Lab 3: Notes

finalni kod je na dnu, copy-pasteala sam ga više puta kroz lab. vježbe da se vidi slijed razmišljanja i dodavanja koda + komentari dodataka (novi kod je uvijek podebljan)

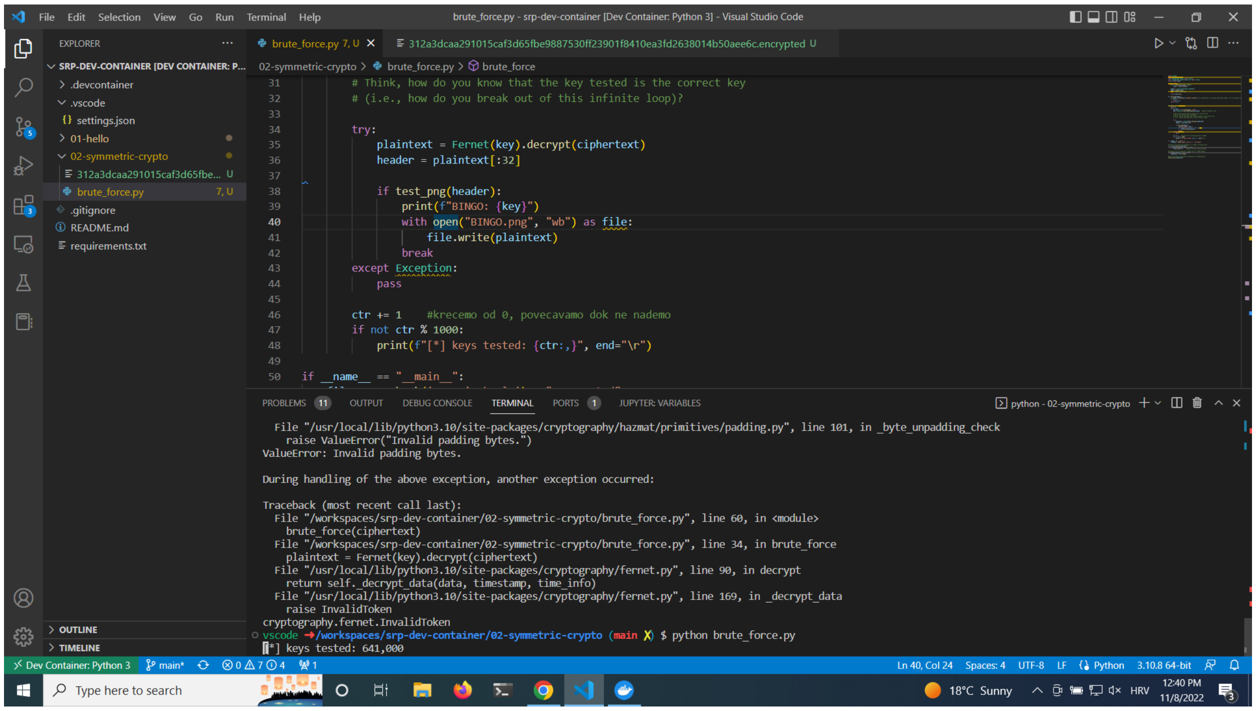
```
from cryptography.hazmat.primitives import hashes

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

filename = hash('pupacic_karla') + ".encrypted"
print(filename)
```

```
from cryptography.hazmat.primitives import hashes
from os import path  #dodano da bi mogli provjeriti postoji li vec dokument kojeg stvaramo

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

if __name__ == "__main__":
    filename = hash('pupacic_karla') + ".encrypted"

#create a file with the filename if it doesn't already exist

if not path.exists(filename):
    with open(filename, "wb") as file:
        file.write(b"")  #prazan string u obliku byteova
```

```
import base64
from cryptography.hazmat.primitives import hashes
from os import path

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

def brute_force():
    ctr = 0
    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)   #import base64 na vrhu
        # Now initialize the Fernet system with the given key
        # and try to decrypt your challenge.
        # Think, how do you know that the key tested is the correct key
        # (i.e., how do you break out of this infinite loop)?

        ctr += 1    #krecemo od 0, povecavamo dok ne nademo
        if not ctr % 1000:
            print(f"[*] keys tested: {ctr:,}", end="\r")

if __name__ == "__main__":
    filename = hash('pupacic_karla') + ".encrypted"

#create a file with the filename if it doesn't already exist
if not path.exists(filename):
    with open(filename, "wb") as file:
        file.write(b"")     #prazan string u obliku byteova
```

```
import base64
from pydoc import plain
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from os import path

def hash(input):
    if not isinstance(input, bytes):
        input = input.encode()

    digest = hashes.Hash(hashes.SHA256())
    digest.update(input)
    hash = digest.finalize()

    return hash.hex()

def test_png(header):
    if header.startswith(b"\211PNG\r\n\032\n"): #u navodnicima je karakteristika PNG formata, sve te datoteke imaju to kao prefiks, pa to t
        return True
    return False


def brute_force(ciphertext):
    ctr = 0
    while True:
        key_bytes = ctr.to_bytes(32, "big")
        key = base64.urlsafe_b64encode(key_bytes)   #import base64 na vrhu

        # Now initialize the Fernet system with the given key
        # and try to decrypt your challenge.
        # Think, how do you know that the key tested is the correct key
        # (i.e., how do you break out of this infinite loop)?

        try:
            plaintext = Fernet(key).decrypt(ciphertext)
            header = plaintext[:32]

            if test_png(header):
                print(f"BINGO: {key}")
                with open("BINGO.png", "wb") as file:
                    file.write(plaintext)
                break
        except Exception:
            pass

        ctr += 1    #krecemo od 0, povecavamo dok ne nademo
        if not ctr % 1000:
            print(f"[*] keys tested: {ctr:,}", end="\r")

if __name__ == "__main__":
    filename = hash('pupacic_karla') + ".encrypted"

#create a file with the filename if it doesn't already exist
if not path.exists(filename):
    with open(filename, "wb") as file:
        file.write(b"")     #prazan string u obliku byteova

#otvaramo file sa izazovom i citamo sto se u njemu nalazi
```

