DESNEVOLVIMENTO PARA INTERNET

DANIEL CORRÊA DA SILVA

2015-1

CAPÍTULO 1

PROGRAMAÇÃO EM REDES UTILIZANDO PYTHON

PROGRAMAÇÃO EM REDES

4 .1 – ENDEREÇAMENTO IP:

PROBLEMA: gerar um intervalo de endereços de rede CIDR, como por exemplo: 123.45.67.89/27.

- **ipaddress:** gera IPv4 e IPv6
- **socket.gethostbyname:** obtem o endereço IP conforme o domínio.
- **geocoder:** encontrar a latitude e longitude do endereço.

```
1  #!/usr/local/bin/python3
2  #
3  #   Arquivo: endIP.py
4  #   Objetivo: programa python que gera um intervalo de
5  #             endereco IP CID.
6  #++++++++++++++++++++++++++++++++++++++++++++++++++++++++
7
8  import ipaddress
9
10 net = ipaddress.ip_network('123.45.67.64/27')
11 net6 = ipaddress.ip_network('12:3456:78:90ab:cd:ef01:23:30/125')
12
13 for ip4 in net:
14     print(ip4)
15 print('++++++++++++++++++++++++++\nIPv6')
16 for ip6 in net6:
17     print(ip6)
18
19 ad = ipaddress.ip_address('123.45.67.95')
20 ad in net # verifica se o endereco 123.45.67.95 está na lista
```

PROGRAMAÇÃO EM REDES

4 .1 – ENDEREÇAMENTO IP:

PROBLEMA: encontra a localização geográfica do endereço (latitude e longitude), bem como devolver o endereço ip de um domínio.

- **ipaddress:** gera IPv4 e IPv6
- **socket.gethostbyname:** obtem o endereço IP conforme o domínio.
- **geocoder:** encontrar a latitude e longitude do endereço.

```
(env01) sh-3.2# pip install pygeocoder
```

```python
#!/usr/local/bin/python3
#
#   Arquivo: geoCode.py
#   Objetivo: programa python que retorna a latitude e
#             longitude do endereço.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++

from pygeocoder import Geocoder
import requests

address = '207 N. Defiance St, Archbold, OH'
print(Geocoder.geocode(address)[0].coordinates)

# buscando documento JSON na API Geocoding do Google
parameters = {'address': address,'sensor':'false'}
base = 'http://maps.googleapis.com/maps/api/geocode/json'
response = requests.get(base,params=parameters)
answer = response.json()
print(answer['results'][0]['geometry']['location'])
```

PROGRAMAÇÃO EM REDES

4 .1 – INTERAGINDO COM UM SERVIÇO HTTP COMO UM CLIENTE:

Definição: acessar serviços HTTP, realizando requisições HTTP com um host cliente utilizando GET.

```python
#!/usr/local/bin/python3
#
# Arquivo: clienteHTTP.py
# Objetivo: criar um cliente que realiza alguma solicitação
#           ao servidor HTTP.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

from urllib import request, parse

url = 'http://httpbin.org/get'

# Dicionário de parâmentros da consulta caso exista
parms = {'parametro1':'value1','paramentro2':'value2'}

# Codifica a string de consulta
queryString = parse.urlencode(parms)

# Faz uma solicitação GET e lê a resposta
u = request.urlopen(url + '?' + queryString)
resposta = u.read()
```

PROGRAMAÇÃO EM REDES

4 .1 – INTERAGINDO COM UM SERVIÇO HTTP COMO UM CLIENTE:

HTTP com Head: acessar serviços HTTP, realizando requisições HTTP com um host cliente utilizando POST.

```python
#!/usr/local/bin/python3
#
# Arquivo: clienteHTTP.py
# Objetivo: criar um cliente que realiza alguma solicitação
#           ao servidor HTTP utilizando POST e HEADERS.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

from urllib import request, parse
import requests

url = 'http://httpbin.org/post'

# Dicionário de parâmentros da consulta caso exista
parms = {'parametro1':'value1','paramentro2':'value2'}
headers = {'User-agent':'none/ofyourbusiness','Span':'Eggs'}

# Codifica a string de consulta
queryString = parse.urlencode(parms)

# Faz uma solicitação POST e lê a resposta
u = request.urlopen(url, queryString.encode('ascii'))
resp = request.post(url,data=parms,headers=headers)
resp.text;
```

PROGRAMAÇÃO EM REDES

4 .1 – INTERAGINDO COM UM SERVIÇO HTTP COMO UM CLIENTE:

