Cybersecurity 701

Obfuscation Lab

Obfuscation Materials

- Materials needed
  - Kali Linux Machine

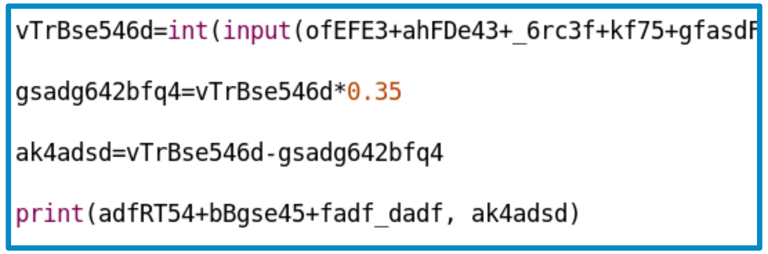
- Software tool used (from Kali Linux)
  - Python (Programming language)
    - This PowerPoint will be using Python3
  - Leafpad
    - Free text editor

Objectives Covered

- Security+ Objectives (SY0-701)
  - Objective 1.4 – Explain the importance of using appropriate cryptographic solutions.
    - Obfuscation
      - Data Masking

What is Obfuscation?

- Obfuscation is taking something simple and making it unclear or obscure
  - Take easy to understand code and make it very confusing
  - Still performs the same task
- How does this help with security?
  - Makes it more difficult for attackers to read the code
    - Not impossible, just more difficult

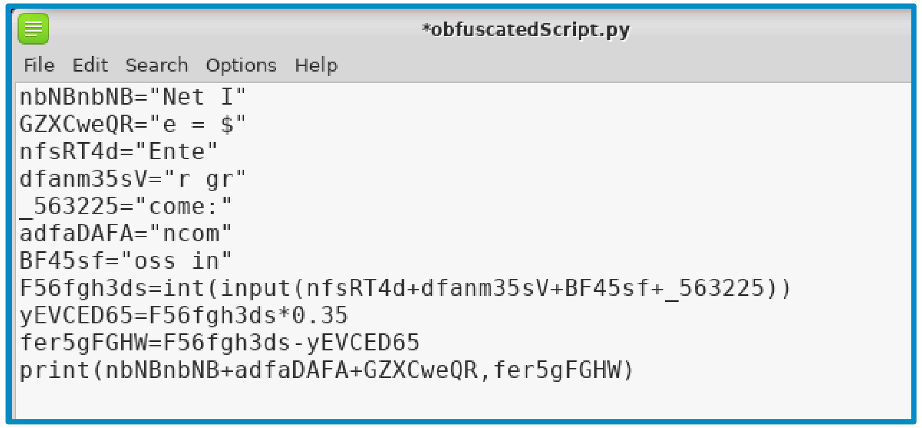


```
vTrBse546d=int(input(ofEFE3+ahFDe43+_6rc3f+kf75+gfasdF

gsadg642bfq4=vTrBse546d*0.35

ak4adsd=vTrBse546d-gsadg642bfq4

print(adfRT54+bBgse45+fadf_dadf, ak4adsd)
```

Simple code made to look confusing

Obfuscation Lab Overview

1. Set up Environment
2. Write Python Script
3. Analyze the Script
4. Copy the Script
5. Obfuscate the Script
6. Run the Obfuscated Script
7. Compare the Scripts



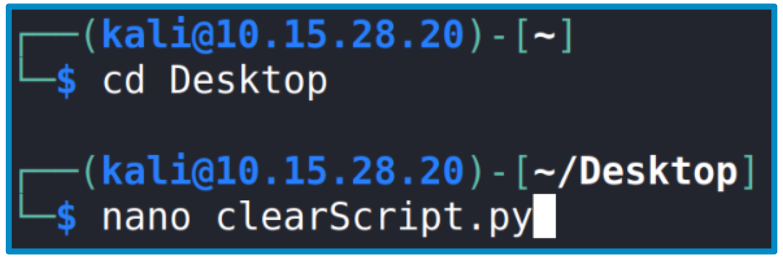
```
nbNBnbNB="Net I"
GZXCweQR="e = $"
nfsRT4d="Ente"
dfanm35sV="r gr"
_563225="come:"
adfaDAFA="ncom"
BF45sf="oss in"
F56fgh3ds=int(input(nfsRT4d+dfanm35sV+BF45sf+_563225))
yEVCED65=F56fgh3ds*0.35
fer5gFGHW=F56fgh3ds-yEVCED65
print(nbNBnbNB+adfaDAFA+GZXCweQR,fer5gFGHW)
```

