In [4]:
```
import requests

pagina = input('Pagina?')
resposta = requests.get('http://letscode.com.br/'+pagina)
print('Status =',resposta.status_code)
print('-'*40,'Conteudo','-'*40)
#print(resposta.text)
```

```
Status = 200
---------------------------------------- Conteudo ----------------------------------------
```

Numbers API

http://numbersapi.com/#42

1

Usando a Numbers API, faça um programa que sorteie um número de 1 a 1000 e imprima uma trivia sobre esse número.

In [11]:
```
import requests
import random

numero = random.randint(1,1001)
print('Obtendo informação sobre o número {} de numbersapi.com'.format(numero))

resposta = requests.get('http://numbersapi.com/{}'.format(numero))
if resposta.status_code > 200:
    print('Atenção! Falha na requisição. Erro:', resposta.status_code)
else:
    print('Resposta:',resposta.text)
```

```
Obtendo informação sobre o número 95 de numbersapi.com
Resposta: 95 is the percentage confidence interval that is considered satisfactory for most purposes in statistics.
```

2

Usando a Numbers API, faça um programa que sorteie um número de 1 a 1000 e imprima um fato matemático (tipo math) sobre esse número.

In [16]:
```
import requests
import random
```

```
numero = random.randint(1,1001)
print('Obtendo informação sobre o número {} de numbersapi.com'.format(numero))

resposta = requests.get('http://numbersapi.com/{}/math'.format(numero))
if resposta.status_code > 200:
    print('Atenção! Falha na requisição. Erro:', resposta.status_code)
else:
    print('Resposta:',resposta.text)
```

```
Obtendo informação sobre o número 344 de numbersapi.com
Resposta: 344 is the smallest number that can be written as the sum of a cube and a 7^{th} power in more than one way.
```

3

Usando a Numbers API, faça um programa que receba a data do seu aniversário (no formato mês/dia, como string) e imprima um fato que ocorreu nesse dia.

In [4]:
```
import requests
from datetime import date

while True:
    try:
        dia = int(input('Informe o dia de seu aniversário'))
        mes = int(input('Informe o mês de seu aniversário'))
        data = date(2000, mes, dia)
        break
    except Exception as e:
        print('Mes ou ano inválido. Tente novamente.')

resposta = requests.get('http://numbersapi.com/{}/{}/date'.format(mes, dia))
if resposta.status_code > 200:
    print('Atenção! Falha na requisição. Erro:', resposta.status_code)
else:
    print('Resposta:',resposta.text)
```

```
Mes ou ano inválido. Tente novamente.

Resposta: December 19th is the day in 1963 that Zanzibar gains independence from the United Kingdom as a constitutional monarchy, under Sultan Jamshid bin Abd
ullah.
```

4

Usando a Exchange Rate API, faça um programa que imprima a taxa de conversão de reais para dólares americanos em tempo real. Seu programa deve imprimir também a data da última atualização da API.

In [1]:
```
import requests

chave = '<chave_pessoal>'
```

```
def req_exrate(req):
    cmd = 'https://v6.exchangerate-api.com/v6/{}/{}'.format(chave, req)
    return requests.get(cmd)

print("Consultando ExchangeRate...")
resposta = req_exrate('latest/USD')
if resposta.status_code > 200:
    print('Atenção! Falha na requisição. Erro:', resposta.status_code)
else:
    print("Ok. Dados recebidos:")
    resp = resposta.json()
    # print(resp)
    dt_atualiz = resp['time_last_update_utc']
    taxa = resp['conversion_rates']['BRL']
    print('Taxa de conversão USD x BRL = {}'.format(taxa))
    print('Dt. de Atualizacao = {}'.format(dt_atualiz))
```

```
Consultando ExchangeRate...
Ok. Dados recebidos:
Taxa de conversão USD x BRL = 5.2322
Dt. de Atualizacao = Thu, 22 Jul 2021 00:00:02 +0000
```

5

Usando a Exchange Rate API, faça um programa que receba um valor em reais e o converta para dólares americanos. Seu programa deve imprimir também a data da última atualização da taxa.

In [4]:
```
import requests

chave = '<chave_pessoal>'

def req_exrate(req):
    cmd = 'https://v6.exchangerate-api.com/v6/{}/{}'.format(chave, req)
    return requests.get(cmd)

while True:
    try:
        valorReais = float(input('Informe valor em R$ (use ponto para informar decimais) '))
        break
    except:
        print('Valor invalido. Tente novamente.')

print("Por favor aguarde, obtendo valor de USD atualizado...")
resposta = req_exrate('latest/USD')
if resposta.status_code > 200:
    print('Atenção! Falha na requisição. Erro:', resposta.status_code)
else:
    print('Dt. da última atualizacao: {}'.format(dt_atualiz))
```

```
    resp = resposta.json()
    # print(resp)
    dt_atualiz = resp['time_last_update_utc']
    valorUSD = float(resp['conversion_rates']['BRL'])
    reaisEmUSD = valorReais / valorUSD
    print('{0} em USD = {1:.2f}'.format(valorReais, reaisEmUSD))
```

```
Por favor aguarde, obtendo valor de USD atualizado...
Dt. da última atualizacao: Thu, 22 Jul 2021 00:00:02 +0000
150.456 em USD = 28.76
```

6

Usando a Exchange Rate API, faça um programa que receba a sigla de duas moedas, ORIGEM e DESTINO, e um valor X. Seu programa deve converter o valor X da moeda ORIGEM para a moeda DESTINO.