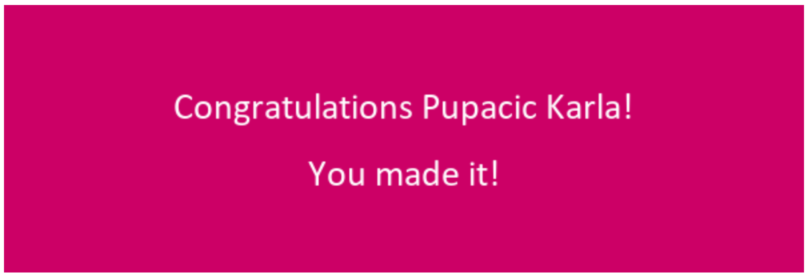
```
with open(filename, "rb") as file:
    ciphertext = file.read()
```



```python
        # Think, how do you know that the key tested is the correct key
        # (i.e., how do you break out of this infinite loop)?

        try:
            plaintext = Fernet(key).decrypt(ciphertext)
            header = plaintext[:32]

            if test_png(header):
                print(f"BINGO: {key}")
                with open("BINGO.png", "wb") as file:
                    file.write(plaintext)
                break
        except Exception:
            pass

        ctr += 1    #krecemo od 0, povecavamo dok ne nademo
        if not ctr % 1000:
            print(f"[*] keys tested: {ctr:,}", end="\r")

if __name__ == "__main__":
```

```
  File "/usr/local/lib/python3.10/site-packages/cryptography/hazmat/primitives/padding.py", line 101, in _byte_unpadding_check
    raise ValueError("Invalid padding bytes.")
ValueError: Invalid padding bytes.

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "/workspaces/srp-dev-container/02-symmetric-crypto/brute_force.py", line 60, in <module>
    brute_force(ciphertext)
  File "/workspaces/srp-dev-container/02-symmetric-crypto/brute_force.py", line 34, in brute_force
    plaintext = Fernet(key).decrypt(ciphertext)
  File "/usr/local/lib/python3.10/site-packages/cryptography/fernet.py", line 90, in decrypt
    return self._decrypt_data(data, timestamp, time_info)
  File "/usr/local/lib/python3.10/site-packages/cryptography/fernet.py", line 169, in _decrypt_data
    raise InvalidToken
cryptography.fernet.InvalidToken
vscode ➜/workspaces/srp-dev-container/02-symmetric-crypto (main ✗) $ python brute_force.py
[*] keys tested: 641,000
```

…vidimo kako se ljučevi testiraju, 1000 po 1000



Congratulations Pupacic Karla!

You made it!