Set up Environment

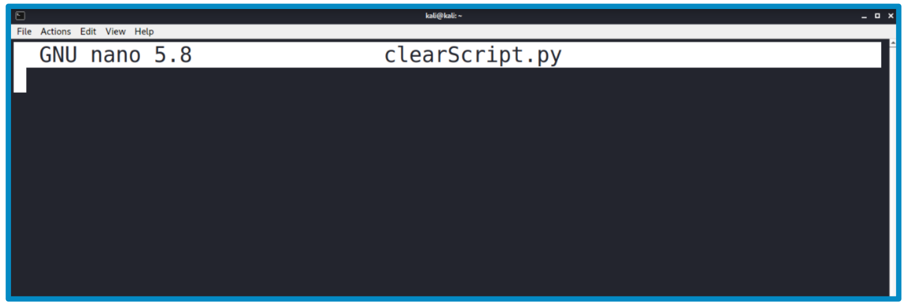
- Log into your range
- Open the Kali Linux Environment
  - You should be on your Kali Linux Desktop

Write Python Script

- Open the Terminal
- Navigate to the Desktop
  - **`cd Desktop`**
- Create a Python script on the desktop (named “clearScript.py”)
  - **`nano clearScript.py`**
    - This should open the nano editor (nothing is saved on the desktop right now)



```
┌──(kali@10.15.28.20)-[~]
└─$ cd Desktop

┌──(kali@10.15.28.20)-[~/Desktop]
└─$ nano clearScript.py
```

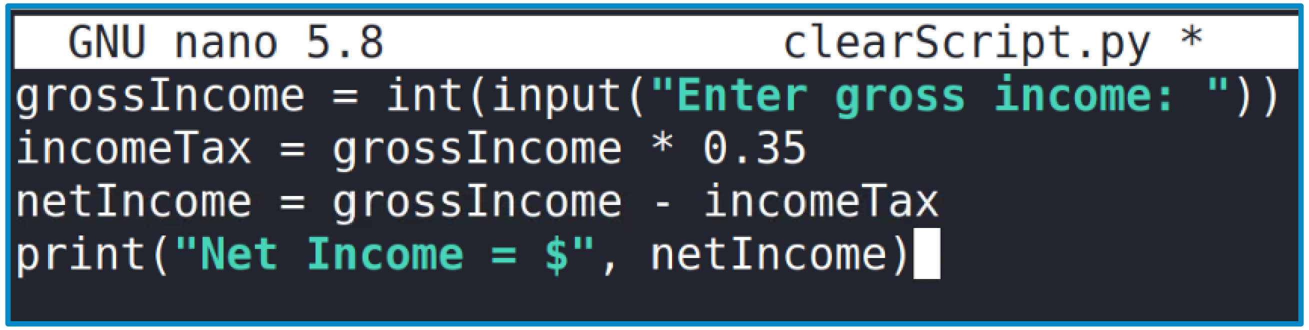


```
  GNU nano 5.8                    clearScript.py
```

Write Python Script

- Write the following Python script:

```
grossIncome = int(input("Enter gross income: "))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print("Net Income = $", netIncome)
```

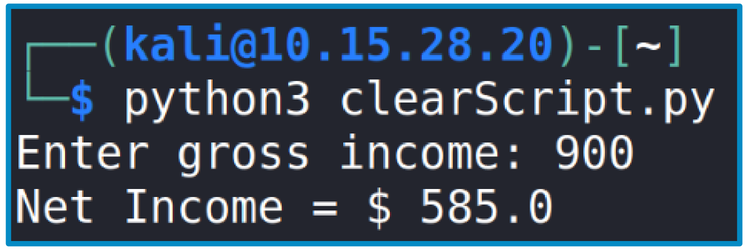


```
  GNU nano 5.8                        clearScript.py *
grossIncome = int(input("Enter gross income: "))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print("Net Income = $", netIncome)
```

Run the Python Script

- Press **`CTRL + x`** to exit the nano editor
- Press **`y`** to save the file/changes
- Press **`<ENTER>`** to save as clearScript.py
- Now run the script*:

  ```
  python3 clearScript.py
  ```
- What is this program doing?
  - What is the tax percentage taken out of the gross income?



```
┌──(kali@10.15.28.20)-[~]
└─$ python3 clearScript.py
Enter gross income: 900
Net Income = $ 585.0
```

*Run the following to install python3 if not installed:

```
sudo apt-get install python3
```