In [4]:
```
import requests

chave = '<chave_pessoal>'

while True:
    try:
        moedaOrigem = input('Informe a sigla da moeda do valor a converter')
        valorOrigem = float(input('Informe valor a converter (use ponto para informar decimais)'))
        moedaDest   = input('Informe a sigla da moeda para a qual converter')
        break
    except:
        print('Valor invalido. Tente novamente.\n* * *\n')

print("Por favor aguarde, obtendo valor atualizado das moedas e calculando...")

resposta = requests.get('https://v6.exchangerate-api.com/v6/{}/pair/{}/{}'.format(chave, moedaOrigem, moedaDest))
if resposta.status_code > 200:
    print('Atenção! Falha na requisição. Erro:', resposta.status_code)
else:
    resp = resposta.json()
    taxa = float(resp['conversion_rate'])
    valorDest = valorOrigem * taxa
    print('{0} {1} = {2} {3:.2f}'.format(moedaOrigem, valorOrigem, moedaDest, valorDest))
```

```
Por favor aguarde, obtendo valor atualizado das moedas e calculando...
BRL 1000.0 = EUR 162.30
```

7

Refaça o exercício 6 orientado a funções. Isto é, faça uma função que recebe duas moedas, ORIGEM e DESTINO, e um valor X. Sua função deve retornar o valor X convertido da moeda ORIGEM para a moeda DESTINO.

In [7]:
```
import requests

chave = '<chave_pessoal>'

def conversaoVlrMonet(moedaOrigem, moedaDest, valorOrigem):
    resposta = requests.get('https://v6.exchangerate-api.com/v6/{}/pair/{}/{}'.format(chave, moedaOrigem, moedaDest))
    if resposta.status_code > 200:
        raise Exception('Atenção! Falha na requisição. Erro:', resposta.status_code)
    resp = resposta.json()
    taxa = float(resp['conversion_rate'])
    return valorOrigem * taxa

while True:
    try:
        moedaOrigem = input('Informe a sigla da moeda do valor a converter')
        valorOrigem = float(input('Informe valor a converter (use ponto para informar decimais)'))
        moedaDest   = input('Informe a sigla da moeda para a qual converter')
        break
    except:
        print('Valor invalido. Tente novamente.\n* * *\n')

print("Por favor aguarde, obtendo valor atualizado das moedas e calculando...")
valorDest = conversaoVlrMonet(moedaOrigem, moedaDest, valorOrigem)
print('{0} {1} = {2} {3:.2f}'.format(moedaOrigem, valorOrigem, moedaDest, valorDest))
```

```
Por favor aguarde, obtendo valor atualizado das moedas e calculando...
USD 100.0 = ARS 9629.91
```

8

Refaça o exercício 6 orientado a objeto. Você deve construir uma classe Moeda. Essa classe deve ter como atributo a sigla da moeda e o dicionário de conversão. Faça um método para requisitar os dados. Faça também um método que receba uma outra moeda e um valor X, e retorne o valor X convertido para essa moeda recebida.

In [4]:
```
import requests

# Erros da classe Moeda
class ErroMoeda(Exception):
    pass

# Classe para conversão de valores em diferentes moedas
class Moeda:

    # Chave e URL da API
    _chaveAPI = '<chave_pessoal>'
```

```
    _urlAPI = 'https://v6.exchangerate-api.com/v6/'+_chaveAPI+'/'

    # Construtor
    # Se o código de moeda desejado não for informado ele é solicitado
    def __init__(self, codigoMoeda=''):
        self._codigo = ''
        self._dicTaxas = {}
        self._alterarCodigo(codigoMoeda)

    # Propriedade "codigo" de moeda
    @property
    def codigo(self):
        return self._codigo

    @codigo.setter
    def codigo(self, novoCodigo):
        _alterarCodigo(novoCodigo)

    # Método para alterar o código de moeda
    # Se o código não mudar, nada é feito.
    # Se um novo código for informado descarta os dados carregados da moeda anterior
    def _alterarCodigo(self, novoCodigo):
        while len(novoCodigo) < 1:
            novoCodigo = input('Informe o código da moeda a carregar')
        if self._codigo != novoCodigo:
            self._codigo = novoCodigo
            self.dicTaxas = {}

    # Busca dados da moeda quando necessário (quando ainda não carregados)
    def _atualizar(self):
        if self._dicTaxas.__len__() < 1:
            try:
                resp = requests.get(Moeda._urlAPI + 'latest/' + self._codigo)
                if resp.status_code != 200:
                    raise ErroMoeda('Falha ao atualizar. Codigo={}'.format(resp.status_code))
                respJson = resp.json()
                self._dicTaxas = respJson['conversion_rates'].copy()
            except Exception as e:
                raise ErroMoeda(e.str())

    # Converte um valor numa outra moeda para essa moeda
    def converterPara(self, moedaDestino, valor):
        try:
            # Garante que os dados de conversão foram carregados
            self._atualizar()
            # Obtem a taxa para a moeda informada (que pode falhar)
            taxaDestino = self._dicTaxas[moedaDestino]
        except Exception as e:
            raise ErroMoeda('Moeda desconhecida: '+moedaDestino)
        # Retorna a conversão do valor fornecido na moeda carregada
        return valor / taxaDestino
```

```
#
# Testes
#

# Laço para executar com várias moedas base
while True:
    print('Informe 0 para encerrar')
    moedaRef = Moeda()
    if moedaRef.codigo == '0':
        break

    # Laço para executar várias conversões com a moeda de referência selecionada
    while True:
        try:
            # Valor a converter
            valorOrigem = float(input('Informe valor a converter (0 para selecionar outra carregar outra moeda-base):'))
            if valorOrigem == 0:
                break
        except:
            print('Valor invalido.\n\n')
        try:
            # Moeda em que o valor foi fornecido
            moedaOrigem = input('Em qual moeda (vazio para voltar para entrada de valor)?')
            if len(moedaOrigem.strip(' ')) > 1:
                # Obtem a quantidade correspondente na moeda de referência e apresenta o resultado
                valorRef = moedaRef.converterPara(moedaOrigem, valorOrigem)
                print('{0: f} {1} = {2:.4f} {3}'.format(valorOrigem, moedaOrigem, valorRef, moedaRef.codigo))
                print()
        except:
            print('Moeda desconhecida.\n\n')
```

```
Informe 0 para encerrar
 1.000000 USD = 5.2329 BRL


 1.000000 EUR = 6.1614 BRL


 100.000000 ARS = 5.4284 BRL


Informe 0 para encerrar
 10.000000 BRL = 1.9112 USD


 10.000000 EUR = 11.7966 USD


 1.000000 EUR = 1.1797 USD
```

```
 1.000000 BRL = 0.1911 USD


Informe 0 para encerrar
```

