar o OMDb.
'''

'coloque aqui a sua chave de acesso à api'
api_key = ''

'''
Antes de fazer qualquer função, vamos experimentar
consultar o OMDb pelo navegador.

Digite, por exemplo, a seguinte URL no Firefox:
   http://www.omdbapi.com/?s=star%20wars&apikey=e51a115c

Observe que vemos uma lista de 10 filmes, mas há mais resultados.

Para ver a página 2, acesse
   http://www.omdbapi.com/?s=star%20wars&page=2&apikey=e51a115c
'''


'''
Observe que nas URLs acima, estamos passando parâmetros.
```

```
Na                                                                  URL
http://www.omdbapi.com/?s=star%20wars&page=2&apikey={SUA-CHAVE-VEM-
AQUI}
definimos 3 parâmetros:
* s=star wars
* page=2
* apikey={SUA-CHAVE-VEM-AQUI}
'''

'''
QUESTÃO 1
Olhando para os resultados da consulta
http://www.omdbapi.com/?s=star%20wars&apikey={SUA-CHAVE-VEM-AQUI},
quantos filmes foram encontrados para o termo "star wars"?

Resposta:

QUESTÃO 2
Consultando a documentação em www.omdbapi.com, você
pode aprender a filtrar os resultados da sua busca,
ficando apenas com filmes, eliminando jogos e séries.

Como fazer isso?

Se você fizer essa consulta, quantos filmes
existem para a busca star wars?

Resposta:

QUESTÃO 3:
E se ao invés de filmes você quiser só jogos,
```

```
quantos existem?


Resposta:


'''




'''
Vou te deixar dois exemplos de como acessar a URL. Nesse exemplo,
eu estou retornando o dicionário inteiro.
'''


def busca_por_id(film_id):
    url = f"http://www.omdbapi.com/?apikey={api_key}&i={film_id}"
    pedido = requests.get(url)
    dicionario_do_pedido = pedido.json()
    return dicionario_do_pedido


def busca_por_texto(texto_buscar):
    url_p1 = "http://www.omdbapi.com/"
    url_p2 = f"?apikey={api_key}&s={texto_buscar}"
    url = url_p1 + url_p2
    pedido = requests.get(url) #conectar na URL
    dicionario_do_pedido = pedido.json() #transformo a string que eu
recebi num dicionário de python
    return dicionario_do_pedido



'''
```

```
Experimente! chame d1=busca_por_texto('star wars') e examine o
dicionário d1 retornado.
'''


'''
Agora, faça uma função busca_qtd_total que retorna quantos
itens (pode ser filme, jogo, série ou o que for) batem com
uma determinada busca.
'''
def busca_qtd_total(texto_buscar):
   pass #implemente!


'''
Faça uma função busca_qtd_filmes que retorna quantos
filmes batem com uma determinada busca.
'''
def busca_qtd_filmes(texto_buscar):
   pass #implemente!



'''
Faça uma função busca_qtd_jogos que retorna quantos
jogos batem com uma determinada busca.
'''
def busca_qtd_jogos(texto_buscar):
   pass #implemente!


'''
Agora, vamos aprender a ver os detalhes de um filme.
```

```
Por exemplo, na lista de filmes podemos ver que o filme
star wars original (de 1977) tem id 'tt0076759'

Acessando a URL
http://www.omdbapi.com/?i=tt0076759&apikey={SUA-CHAVE-VEM-AQUI}
podemos ver mais detalhes.

Observe que agora não temos mais o parâmetro 's=star%20wars'
mas sim i=tt0076759. Mudou o nome da "variável", não só
o valor.
'''

'''
Faça uma função nome_do_filme_por_id que recebe a id de
um filme e retorna o seu nome.
'''
def nome_do_filme_por_id(id_filme):
    pass #implemente!

'''
Faça uma função ano_do_filme_por_id que recebe a id de
um filme e retorna o seu ano de lançamento.
'''
def ano_do_filme_por_id(id_filme):
    pass #implemente!

'''
Peguemos vários dados de um filme de uma vez.

A ideia é receber uma id e retornar
```

```
um dicionário com diversos dados do filme.

O dicionário deve ter as seguintes chaves:
* ano
* nome
* diretor
* genero

E os dados devem ser preenchidos baseado nos dados do site.
'''
def dicionario_do_filme_por_id(id_filme):
    pass #implemente!

'''
Voltando para a busca...

Faça uma função busca_filmes que, dada uma busca, retorna
os dez primeiros items (filmes, series, jogos ou o que for)
que batem com a busca.