Analyze the Script

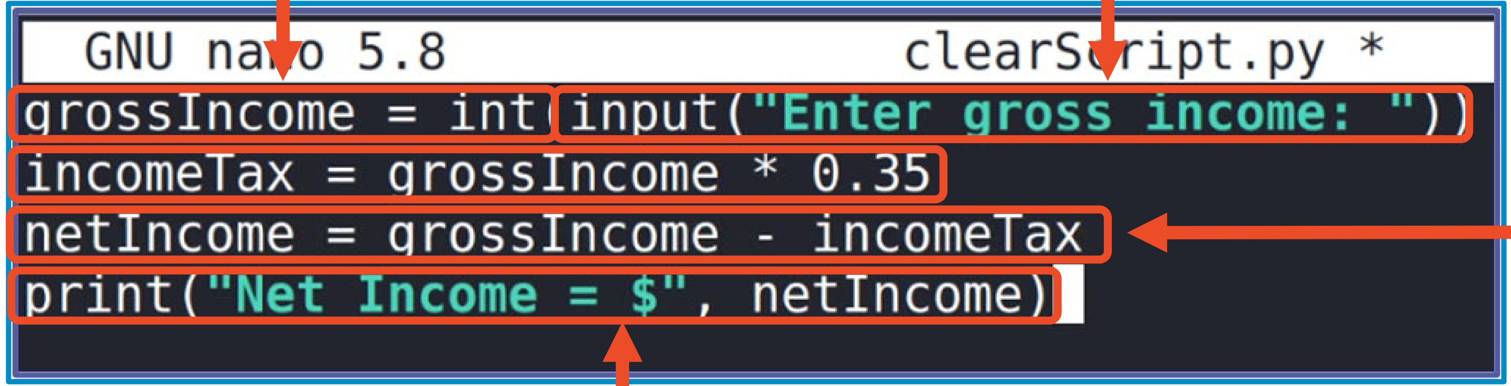
- What is the script doing?

Stores the user's input as an integer for the variable "grossIncome"

Asking the user for an input with the prompt "Enter gross income: "

Variable "incomeTax" is calculated by multiplying grossIncome by 35%

Variable netIncome found by subtracting incomeTax from grossIncome



```
  GNU nano 5.8                          clearScript.py *
grossIncome = int(input("Enter gross income: "))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print("Net Income = $", netIncome)
```

Finally, it prints back to the user "Net Income = $" and the integer netIncome

Notice how easy it is to read this script!

Copy the Script

- In the Terminal, let's copy the script as "obfuscatedScript.py"

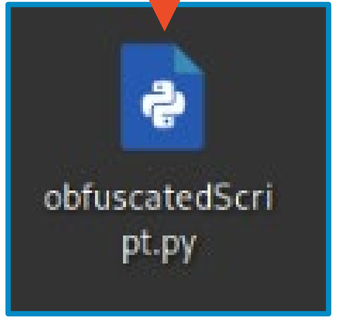
```
cp clearScript.py obfuscatedScript.py
```

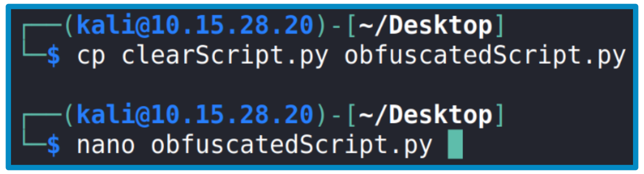
  - You should see the new obfuscatedScript.py file appear on the desktop

- Open the new script in the nano editor

```
nano obfuscatedScript.py
```

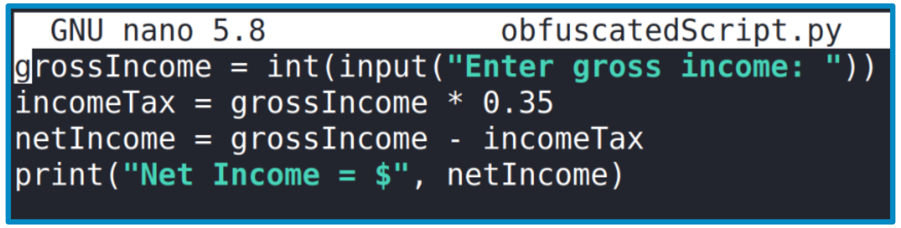
File on the desktop





```
┌──(kali@10.15.28.20)-[~/Desktop]
└─$ cp clearScript.py obfuscatedScript.py

┌──(kali@10.15.28.20)-[~/Desktop]
└─$ nano obfuscatedScript.py
```

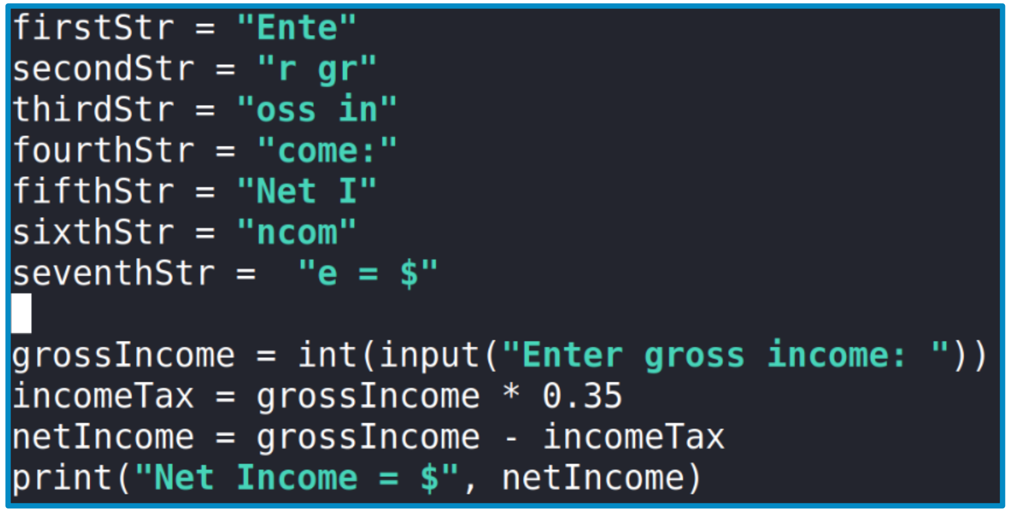
File in the nano editor



```
  GNU nano 5.8                  obfuscatedScript.py
grossIncome = int(input("Enter gross income: "))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print("Net Income = $", netIncome)
```