9

Para os exercícios 9 e 10, usaremos a Open Weather API. Você deve realizar consultas através do link: http://api.openweathermap.org/data/2.5/weather?q={NOME_DA_CIDADE}&appid={CHAVE_DE_ACESSO}&units=metric

Substituindo NOME_DA_CIDADE pelo nome da cidade desejada e CHAVE_DE_ACESSO pela seguinte string:

Usando a Open Weather API, faça uma função que receba um nome de uma cidade e imprima a temperatura máxima e mínima, a umidade, a descrição do tempo e o horário da última atualização.

In [67]:
```python
import requests
import time

urlAPI = 'http://api.openweathermap.org/data/2.5/weather?q={0}&appid=cb926da04c58d12e68807544e9a35f6c&units=metric'

# Recebe nome da cidade na forma <nome>,<UF>,<País> e exibe informações sobre o clima
def infoCidade(nomeCidade):
    print('CLIMA')
    print('=====')
    print('Dados para '+nomeCidade)
    print()

    # Consulta serviço
    respostaAPI = requests.get(urlAPI.format(nomeCidade))
    if respostaAPI.status_code != 200:
        # Algum erro ocorreu
        print('Erro na requisição dos dados. Codigo = {}'.format(respostaAPI.status_code))
    else:
        # Converte resposta para JSON
        dados = respostaAPI.json()

        # Coleta dados de temperaturas e umidade
        principal = dados['main']
        tempMin = principal['temp_min']
        tempMax = principal['temp_max']
        umidade = principal['humidity']
        # Coleta descrição do clima
        clima = dados['weather'][0]
        descr   = clima['description']

        # Coleta data e hora da atualização
        # Como a hora é armazenada no provedor em UTC utilizei o método localtime() que traduz automaticamente para o fuso horário local
        timeStamp = time.localtime(int(dados['dt']))
```

```
        dataStr = time.strftime('%d/%m/%y %H:%M', timeStamp)

        # Apresenta dados coletados
        print('Temperatura máxima: {}'.format(tempMax))
        print('Temperatura mínima: {}'.format(tempMin))
        print('Umidade...........: {}'.format(umidade))
        print('Clima.............: {}'.format(descr))
        print()
        print('Atualizado em.....: {}'.format(dataStr))
        print()

# Testes
infoCidade('Santo Andre,SP,Brazil')
infoCidade('New York,NY,United States')
infoCidade('Ancara,Turkey')
```

```
CLIMA
=====
Dados para Santo Andre,SP,Brazil

Temperatura máxima: 23.8
Temperatura mínima: 23.8
Umidade...........: 93
Clima.............: scattered clouds

Atualizado em.....: 22/07/21 01:08

CLIMA
=====
Dados para New York,NY,United States

Temperatura máxima: 23.45
Temperatura mínima: 18.39
Umidade...........: 67
Clima.............: clear sky

Atualizado em.....: 22/07/21 01:13

CLIMA
=====
Dados para Ancara,Turkey

Temperatura máxima: 20.67
Temperatura mínima: 18.51
Umidade...........: 63
Clima.............: clear sky

Atualizado em.....: 22/07/21 01:11
```

Refaça o exercício 9 utilizando orientação a objetos. Isto é, faça uma classe Cidade cujos atributos são: nome, temperatura máxima, temperatura mínima, umidade, descrição do tempo e última atualização. Construa métodos para pegar os dados e use o método mágico repr para exibir os dados.

