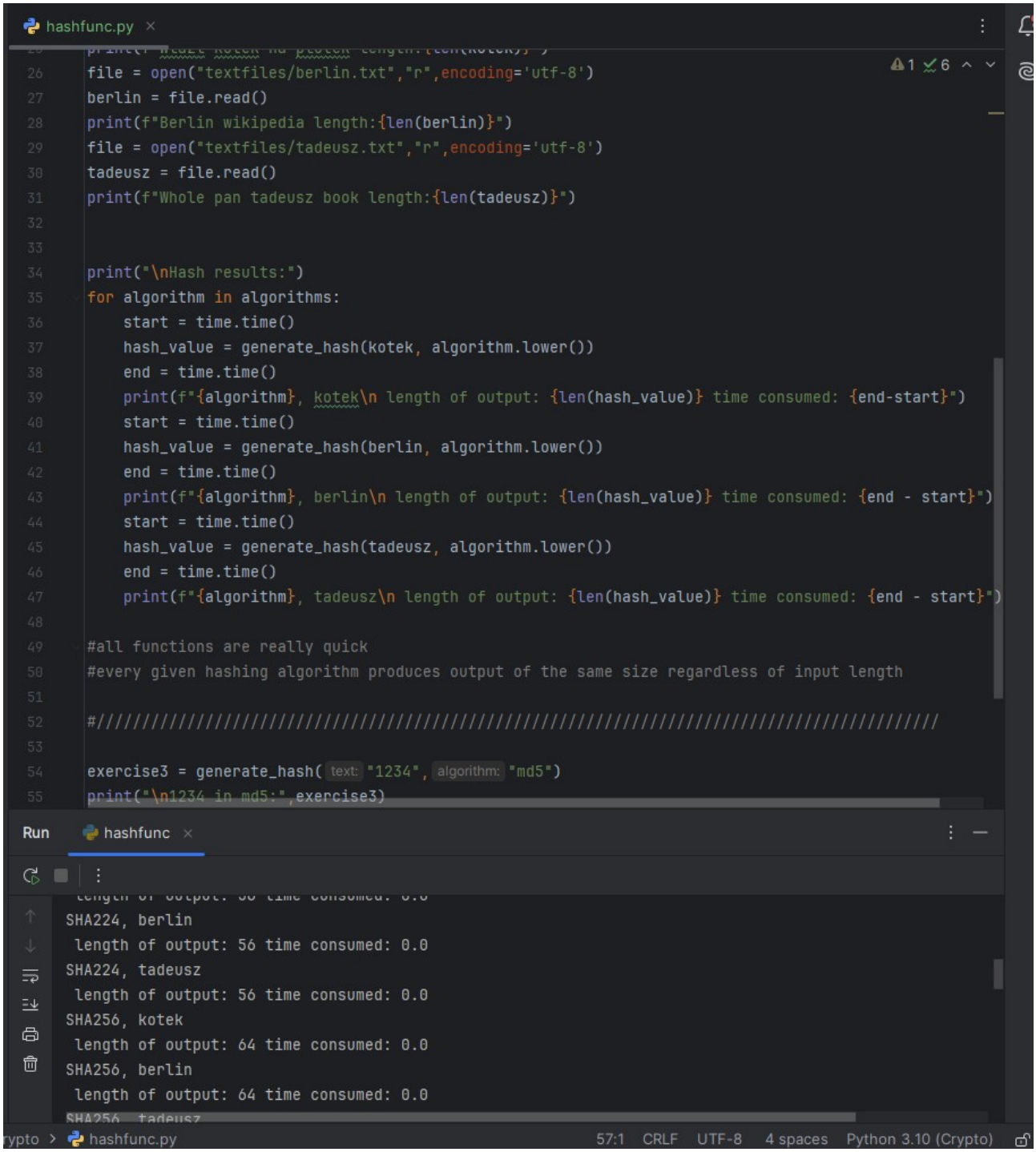
1.



2.uzyłem biblioteki hashlib w pythonie do wygenerowania funkcji skrótu
3. “1234” w md5: 81dc9bdb52d04dc20036dbd8313ed055
poniżej przesyłam link do strony w której możemy znaleźć dane wartości pod danymi hashami
https://md5.gromweb.com/?md5=81dc9bdb52d04dc20036dbd8313ed055
Jak widać nie jest ciężko znaleźć pierwotne hasło, wystarczy jedno wyszukanie w Google.