Obfuscate the Script

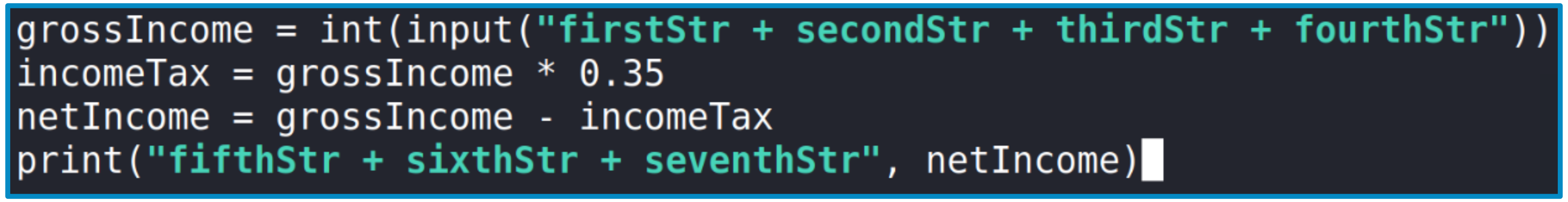
- First, let’s break up the strings
- Make 7 variables at the top and assign the strings
  - firstStr = “Ente”
  - secondStr = “r gr”
  - thirdStr = “oss in”
  - fourthStr = “come:”
  - fifthStr = “Net I”
  - sixthStr = “ncom”
  - seventhStr = “e = $”



```
firstStr = "Ente"
secondStr = "r gr"
thirdStr = "oss in"
fourthStr = "come:"
fifthStr = "Net I"
sixthStr = "ncom"
seventhStr =  "e = $"

grossIncome = int(input("Enter gross income: "))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print("Net Income = $", netIncome)
```

Obfuscate the Script – Variables

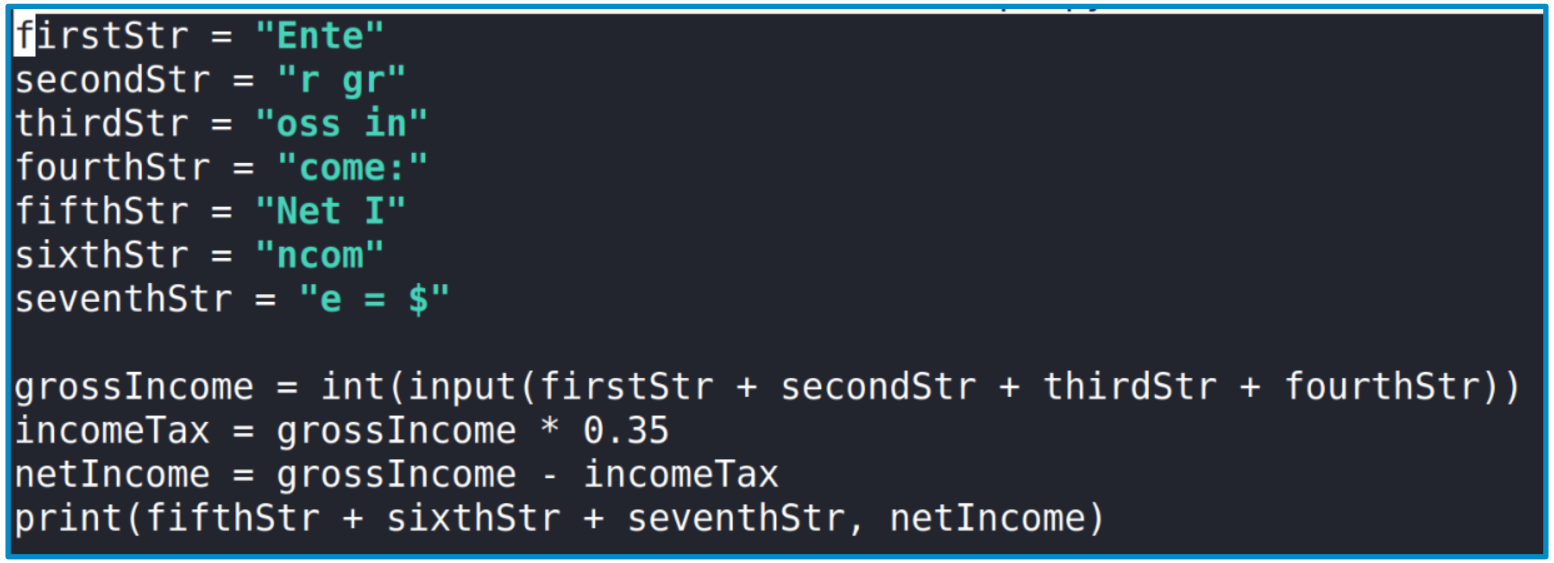
- Replace the strings with the variables
  - “Enter gross income: ” becomes the following:
    - firstStr + secondStr + thirdStr + fourthStr
  - “Net Income = $” becomes the following:
    - fifthStr + sixthStr + seventhStr



```
grossIncome = int(input("firstStr + secondStr + thirdStr + fourthStr"))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print("fifthStr + sixthStr + seventhStr", netIncome)
```