In [89]:
```python
import requests
import time

class ErroClima(Exception):
    pass

class Clima:

    # Local da API
    _urlAPI = 'http://api.openweathermap.org/data/2.5/weather?q={0}&appid=cb926da04c58d12e68807544e9a35f6c&units=metric'

    # Construtor
    def __init__(self, nomeCidade):
        self._nome = nomeCidade
        self._tempMin = 0
        self._tempMax = 0
        self._umidade = 0
        self._descricao = ''
        self._ultimaAtualiz = None
        self.atualizar()

    # Atualiza dados da cidade a partir do serviço remoto, via API
    def atualizar(self):
        respostaAPI = requests.get(urlAPI.format(self.nome))
        if respostaAPI.status_code != 200:
            # Algum erro ocorreu
            if respostaAPI.status_code == 404:
                raise ErroClima('Cidade não localizada.')
            else:
                raise ErroClima('Erro na requisição dos dados. Codigo = {}'.format(respostaAPI.status_code))

        # Converte resposta para JSON
        dados = respostaAPI.json()

        # Coleta dados de temperaturas e umidade
        principal = dados['main']
        self._tempMin = principal['temp_min']
        self._tempMax = principal['temp_max']
        self._umidade = principal['humidity']
        # Coleta descrição do clima
        clima = dados['weather'][0]
        self._descricao = clima['description']

        # Coleta data e hora da atualização
        # Como a hora é armazenada no provedor em UTC utilizei o método localtime() que traduz automaticamente para o fuso horário local
        timeStamp = time.localtime(int(dados['dt']))
        self._ultimaAtualiz = time.strftime('%d/%m/%y %H:%M', timeStamp)
```

```
    # Representação do objeto para exibição
    def __repr__(self):
        s = 'CLIMA\n'+\
            '=====\n'+\
            'Dados de {}\n\n'.format(self._nome)+\
            'Temperatura máxima: {:3.2f}C / {:3.2f}F\n'.format(self.tempMaxima, self.tempMaximaF)+\
            'Temperatura mínima: {:3.2f}C / {:3.2f}F\n'.format(self.tempMinima, self.tempMinimaF)+\
            'Umidade...........: {}%\n'.format(self.umidade)+\
            'Clima.............: {}\n\n'.format(self.descricao)+\
            'Atualizado em.....: {}\n'.format(self.ultimaAtualizacao)
        return s

    def _farenheit(self, tempC):
        return tempC * 9/5 + 32

    @property
    def nome(self):
        return self._nome

    @nome.setter
    def nome(self, novoNome):
        self._nome = novoNome
        self.atualizar()

    @property
    def tempMinima(self):
        return self._tempMin

    @property
    def tempMinimaF(self):
        return self._farenheit(self.tempMinima)

    @property
    def tempMaxima(self):
        return self._tempMax

    @property
    def tempMaximaF(self):
        return self._farenheit(self.tempMaxima)

    @property
    def umidade(self):
        return self._umidade

    @property
    def descricao(self):
        return self._descricao

    @property
    def ultimaAtualizacao(self):
        return self._ultimaAtualiz
```

```python
# Testes
while True:
    cidade = input('\nInforme uma cidade a pesquisar (FIM para sair):')
    if cidade.upper() == 'FIM':
        break
    try:
        clima = Clima(cidade)
        print(clima)
    except Exception as e:
        print(e.__str__())
```

```
CLIMA
=====
Dados de sao paulo, br

Temperatura máxima: 12.88C / 55.18F
Temperatura mínima: 7.94C / 46.29F
Umidade...........: 89%
Clima.............: scattered clouds

Atualizado em.....: 22/07/21 01:45


CLIMA
=====
Dados de santo andre, br

Temperatura máxima: 12.71C / 54.88F
Temperatura mínima: 7.76C / 45.97F
Umidade...........: 86%
Clima.............: scattered clouds

Atualizado em.....: 22/07/21 01:46


CLIMA
=====
Dados de natal, br

Temperatura máxima: 23.12C / 73.62F
Temperatura mínima: 22.36C / 72.25F
Umidade...........: 83%
Clima.............: scattered clouds

Atualizado em.....: 22/07/21 01:49
```

11

Para os exercícios 9 ao 11, vamos utilizar uma API do Star Wars, chamada de SWAPI. Entre no site deles para se informar sobre as respostas e requisições: https://swapi.dev/

Faça um programa que imprima o nome e o ano de nascimento dos 50 primeiros personagens listados no site.

In [8]:
```
import requests

urlAPI = 'https://swapi.dev/api/people/{}'

for i in range(1,51):
    dados = requests.get(urlAPI.format(i))
    if dados.status_code != 200:
        print('Algo deu errado. Erro {}'.format(dados.status_code))
    else:
        dadosJson = dados.json()
        print('{}) - Nome: {} - Nascimento: {}'.format(i, dadosJson['name'], dadosJson['birth_year']))
```

1) - Nome: Luke Skywalker - Nascimento: 19BBY
2) - Nome: C-3PO - Nascimento: 112BBY
3) - Nome: R2-D2 - Nascimento: 33BBY
4) - Nome: Darth Vader - Nascimento: 41.9BBY
5) - Nome: Leia Organa - Nascimento: 19BBY
6) - Nome: Owen Lars - Nascimento: 52BBY
7) - Nome: Beru Whitesun lars - Nascimento: 47BBY
8) - Nome: R5-D4 - Nascimento: unknown
9) - Nome: Biggs Darklighter - Nascimento: 24BBY
10) - Nome: Obi-Wan Kenobi - Nascimento: 57BBY
11) - Nome: Anakin Skywalker - Nascimento: 41.9BBY
12) - Nome: Wilhuff Tarkin - Nascimento: 64BBY
13) - Nome: Chewbacca - Nascimento: 200BBY
14) - Nome: Han Solo - Nascimento: 29BBY
15) - Nome: Greedo - Nascimento: 44BBY
16) - Nome: Jabba Desilijic Tiure - Nascimento: 600BBY
Algo deu errado. Erro 404
18) - Nome: Wedge Antilles - Nascimento: 21BBY
19) - Nome: Jek Tono Porkins - Nascimento: unknown
20) - Nome: Yoda - Nascimento: 896BBY
21) - Nome: Palpatine - Nascimento: 82BBY
22) - Nome: Boba Fett - Nascimento: 31.5BBY
23) - Nome: IG-88 - Nascimento: 15BBY
24) - Nome: Bossk - Nascimento: 53BBY
25) - Nome: Lando Calrissian - Nascimento: 31BBY
26) - Nome: Lobot - Nascimento: 37BBY
27) - Nome: Ackbar - Nascimento: 41BBY
28) - Nome: Mon Mothma - Nascimento: 48BBY
29) - Nome: Arvel Crynyd - Nascimento: unknown
30) - Nome: Wicket Systri Warrick - Nascimento: 8BBY
31) - Nome: Nien Nunb - Nascimento: unknown
32) - Nome: Qui-Gon Jinn - Nascimento: 92BBY
33) - Nome: Nute Gunray - Nascimento: unknown
34) - Nome: Finis Valorum - Nascimento: 91BBY
35) - Nome: Padmé Amidala - Nascimento: 46BBY
36) - Nome: Jar Jar Binks - Nascimento: 52BBY
37) - Nome: Roos Tarpals - Nascimento: unknown
38) - Nome: Rugor Nass - Nascimento: unknown
39) - Nome: Ric Olié - Nascimento: unknown
40) - Nome: Watto - Nascimento: unknown
41) - Nome: Sebulba - Nascimento: unknown
42) - Nome: Quarsh Panaka - Nascimento: 62BBY
43) - Nome: Shmi Skywalker - Nascimento: 72BBY
44) - Nome: Darth Maul - Nascimento: 54BBY
45) - Nome: Bib Fortuna - Nascimento: unknown
46) - Nome: Ayla Secura - Nascimento: 48BBY
47) - Nome: Ratts Tyerel - Nascimento: unknown
48) - Nome: Dud Bolt - Nascimento: unknown
49) - Nome: Gasgano - Nascimento: unknown
50) - Nome: Ben Quadinaros - Nascimento: unknown