HTTP com Head: acessar serviços HTTP, realizando requisições HTTP com um host cliente utilizando POST e Head.

```python
#!/usr/local/bin/python3
#
# Arquivo: clienteHTTP.py
# Objetivo: criar um cliente que realiza alguma solicitação
#           ao servidor HTTP utilizando POST e HEADERS.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

import requests

url = 'http://httpbin.org/post'

# Dicionário de parâmentros da consulta caso exista
parms = {'parametro1':'value1','paramentro2':'value2'}
headers = {'User-agent':'none/ofyourbusiness','Span':'Eggs'}

# Faz uma solicitação POST e lê a resposta
resp = requests.post(url,data=parms,headers=headers)
print(resp.text)
```

PROGRAMAÇÃO EM REDES

4 .1 – INTERAGINDO COM UM SERVIÇO HTTP COMO UM CLIENTE:

HTTP com Head: acessar serviços HTTP, realizando requisições HTTP com um host cliente utilizando POST e Head. Acessar campos específicos do header.

```python
#!/usr/local/bin/python3
#
# Arquivo: clienteHTTPinfHeader.py
# Objetivo: criar um cliente que realiza alguma solicitação
#           ao servidor HTTP e obter informações do HEADER.
#++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

import requests
import yaml

url = 'http://www.google.com.br'

resp = requests.head(url)

status = resp.status_code
content_type = resp.headers['content-type']
content_length = resp.headers['content-length']

print(status)
print(content_type)
print(content_length)

#print(yaml.dump(resp.headers))
```

PROGRAMAÇÃO EM REDES

4 .1 – INTERAGINDO COM UM SERVIÇO HTTP COMO UM CLIENTE:

HTTP com Head: acessar serviços HTTP, realizando requisições HTTP com um host cliente utilizando POST e Login.

```
1  #!/usr/local/bin/python3
2  #
3  # Arquivo: clienteHTTPinfHeader.py
4  # Objetivo: criar um cliente que realiza alguma solicitação
5  #           ao servidor HTTP para fazer login.
6  #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
7
8  import requests
9  import yaml
10
11 url = 'https://pypi.python.org/pypi?%3Aaction=login_form'
12 resp1 = requests.get(url,auth=('danielcs80','Senai12345'))
13 #resp2 = requests.get(url,coockies= resp1.coockies)
```

PROGRAMAÇÃO EM REDES

4 .2 – CLIENTE-SERVIDOR TCP:

MÁQUINA SERVIDOR:

- **socket.soket():** Cria um novo soquete usando a família de endereços, o tipo de soquete e o número de protocolo fornecidos.
- **socket.bind():** vincula o soquete ao endereço. O soquete não deve estar vinculado a outro endereço.
- **socket.listen():** fica ouvindo as conexões feitas no soquete.
- **socket.acept():** aceita conexão.
- **socket.recv():** recebe dados do socket.
- **socket.close():** fecha a conexão com o socket

```
1  #!/usr/local/bin/python3
2  #
3  #   Arquivo: servidorTCP.py
4  #   Objetivo: programa python servidor que aceita
5  #             conecxão dos HOSTs clientes.
6  #++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
7
8  import socket
9  HOST = '127.0.0.1'     # Endereco IP do Servidor
10 PORT = 5000            # Porta que o Servidor esta
11 tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
12 orig = (HOST, PORT)
13 tcp.bind(orig)
14 tcp.listen(1)
15 while True:
16   con, cliente = tcp.accept()
17   print ("Concetado por", cliente)
18   while True:
19       msg = con.recv(1024)
20       if not msg:
21          break
22       print (cliente, msg)
23   print ('Finalizando conexao do cliente', cliente)
24   con.close()
```

PROGRAMAÇÃO EM REDES

4 .2 – CLIENTE-SERVIDOR TCP:

MÁQUINA CLIENTE:

- **socket.soket():** Cria um novo soquete usando a família de endereços, o tipo de soquete e o número de protocolo fornecidos.
- **socket.acept():** aceita conexão.
- **socket.recv():** recebe dados do socket.
- **socket.close():** fecha a conexão com o socket

```
1  #!/usr/local/bin/python3
2  #
3  #   Arquivo: clienteTCP.py
4  #   Objetivo: programa python cliente que se conecta
5  #             ao servidor localizado em algum HOST.
6  #++++++++++++++++++++++++++++++++++++++++++++++++++++++
7
8  import socket
9  import sys
10
11 HOST = '127.0.0.1'      # Endereco IP do Servidor
12 PORT = 5000             # Porta que o Servidor esta
13 tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
14 dest = (HOST, PORT)
15 tcp.connect(dest)
16 print ("Para sair use CTRL+X")
17 msg = input()
18 while msg != '\x18':
19       tcp.send(msg.encode('utf-8'))
20       msg = input()
21 tcp.close()
```