Obfuscate the Script – Checking

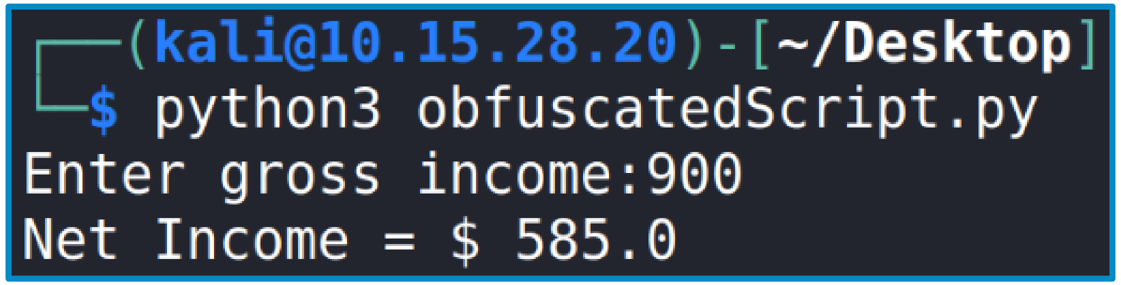
- Check the script



```
firstStr = "Ente"
secondStr = "r gr"
thirdStr = "oss in"
fourthStr = "come:"
fifthStr = "Net I"
sixthStr = "ncom"
seventhStr = "e = $"

grossIncome = int(input(firstStr + secondStr + thirdStr + fourthStr))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print(fifthStr + sixthStr + seventhStr, netIncome)
```

Obfuscate the Script – Still Work?

- Exit out of the nano editor with **CTRL + X**
- Type y to save the changes
- Press **<ENTER>** to keep the name change
- Run the script to make sure it still works
  - **`python3 obfuscatedScript.py`**
- Does it still work?



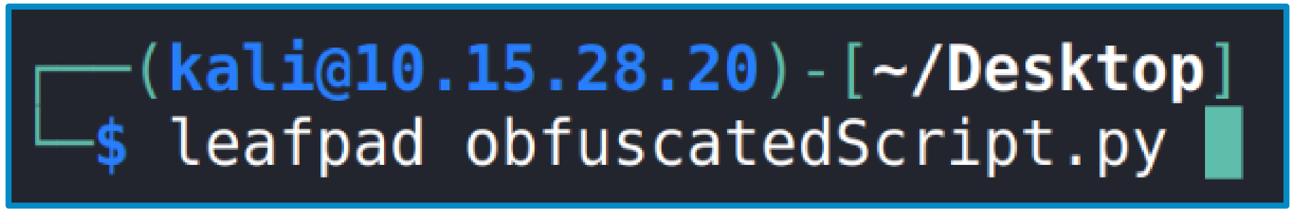
```
┌──(kali@10.15.28.20)-[~/Desktop]
└─$ python3 obfuscatedScript.py
Enter gross income:900
Net Income = $ 585.0
```

Obfuscate the Script – Rename Variables

- Rename the variables, to do this use a text editor that allows a find and replace, like Leafpad
- Open the script in the Leafpad text editor*

  `leafpad obfuscatedScript.py`

  - This should open the script in the Leafpad text editor

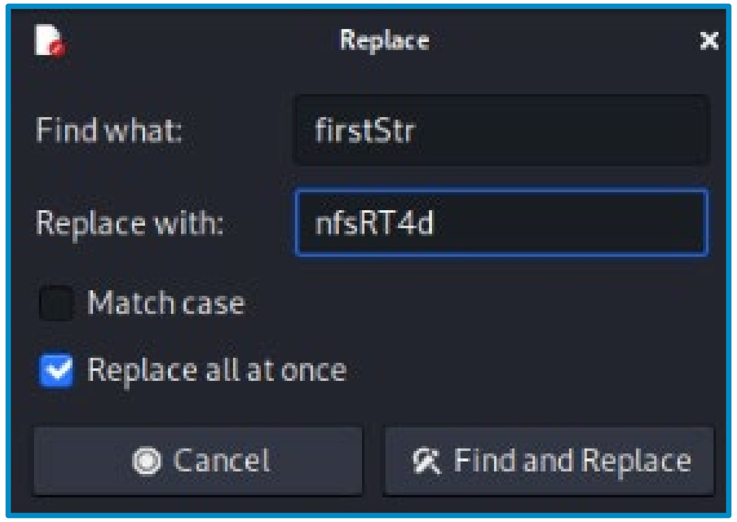


```
┌──(kali@10.15.28.20)-[~/Desktop]
└─$ leafpad obfuscatedScript.py
```

*Run the following to install Leafpad if not updated or installed:
`sudo apt-get install leafpad`

Obfuscate the Script – Search and Replace

- Under the “Search” tab, use the “Replace…” tool
- Replace all the variables with scrambled names
  - Each new name must start with a letter (or the “_” symbol)
- Don’t forget the income variables too!