12

Implemente o programa criado no exercício 11, adicionando o nome do planeta de origem de cada personagem.

Obs: você precisará fazer uma nova requisição para o planeta, caso ele não seja desconhecido.

In [9]:
```python
import requests

urlAPI = 'https://swapi.dev/api/people/{}'

def reqJson(url):
    resposta = requests.get(url)
    status = resposta.status_code
    if status == 200:
        descrStatus = ''
        dadosJson = resposta.json()
    else:
        descrStatus = 'Algo deu errado. Erro {}'.format(status)
        dadosJson = None
    return (status, descrStatus, dadosJson)

for i in range(1,51):
    tuplaResp = reqJson(urlAPI.format(i))
    if tuplaResp[0] != 200:
        print(tuplaResp[1])
    else:
        dados = tuplaResp[2]
        nome = dados['name']
        nascto = dados['birth_year']
        urlPlaneta = dados['homeworld']

        tuplaResp = reqJson(urlPlaneta)
        if tuplaResp[0] != 200:
            nomePlaneta = '*** Desconhecido ***'
        else:
            dados = tuplaResp[2]
            nomePlaneta = dados['name']
        print('{}) - Nome: {} - Nascimento: {} - Planeta: {}'.format(i, nome, nascto, nomePlaneta))
```

1) - Nome: Luke Skywalker - Nascimento: 19BBY - Planeta: Tatooine
2) - Nome: C-3PO - Nascimento: 112BBY - Planeta: Tatooine
3) - Nome: R2-D2 - Nascimento: 33BBY - Planeta: Naboo
4) - Nome: Darth Vader - Nascimento: 41.9BBY - Planeta: Tatooine
5) - Nome: Leia Organa - Nascimento: 19BBY - Planeta: Alderaan
6) - Nome: Owen Lars - Nascimento: 52BBY - Planeta: Tatooine
7) - Nome: Beru Whitesun lars - Nascimento: 47BBY - Planeta: Tatooine
8) - Nome: R5-D4 - Nascimento: unknown - Planeta: Tatooine
9) - Nome: Biggs Darklighter - Nascimento: 24BBY - Planeta: Tatooine
10) - Nome: Obi-Wan Kenobi - Nascimento: 57BBY - Planeta: Stewjon
11) - Nome: Anakin Skywalker - Nascimento: 41.9BBY - Planeta: Tatooine
12) - Nome: Wilhuff Tarkin - Nascimento: 64BBY - Planeta: Eriadu
13) - Nome: Chewbacca - Nascimento: 200BBY - Planeta: Kashyyyk
14) - Nome: Han Solo - Nascimento: 29BBY - Planeta: Corellia
15) - Nome: Greedo - Nascimento: 44BBY - Planeta: Rodia
16) - Nome: Jabba Desilijic Tiure - Nascimento: 600BBY - Planeta: Nal Hutta
Algo deu errado. Erro 404
18) - Nome: Wedge Antilles - Nascimento: 21BBY - Planeta: Corellia
19) - Nome: Jek Tono Porkins - Nascimento: unknown - Planeta: Bestine IV
20) - Nome: Yoda - Nascimento: 896BBY - Planeta: unknown
21) - Nome: Palpatine - Nascimento: 82BBY - Planeta: Naboo
22) - Nome: Boba Fett - Nascimento: 31.5BBY - Planeta: Kamino
23) - Nome: IG-88 - Nascimento: 15BBY - Planeta: unknown
24) - Nome: Bossk - Nascimento: 53BBY - Planeta: Trandosha
25) - Nome: Lando Calrissian - Nascimento: 31BBY - Planeta: Socorro
26) - Nome: Lobot - Nascimento: 37BBY - Planeta: Bespin
27) - Nome: Ackbar - Nascimento: 41BBY - Planeta: Mon Cala
28) - Nome: Mon Mothma - Nascimento: 48BBY - Planeta: Chandrila
29) - Nome: Arvel Crynyd - Nascimento: unknown - Planeta: unknown
30) - Nome: Wicket Systri Warrick - Nascimento: 8BBY - Planeta: Endor
31) - Nome: Nien Nunb - Nascimento: unknown - Planeta: Sullust
32) - Nome: Qui-Gon Jinn - Nascimento: 92BBY - Planeta: unknown
33) - Nome: Nute Gunray - Nascimento: unknown - Planeta: Cato Neimoidia
34) - Nome: Finis Valorum - Nascimento: 91BBY - Planeta: Coruscant
35) - Nome: Padmé Amidala - Nascimento: 46BBY - Planeta: Naboo
36) - Nome: Jar Jar Binks - Nascimento: 52BBY - Planeta: Naboo
37) - Nome: Roos Tarpals - Nascimento: unknown - Planeta: Naboo
38) - Nome: Rugor Nass - Nascimento: unknown - Planeta: Naboo
39) - Nome: Ric Olié - Nascimento: unknown - Planeta: Naboo
40) - Nome: Watto - Nascimento: unknown - Planeta: Toydaria
41) - Nome: Sebulba - Nascimento: unknown - Planeta: Malastare
42) - Nome: Quarsh Panaka - Nascimento: 62BBY - Planeta: Naboo
43) - Nome: Shmi Skywalker - Nascimento: 72BBY - Planeta: Tatooine
44) - Nome: Darth Maul - Nascimento: 54BBY - Planeta: Dathomir
45) - Nome: Bib Fortuna - Nascimento: unknown - Planeta: Ryloth
46) - Nome: Ayla Secura - Nascimento: 48BBY - Planeta: Ryloth
47) - Nome: Ratts Tyerel - Nascimento: unknown - Planeta: Aleen Minor
48) - Nome: Dud Bolt - Nascimento: unknown - Planeta: Vulpter
49) - Nome: Gasgano - Nascimento: unknown - Planeta: Troiken
50) - Nome: Ben Quadinaros - Nascimento: unknown - Planeta: Tund