Użycie soli pozwala uniknąć takiej sytuacji. Przed wygenerowaniem hasha dla naszego hasła dodawane są do niego losowe bity dzięki czemu te same hasła nie wygenerują tego samego hasha!

4. Niestety znalezione zostały całościowe kolizje w MD5 co sprawia, że nie jest to bezpieczna funkcja skrótu. SHA1 też nie jest odpowiednim zastępnikiem bo też ma małą liczbę bitów na wyjsciu, co w połączeniu ze zjawiskiem birthday paradoxu jest całkiem niebezpieczne.

5.

```
Wlazl kotek na plotek length:315
Berlin wikipedia length:1351
Whole pan tadeusz book length:44469
```

← Długość danych wejściowych w ilości znaków

```
print("\nHash results:")
for algorithm in algorithms:
    start = time.time()
    hash_value = generate_hash(kotek, algorithm.lower())
    end = time.time()
    print(f"{algorithm}, kotek\n length of output: {len(hash_value)} time consumed: {end-start}")
    start = time.time()
    hash_value = generate_hash(berlin, algorithm.lower())
    end = time.time()
    print(f"{algorithm}, berlin\n length of output: {len(hash_value)} time consumed: {end - start}")
    start = time.time()
    hash_value = generate_hash(tadeusz, algorithm.lower())
    end = time.time()
    print(f"{algorithm}, tadeusz\n length of output: {len(hash_value)} time consumed: {end - start}")
```

Każdy dany algorytm hashujący zwraca ciąg tylu samu bitów(na screenie widać ilość znaków kodu szestnastkowego) niezależnie od długości ciągu bitów na wejsciu.

```
MD5, kotek
 length of output: 32 time consumed: 0.0
MD5, berlin
 length of output: 32 time consumed: 0.0
MD5, tadeusz
 length of output: 32 time consumed: 0.0
SHA1, kotek
 length of output: 40 time consumed: 0.0
SHA1, berlin
 length of output: 40 time consumed: 0.0
SHA1, tadeusz
 length of output: 40 time consumed: 0.0
SHA224, kotek
 length of output: 56 time consumed: 0.0
SHA224, berlin
 length of output: 56 time consumed: 0.0
SHA224, tadeusz
 length of output: 56 time consumed: 0.0
SHA256, kotek
 length of output: 64 time consumed: 0.0
SHA256, berlin
 length of output: 64 time consumed: 0.0
SHA256, tadeusz
 length of output: 64 time consumed: 0.0
SHA384, kotek
```

Każdy z nich bardzo dobrze sobie radzi pod względem prędkości dla podanej wielkości plików.

```
Partial collision found!
45f1a4b4a2f6bdfa31838a823064389f 45fb9b76950ea7ac85e5e489dab1c654
nepal conduct
```

Szansa na częściową kolizje jest bardzo duża ponieważ 12 bitów a więc 3 znaki hexadecymalne to tylko 2 ^ 12 możliwości więc 4096. Gdyby pobrać cały słownik języka angielskiego nie trudno o te kolizje. Na dole przykład słów dla których występuje częściowa kolizja w MD5

Zmienienie jednego bita w wiadomosci przed zhashowaniem sprawia, że połowa bitów się zmienia.

```
message: give me 1 dollar
01100111011010010111011001100101001000000110110101100101001000000011000100100000011001000110111101101100011011000110000101110010
11100111011010010111011001100101001000000110110101100101001000000011000100100000011001000110111101101100011011000110000101110010
message after changing first bit: çive me 1 dollar
sac1 hash: 10010111111111111001000001110010110111100101100111000001100101010011100110000000001101101010101100000011100010100100110100100
sac2 hash: 10100001011111010010010110111111011000111111000011011110110100101000000010010100000001111110000110000000011001100111110000010
xored hashes: 00110110100000101011010111001100000110011011100001111100001001111011100100010100001100010100101010000011111011000011000100
avalanche test passed
```

Przykład dla “give me 1 dollar” I “çive me 1 dollar” zastosowany dla funkcji skrótu SHA1.