Find what: firstStr

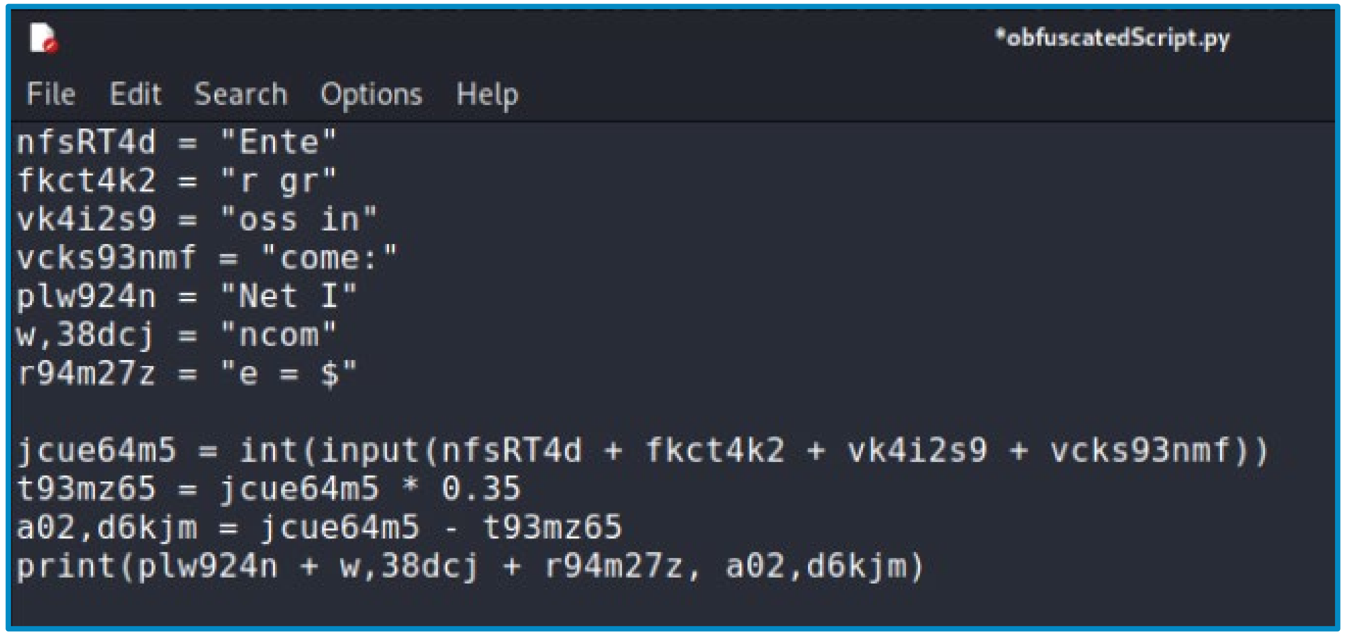
Replace with: nfsRT4d

Match case [ ]

Replace all at once [✓]