13

Usando a API do Star Wars, faça um programa que imprima o nome dos personagens que apareceram no filme 4: "The Phantom Menace".

In [20]:
```
import requests

urlAPI_filme = 'https://swapi.dev/api/films/{}/'
urlAPI_pessoa = 'https://swapi.dev/api/people/{}/'

# Função que simplifica tratamento de requisições
def reqJson(url):
    resposta = requests.get(url)
    status = resposta.status_code
    if status == 200:
        descrStatus = ''
        dadosJson = resposta.json()
    else:
        descrStatus = 'Algo deu errado. Erro {}'.format(status)
        dadosJson = None
    return (status, descrStatus, dadosJson)

# Titulo do filme desejado
tituloDesejado = 'The Phantom Menace'

# Procura o filme desejado
idFilme = 1
while idFilme > 0:
    resposta = reqJson(urlAPI_filme.format(idFilme))
    if resposta[0] == 404:
        print('Erro 404')
        idFilme = -1
    elif resposta[0] != 200:
        print(resposta[1])
        idFilme += 1
    elif resposta[2]['title'] != tituloDesejado:
        idFilme += 1
    else:
        # Lista os personagens do filme
        print('Lista dos personagens do filme "{}"'.format(tituloDesejado))
        listaURLpersonagens = resposta[2]['characters']
        for url in listaURLpersonagens:
            resposta = reqJson(url)
            if resposta[0] != 200:
                print(resposta[1])
            else:
                print(resposta[2]['name'])
        idFilme = 0
print('\n* Fim *')
```

```
Lista dos personagens do filme "The Phantom Menace"
C-3PO
R2-D2
Obi-Wan Kenobi
Anakin Skywalker
Jabba Desilijic Tiure
Yoda
Palpatine
Qui-Gon Jinn
Nute Gunray
Finis Valorum
Padmé Amidala
Jar Jar Binks
Roos Tarpals
Rugor Nass
Ric Olié
Watto
Sebulba
Quarsh Panaka
Shmi Skywalker
Darth Maul
Ayla Secura
Ratts Tyerel
Dud Bolt
Gasgano
Ben Quadinaros
Mace Windu
Ki-Adi-Mundi
Kit Fisto
Eeth Koth
Adi Gallia
Saesee Tiin
Yarael Poof
Plo Koon
Mas Amedda

* Fim *
```

14

Para os desafios 1 e 2, vamos utilizar uma API do Governo Federal para analisar os gastos por meio de cartão de pagamento. O link da API é:

"http://www.transparencia.gov.br/api-de-dados/cartoes?mesExtratoInicio={}%2F{}&mesExtratoFim={}%2F{}&pagina={}".format(mes_ini, ano_ini, mes_fim, ano_fim, pagina)

onde os dados retornados correspondem a um intervalo definido por um mês e ano inicial (mes_ini e ano_ini) e um mês e ano final (mes_fim, ano_fim). São apresentados 14 dados por página, com a primeira página sendo definida por pagina = 1.

Desafio 1 - Faça uma requisição da API para obter os dados entre 06/2018 e 07/2018 e responda:

a. Quantos pagamentos foram realizados por meio de cartão de pagamento nesse intervalo?

Obs: lembre-se de checar todas as páginas possíveis. Você pode usar um loop infinito para isso, e sair do loop quando a requisição não retornar 200;

b. Qual foi o maior valor de transação?

c. Qual o nome do portador do cartão responsável por esse gasto?