A sua resposta deve ser uma lista, cada filme representado por
um dicionário. cada dicionario deve conter os campos
'nome' (valor Title da resposta) e 'ano' (valor Year da resposta).
'''
def busca_filmes(texto_buscar):
    pass #implemente!


'''
Faça uma função busca_filmes_grande que, dada uma busca, retorna
```

```
os VINTE primeiros filmes que batem com a busca.
'''
def busca_filmes_grande(texto_buscar):
    pass #implemente!
```

Fonte: do autor, 2021

5.4. Gabarito (das atividades feitas no vídeo)

Codificação 5.3. Gabarito dos exercícios

```
import requests
from pprint import pprint

'''
A primeira coisa a fazer é ir ao site http://www.omdbapi.com/
e clicar no link API key.

Cadastre-se, abra o e-mail e valide sua chave. Depois, você
poderá acessar o OMDb.
'''

'coloque aqui a sua chave de acesso à api'
api_key = 'e51a115c'

'''
Antes de fazer qualquer função, vamos experimentar
consultar o OMDb pelo navegador.

Digite, por exemplo, a seguinte URL no Firefox:
    http://www.omdbapi.com/?s=star%20wars&apikey=e51a115c

Observe que vemos uma lista de 10 filmes, mas há mais resultados.
```

```
Para ver a página 2, acesse

   http://www.omdbapi.com/?s=star%20wars&page=2&apikey=e51a115c
'''



'''
Observe que nas URLs acima, estamos passando parâmetros.
Na                                                          URL
http://www.omdbapi.com/?s=star%20wars&page=2&apikey={SUA-CHAVE-VEM-
AQUI}
definimos 3 parâmetros:
* s=star wars
* page=2
* apikey={SUA-CHAVE-VEM-AQUI}
'''


'''
QUESTÃO 1
Olhando para os resultados da consulta
http://www.omdbapi.com/?s=star%20wars&apikey={SUA-CHAVE-VEM-AQUI},
quantos filmes foram encontrados para o termo "star wars"?

Resposta: 628


QUESTÃO 2
Consultando a documentação em www.omdbapi.com, você
pode aprender a filtrar os resultados da sua busca,
ficando apenas com filmes, eliminando jogos e séries.

Como fazer isso?
```

```
Se você fizer essa consulta, quantos filmes
existem para a busca star wars?

Resposta: 436

QUESTÃO 3:
E se ao invés de filmes você quiser só jogos,
quantos existem?

Resposta: 110

'''




'''
Vou te deixar dois exemplos de como acessar a URL. Nesse exemplo,
eu estou retornando o dicionário inteiro.
'''

def busca_por_id(film_id):
    url = f"http://www.omdbapi.com/?apikey={api_key}&i={film_id}"
    pedido = requests.get(url)
    dicionario_do_pedido = pedido.json()
    return dicionario_do_pedido

def busca_por_texto(texto_buscar):
    url_p1 = "http://www.omdbapi.com/"
    url_p2 = f"?apikey={api_key}&s={texto_buscar}"
```

```
    url = url_p1 + url_p2
    pedido = requests.get(url) #conectar na URL
    dicionario_do_pedido = pedido.json() #transformo a string que eu
recebi num dicionário de python
    return dicionario_do_pedido



'''
Experimente! chame d1=busca_por_texto('star wars') e examine o
dicionário d1 retornado.
'''


'''
Agora, faça uma função busca_qtd_total que retorna quantos
itens (pode ser filme, jogo, série ou o que for) batem com
uma determinada busca.
'''
def busca_qtd_total(texto_buscar):
    dicionario_do_pedido = busca_por_texto(texto_buscar)
    return dicionario_do_pedido['totalResults']


'''
Faça uma função busca_qtd_filmes que retorna quantos
filmes batem com uma determinada busca.
'''
def busca_qtd_filmes(texto_buscar):
    url_p1 = "http://www.omdbapi.com/"
    url_p2 = f"?apikey={api_key}&s={texto_buscar}&type=movie"
    url = url_p1 + url_p2
    pedido = requests.get(url) #conectar na URL
    dicionario_do_pedido = pedido.json() #transformo a string que eu
```

```
recebi num dicionário de python
    return dicionario_do_pedido['totalResults']




'''
Faça uma função busca_qtd_jogos que retorna quantos
jogos batem com uma determinada busca.
'''
def busca_qtd_jogos(texto_buscar):
    url_p1 = "http://www.omdbapi.com/"
    url_p2 = f"?apikey={api_key}&s={texto_buscar}&type=game"
    url = url_p1 + url_p2
    pedido = requests.get(url) #conectar na URL
    dicionario_do_pedido = pedido.json() #transformo a string que eu
recebi num dicionário de python
    return dicionario_do_pedido['totalResults']



'''
Agora, vamos aprender a ver os detalhes de um filme.