Obfuscate the Script – Second Look

- Your code should look something like this

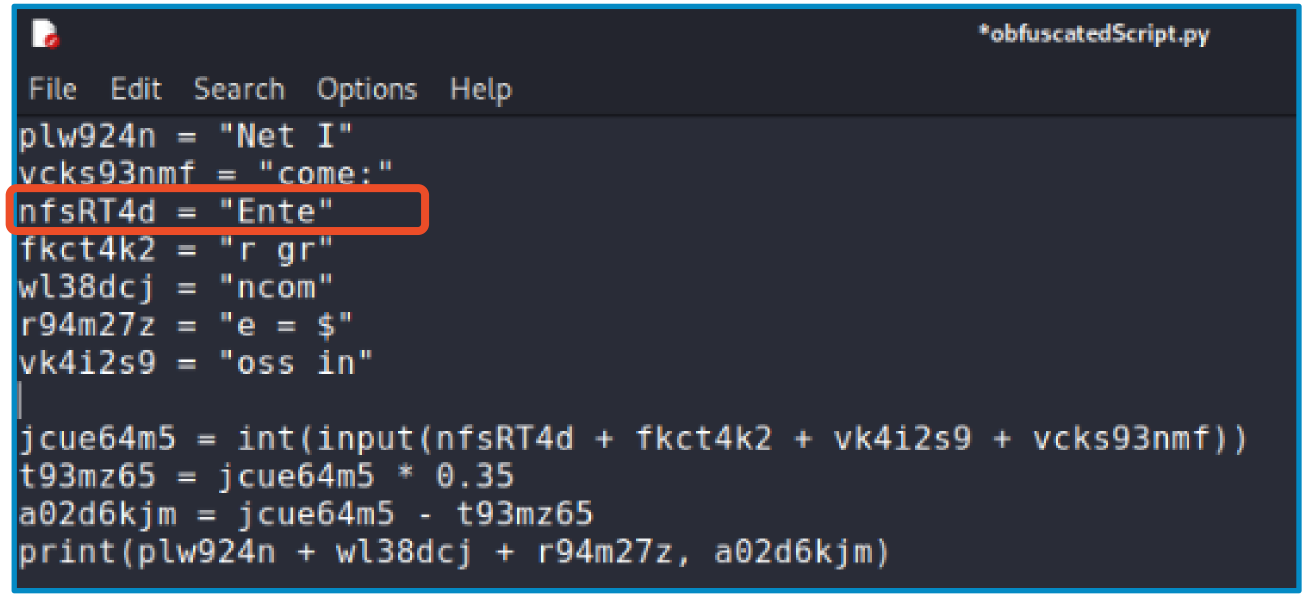


```
nfsRT4d = "Ente"
fkct4k2 = "r gr"
vk4i2s9 = "oss in"
vcks93nmf = "come:"
plw924n = "Net I"
w,38dcj = "ncom"
r94m27z = "e = $"

jcue64m5 = int(input(nfsRT4d + fkct4k2 + vk4i2s9 + vcks93nmf))
t93mz65 = jcue64m5 * 0.35
a02,d6kjm = jcue64m5 - t93mz65
print(plw924n + w,38dcj + r94m27z, a02,d6kjm)
```

Obfuscate the Script – Mix the Strings

- Mix up the top strings

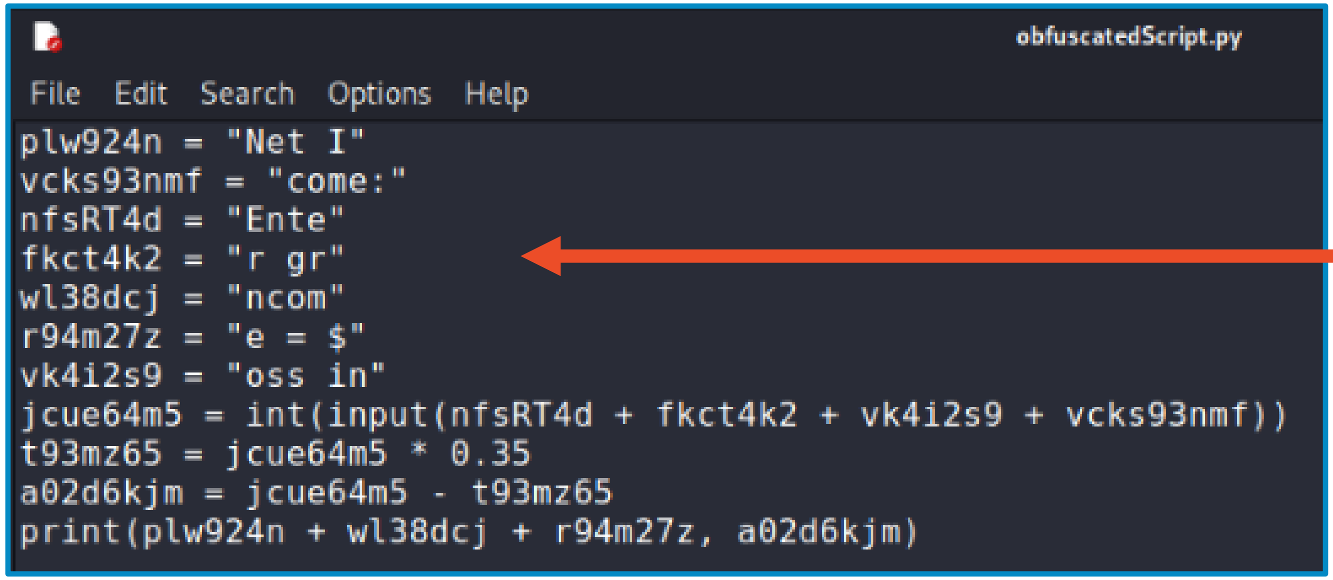
Mix the order of these top strings



```
plw924n = "Net I"
vcks93nmf = "come:"
nfsRT4d = "Ente"
fkct4k2 = "r gr"
wl38dcj = "ncom"
r94m27z = "e = $"
vk4i2s9 = "oss in"

jcue64m5 = int(input(nfsRT4d + fkct4k2 + vk4i2s9 + vcks93nmf))
t93mz65 = jcue64m5 * 0.35
a02d6kjm = jcue64m5 - t93mz65
print(plw924n + wl38dcj + r94m27z, a02d6kjm)
```