In [4]:
```
# Doc da API: http://api.portaldatransparencia.gov.br/swagger-ui.html

# Formato da resposta:
#[
#  {
#    "dataTransacao": "string",
#    "estabelecimento": {
#      "cnpjFormatado": "string",
#      "cpfFormatado": "string",
#      "id": 0,
#      "nome": "string",
#      "nomeFantasiaReceita": "string",
#      "numeroInscricaoSocial": "string",
#      "razaoSocialReceita": "string",
#      "tipo": "string"
#    },
#    "id": 0,
#    "mesExtrato": "string",
#    "portador": {
#      "cpfFormatado": "string",
#      "nis": "string",
#      "nome": "string"
#    },
#    "tipoCartao": {
#      "codigo": "string",
#      "descricao": "string",
#      "id": 0
#    },
#    "unidadeGestora": {
#      "codigo": "string",
#      "descricaoPoder": "string",
#      "nome": "string",
#      "orgaoMaximo": {
#        "codigo": "string",
#        "nome": "string",
#        "sigla": "string"
#      },
#      "orgaoVinculado": {
#        "cnpj": "string",
#        "codigoSIAFI": "string",
#        "nome": "string",
#        "sigla": "string"
```

```
#      }
#    },
#    "valorTransacao": "string"
#  }
#]
#
import requests
import time

urlAPI = 'http://api.portaldatransparencia.gov.br/api-de-dados/cartoes?mesExtratoFim={2}%2F{3}&mesExtratoInicio={0}%2F{1}&pagina={4}'
chaveAPI = '<chave_pessoal>'

# Função que simplifica tratamento de requisições
def reqJson(url):
    # Pausa para não sobrecarregar o serviço, que possui limitação de atendimento de 90 requests por minuto
    time.sleep(0.15)
    resposta = requests.get(url, headers={'chave-api-dados':chaveAPI})
    status = resposta.status_code
    if status == 200:
        descrStatus = ''
        dadosJson = resposta.json()
    else:
        descrStatus = 'Algo deu errado. Erro {}'.format(status)
        dadosJson = None
    return (status, descrStatus, dadosJson)

# Periodo a consultar
mesInicial = '06'
anoInicial = '2018'
mesFinal   = '07'
anoFinal   = '2018'
# Contagem de pagamentos
qtdPagamentos = 0
# Responsável e valor do maior pagamento
respMaiorPagamento = None
maiorPagamento = 0

# Processar todas as páginas
pagina = 1
while pagina > 0:
    # Habilitar a linha a seguir para um feedback visual de progresso (no início de cada grupo de N paginas)
    if pagina % 10 == 1:
        print(pagina, end=",")
    resposta = reqJson(urlAPI.format(mesInicial, anoInicial, mesFinal, anoFinal, pagina))
    if resposta[0] != 200:
        # Fim da pesquisa
        pagina = 0
    else:
        # Obtem lista de lançamentos da página
        lancamentos = resposta[2]
        processados = 0
        for lcto in lancamentos:
```

```
            processados += 1
            valorLcto = float(lcto['valorTransacao'].replace(' ','').replace('.','').replace(',','.'))
            # Pagamentos sao positivos, devoluções negativas. Queremos contabilizar apenas pagamentos, então descartamos as devoluções
            if valorLcto > 0:
                qtdPagamentos += 1
                if valorLcto > maiorPagamento:
                    # Atualiza dados do maior pagamento realizado
                    portador = lcto['portador']
                    respMaiorPagamento = portador['nome']
                    maiorPagamento = valorLcto
        if processados > 0:
            pagina += 1
        else:
            pagina = 0
print()
print('Total de pagamentos realizados no período', qtdPagamentos)
print('Maior pagamento realizado: {}', maiorPagamento)
print('Responsável pelo pagamento: ', respMaiorPagamento)
```

```
1,11,21,31,41,51,61,71,81,91,101,111,121,131,141,151,161,171,181,191,201,211,221,231,241,251,261,271,281,291,301,311,321,331,341,351,361,371,381,391,401,411,4
21,431,441,451,461,471,481,491,501,511,521,531,541,551,561,571,581,591,601,611,621,631,641,651,661,671,681,691,701,711,721,731,741,751,761,771,781,791,801,81
1,821,831,841,851,861,871,881,891,901,911,921,931,941,951,961,971,981,991,1001,1011,1021,1031,1041,1051,1061,1071,1081,1091,1101,1111,1121,1131,1141,1151,116
1,1171,1181,1191,1201,1211,1221,1231,1241,1251,1261,1271,1281,1291,1301,1311,1321,1331,1341,1351,1361,1371,1381,1391,1401,1411,1421,1431,1441,1451,1461,1471,1
481,1491,1501,1511,1521,1531,1541,1551,1561,1571,1581,1591,1601,1611,1621,1631,1641,1651,1661,1671,1681,1691,1701,1711,1721,1731,1741,1751,1761,1771,1781,179
1,1801,1811,1821,1831,1841,1851,1861,1871,1881,1891,1901,1911,1921,1931,1941,1951,1961,1971,1981,1991,2001,2011,2021,2031,2041,2051,2061,2071,2081,2091,2101,2
111,2121,2131,2141,2151,2161,2171,2181,2191,2201,2211,2221,2231,2241,2251,2261,2271,2281,2291,2301,2311,2321,2331,2341,2351,2361,2371,2381,2391,2401,2411,242
1,2431,2441,2451,2461,2471,2481,2491,2501,2511,2521,2531,2541,2551,2561,2571,2581,2591,2601,2611,2621,2631,2641,2651,2661,2671,2681,2691,2701,2711,2721,2731,2
741,2751,2761,2771,2781,2791,2801,2811,2821,2831,2841,2851,2861,2871,2881,2891,2901,2911,2921,2931,2941,2951,2961,2971,2981,2991,3001,3011,3021,3031,3041,305
1,3061,3071,3081,3091,3101,3111,3121,3131,3141,3151,3161,3171,3181,3191,3201,3211,3221,3231,3241,3251,3261,3271,3281,3291,3301,3311,3321,3331,3341,3351,3361,3
371,3381,3391,3401,3411,3421,3431,3441,3451,3461,3471,3481,3491,3501,3511,3521,3531,3541,3551,3561,3571,3581,3591,3601,
Total de pagamentos realizados no período 53993
Maior pagamento realizado: {.2f} 187572.71
Responsável pelo pagamento:  LUIZ EDUARDO MACHADO
```