Por exemplo, na lista de filmes podemos ver que o filme
star wars original (de 1977) tem id 'tt0076759'

Acessando a URL
http://www.omdbapi.com/?i=tt0076759&apikey={SUA-CHAVE-VEM-AQUI}
podemos ver mais detalhes.

Observe que agora não temos mais o parâmetro 's=star%20wars'
mas sim i=tt0076759. Mudou o nome da "variável", não só
```

```
o valor.
'''


'''
Faça uma função nome_do_filme_por_id que recebe a id de
um filme e retorna o seu nome.
'''
def nome_do_filme_por_id(id_filme):
    url = f"http://www.omdbapi.com/?i={id_filme}&apikey={api_key}"
    pedido = requests.get(url)
    dicionario = pedido.json()
    return dicionario['Title']


'''
Faça uma função ano_do_filme_por_id que recebe a id de
um filme e retorna o seu ano de lançamento.
'''
def ano_do_filme_por_id(id_filme):
    url = f"http://www.omdbapi.com/?i={id_filme}&apikey={api_key}"
    pedido = requests.get(url)
    dicionario = pedido.json()
    return dicionario['Year']


'''
Peguemos vários dados de um filme de uma vez.

A ideia é receber uma id e retornar
um dicionário com diversos dados do filme.

O dicionário deve ter as seguintes chaves:
```

```
* ano
* nome
* diretor
* genero

E os dados devem ser preenchidos baseado nos dados do site.
'''
def dicionario_do_filme_por_id(id_filme):
    #{"ano": 1997, "diretor": "George Lucas",
    #"nome": "Star Wars 40", "genero": "Action, Adventure"}
    dicionario_en = busca_por_id(id_filme)
    dicionario_pt = {}
    #coloco o valor dicionario['Year'] na chave ano do
    #dicionario_pt
    dicionario_pt['ano'] = dicionario_en['Year']
    dicionario_pt['nome'] = dicionario_en['Title']
    dicionario_pt['diretor'] = dicionario_en['Director']
    dicionario_pt['genero'] = dicionario_en['Genre']
    return dicionario_pt

'''
Voltando para a busca...

Faça uma função busca_filmes que, dada uma busca, retorna
os dez primeiros items (filmes, series, jogos ou o que for)
que batem com a busca.

A sua resposta deve ser uma lista, cada filme representado por
um dicionário. cada dicionario deve conter os campos
'nome' (valor Title da resposta) e 'ano' (valor Year da resposta).
```

```
'''
def busca_filmes(texto_buscar):
    dic_busca = busca_por_texto(texto_buscar)
    lista_de_filmes = dic_busca['Search']
    lista_resposta = []
    for filme in lista_de_filmes:
        dic = {}
        dic['nome'] = filme['Title']
        dic['ano'] = filme['Year']
        #guardar o dicionario dic na lista?
        lista_resposta.append(dic)
    return lista_resposta
```

Fonte: do autor, 2021

5.5. Gabarito (função busca filmes grande)

Na videoaula, deixamos um desafio final por resolver: a função busca_filmes_grande. Segue o seu gabarito comentado.

Codificação 5.4. Gabarito do desafio final 1

```
    def busca_por_texto(texto_buscar, pagina=1):
        url_p1 = f"http://www.omdbapi.com/?apikey={api_key}"
        url_p2 = f"&s={texto_buscar}&page={pagina}"
        url = url_p1+url_p2
        pedido = requests.get(url)
        dicionario_do_pedido = pedido.json()
        return dicionario_do_pedido
```

Fonte: do autor, 2021.

Primeiramente, redefinimos a função busca_por_texto, que agora pode ser chamada com ou sem o número de uma página. Ao chamar `>>> busca_por_texto("menace")`, fazemos o acesso como antes, pegando os primeiros 10 filmes. Ao chamar `>>> busca_por_texto("menace",pagina=2)` temos a segunda página dos resultados (ou seja, do 11 ao 20).

Tendo melhorado essas função, podemos fazer a busca de filmes grande. Basta baixar as duas primeiras páginas de resultados.

Codificação 5.5. Gabarito do desafio final 1

```
def busca_filmes_grande(texto_buscar):
    dic_busca1 = busca_por_texto(texto_buscar, pagina=1)
    dic_busca2 = busca_por_texto(texto_buscar, pagina=2)
    lista_de_filmes1 = dic_busca1['Search']
    lista_de_filmes2 = dic_busca2['Search']
    lista_de_filmes = lista_de_filmes1+lista_de_filmes2
    lista_resposta = []
    for filme in lista_de_filmes:
        dic = {}
        dic['nome'] = filme['Title']
        dic['ano'] = filme['Year']
        #guardar o dicionario dic na lista?
        lista_resposta.append(dic)
    return lista_resposta
```

Fonte: do autor, 2021.

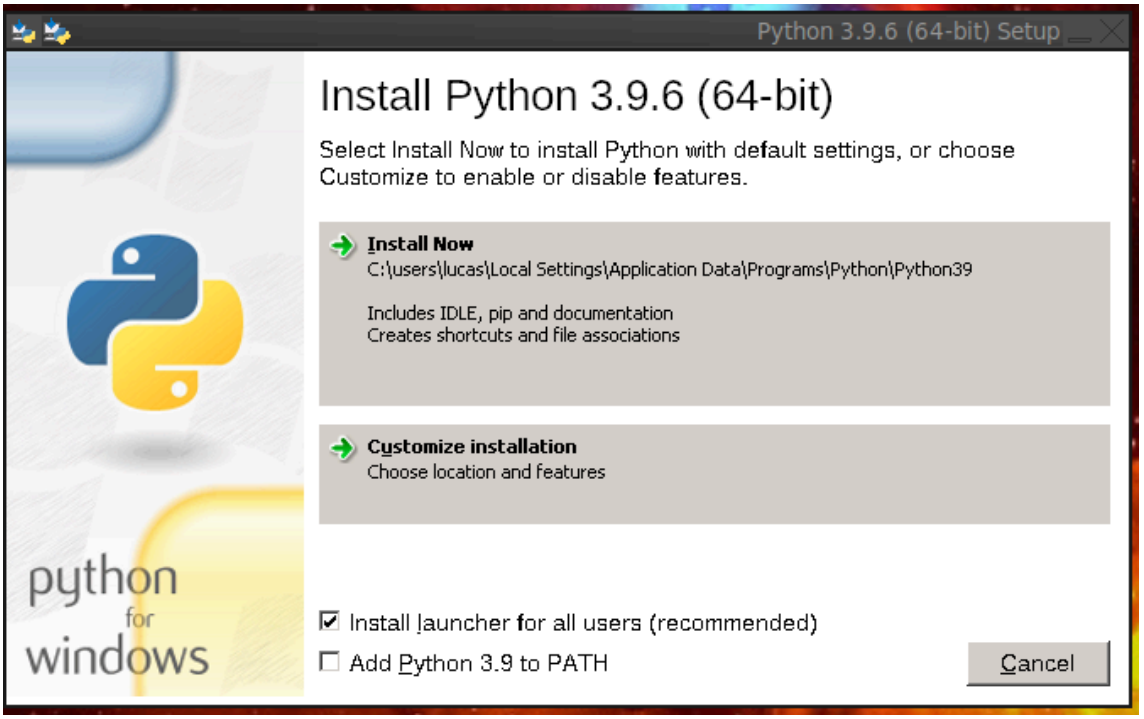
5.6. Solução de problemas

Se você tiver o erro: **O comando pip não é um programa válido - 'pip' não é reconhecido como um comando interno ou externo.** Use os passos abaixo para resolver o problema:

1) Se você está usando linux ou mac, rode o comando de instalação usando pip3 no lugar do pip;
2) Se você está usando windows, experimente o comando python no cmd. Se funcionar (ou seja, o python funciona e o pip não), sua situação não é usual. Peça ajuda no fórum ou ao professor;
3) Se ambos os comandos (pip e python) não funcionarem no cmd, reinstalar o python deve resolver.

Ao reinstalar, marcar a opção “adicionar o python no path” ou “adicionar o python nas variáveis de ambiente”. Isso faz com que os comandos “python” e “pip” passem a ser comandos válidos no cmd.

Figura 5.2. Instalador do python



Fonte: Python, 2021.

Na figura 5.2, vemos que a caixa **Add Python 3.9 to PATH**, está desmarcada. Ache essa opção na parte de baixo da imagem.

Essa é a opção que faz com que os comandos **python** e **pip** estejam disponíveis no **cmd**, e deve ser marcada.

Depois de desinstalar e reinstalar, feche o cmd e abra um novo, para ele carregar os novos comandos.

Referências

PYTHON. **Requests:** HTTP para humanos. Python, [s.d.]. Disponível em: <https://docs.python-requests.org/pt_BR/latest/>. Acesso em: 22 ago. 2021.

PYTHON. **Requests:** HTTP for humans. Python, [s.d.]. Disponível em: <https://docs.python-requests.org/en/master/>. Acesso em: 22 ago. 2021.