Obfuscate the Script – Remove Spaces

- Get rid of all the spacing

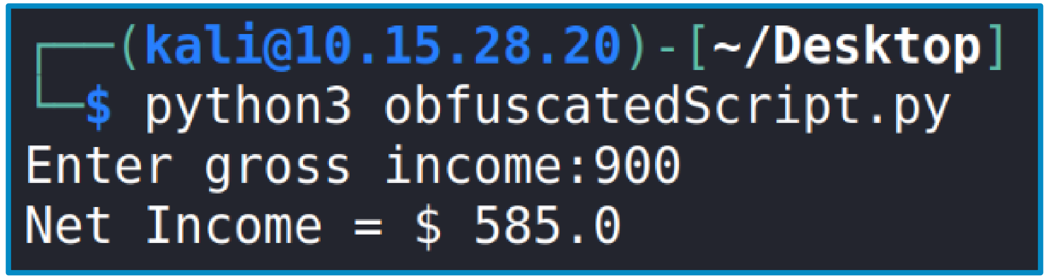


```
plw924n = "Net I"
vcks93nmf = "come:"
nfsRT4d = "Ente"
fkct4k2 = "r gr"
wl38dcj = "ncom"
r94m27z = "e = $"
vk4i2s9 = "oss in"
jcue64m5 = int(input(nfsRT4d + fkct4k2 + vk4i2s9 + vcks93nmf))
t93mz65 = jcue64m5 * 0.35
a02d6kjm = jcue64m5 - t93mz65
print(plw924n + wl38dcj + r94m27z, a02d6kjm)
```

All the spacing has been eliminated (Except for strings inside of quotations)

Run the Obfuscated Script

- Save and exit out of Leafpad
- Test the script
  - python3 obfuscatedScript.py

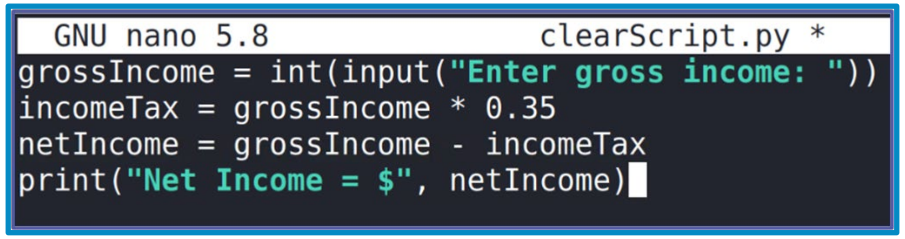


```
┌──(kali@10.15.28.20)-[~/Desktop]
└─$ python3 obfuscatedScript.py
Enter gross income:900
Net Income = $ 585.0
```

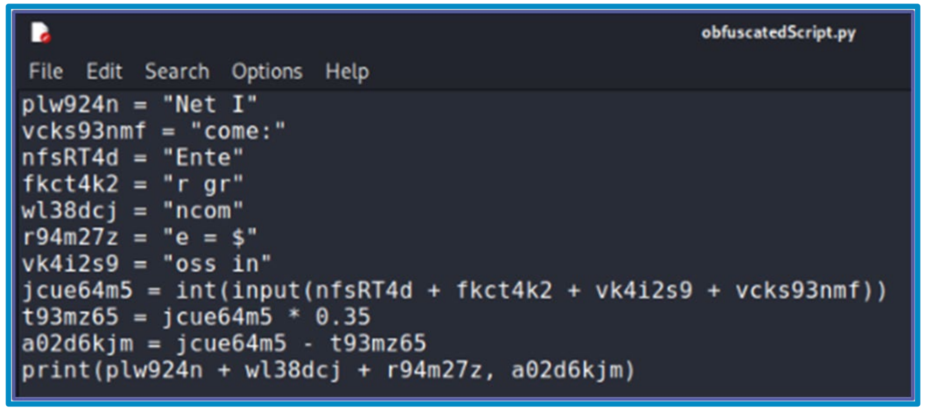
- Notice it still works!

Comparing the Scripts

- Both run the same program and do the same thing in the end?
- How much harder is it to read the obfuscated script?



```
  GNU nano 5.8                     clearScript.py *
grossIncome = int(input("Enter gross income: "))
incomeTax = grossIncome * 0.35
netIncome = grossIncome - incomeTax
print("Net Income = $", netIncome)
```



```
plw924n = "Net I"
vcks93nmf = "come:"
nfsRT4d = "Ente"
fkct4k2 = "r gr"
wl38dcj = "ncom"
r94m27z = "e = $"
vk4i2s9 = "oss in"
jcue64m5 = int(input(nfsRT4d + fkct4k2 + vk4i2s9 + vcks93nmf))
t93mz65 = jcue64m5 * 0.35
a02d6kjm = jcue64m5 - t93mz65
print(plw924n + wl38dcj + r94m27z, a02d6kjm)
```