15

Para os desafios 1 e 2, vamos utilizar uma API do Governo Federal para analisar os gastos por meio de cartão de pagamento. O link da API é:

"http://www.transparencia.gov.br/api-de-dados/cartoes?mesExtratoInicio={}%2F{}&mesExtratoFim={}%2F{}&pagina={}".format(mes_ini, ano_ini, mes_fim, ano_fim, pagina)

onde os dados retornados correspondem a um intervalo definido por um mês e ano inicial (mes_ini e ano_ini) e um mês e ano final (mes_fim, ano_fim). São apresentados 14 dados por página, com a primeira página sendo definida por pagina = 1.

Desafio 2 - Imprima uma tabela com nome do portador e valor da transação para todos os pagamentos por meio de cartão realizados entre 11/2017 e 12/2017.

In [10]:
```
import requests
import time

urlAPI = 'http://api.portaldatransparencia.gov.br/api-de-dados/cartoes?mesExtratoFim={2}%2F{3}&mesExtratoInicio={0}%2F{1}&pagina={4}'
```

```
chaveAPI = '<chave_pessoal>'

# Função que simplifica tratamento de requisições
def reqJson(url):
    # Pausa para não sobrecarregar o serviço, que possui limitação de atendimento de 90 requests por minuto
    time.sleep(0.15)
    resposta = requests.get(url, headers={'chave-api-dados':chaveAPI})
    status = resposta.status_code
    if status == 200:
        descrStatus = ''
        dadosJson = resposta.json()
    else:
        descrStatus = 'Algo deu errado. Erro {}'.format(status)
        dadosJson = None
    return (status, descrStatus, dadosJson)

# Periodo a consultar
mesInicial = '06'
anoInicial = '2018'
mesFinal   = '07'
anoFinal   = '2018'
# Contagem de pagamentos
qtdPagamentos = 0
vlrTotalPagamentos = 0.0

# Processar todas as páginas
pagina = 1
print('Nome do Portador'.ljust(60)+'Valor'.rjust(20))
while pagina > 0:
    resposta = reqJson(urlAPI.format(mesInicial, anoInicial, mesFinal, anoFinal, pagina))
    if resposta[0] != 200:
        # Fim da pesquisa
        pagina = 0
    else:
        # Obtem lista de lançamentos da página
        lancamentos = resposta[2]
        processados = 0
        for lcto in lancamentos:
            processados += 1
            valorLcto = float(lcto['valorTransacao'].replace(' ','').replace('.','').replace(',','.'))
            # Pagamentos sao positivos, devoluções negativas. Queremos contabilizar apenas pagamentos, então descartamos as devoluções
            if valorLcto > 0:
                portador = lcto['portador']
                responsavel = portador['nome']
                print('{}{}'.format(responsavel.ljust(60), str(valorLcto).replace('.',',').rjust(20)))
                qtdPagamentos += 1
                vlrTotalPagamentos += valorLcto
        if processados > 0 and pagina < 2:
            pagina += 1
        else:
            pagina = 0
```

```
print('{0} pagamentos totalizando {1:.2f}'.format(qtdPagamentos, vlrTotalPagamentos))
```

```
Nome do Portador                                        Valor
BRUNO STOCO DE OLIVEIRA                                 360,0
LEANDRO CARVALHO DOS SANTOS                             130,0
ERALDO DE CARVALHO NERES                                380,0
JOSE RENATO FERREIRA MANHAES                             56,0
FERNANDO SNOVARESKI BARBOZA                             209,0
MARCOS ROBERTO BERALDO DE OLIVEIRA                       10,0
ELIANGELO CAVALCANTE SOUZA                              179,0
ROSEMAR FERNANDES FERRUGEM JUNIOR                       98,44
IRAIDES JACINTO                                          40,0
JULIO CESAR RODRIGUES                                   76,82
CARLOS ALBERTO MONTEIRO DE ALMEIDA                       70,0
RUBENS CLAUDIO VELASCO DE ALMEIDA                       500,0
ANTONIO CARDOSO DE MOURA                               100,03
ANA LIVIA KASSEBOEHMER                                 110,69
ANA LIVIA KASSEBOEHMER                                   40,0
CHRISTIANO BETTEGA BRAUNERT                              64,8
REINALDO LEAO DE MORAES FILHO                           200,0
REINALDO LEAO DE MORAES FILHO                           210,0
REINALDO LEAO DE MORAES FILHO                           555,0
MARIA DO CARMO COSTA SILVA                             164,92
AUGUSTO MENDONCA DA COSTA                                96,0
JOSE ANTONIO PINTO                                      170,0
CARLA PATRICIA RODRIGUES RIBEIRO                        45,99
LILIANE APARECIDA DA SILVA                              469,6
SANDRA MARIA DA SILVA FREIRES                           235,0
ARINEIDE BARRETO CARNEIRO                                18,0
NERO TORRES NETO                                        100,0
MAX ROBERTO DE ALMEIDA                                  150,0
AMILTON MARINHO MACHADO                                 47,88
CARLOS ROBERTO ALMEIDA VICENTE                          445,0
30 pagamentos totalizando 5332.17
```

In [ ]: