In [1]:
```python
#zad 1
import hashlib
import time

my_set = hashlib.algorithms_available
my_list = list(my_set)

mess = input("Type message to hash: ")
mess_as_bytes = str.encode(mess)

for i in range(len(my_list)):
    print("{}: {}".format(i, my_list[i]))
    start_time = time.time()
    h = hashlib.new(my_list[i])
    h.update(mess_as_bytes)
    if(my_list[i] == 'shake_256' or my_list[i] == 'shake_128'): #needed because of dynamic length
        print(h.hexdigest(32))
    else:
        print(h.hexdigest())

    end_time = time.time()

    print("Time: ", end_time-start_time, "\n")
```

```
Type message to hash: message typed in console
0: ripemd160
fd9f2188a36efbd6476ece6f54f794e143bcbdde
Time:  0.0009999927520751953

1: sha224
8a129a20b8aec69f335cb4c356a42948c50cfa13012a421ba0f1957e
Time:  0.0

2: md5-sha1
3de1680c196ed5fb5c3a9bd130cc4adbab86a158b0e3bbe115ed5089984be6ba52735161
Time:  0.0

3: sha256
039597194eadec4d0d6b231d4d4eb4158cfd03300fb8c4e6660290d952dd308a
Time:  0.0

4: shake_256
e30fb151cbd10d8a894c073f4ebf51a979576c3064be2886cd5a520b06162ad3
Time:  0.0

5: md4
436f5f40f4edbab20faaf33505535092
Time:  0.0

6: mdc2
8d08488c3b84f78877cee94ded04eb59
Time:  0.0

7: sha3_256
b9d5d9be30a600fae12e859080bbaec59176b08b7db45fe7e61e1d0431bb5780
Time:  0.0

8: sha512
89254221066f11a4ff96e6431cbb986f999a9fe0782289335f135fa06d225db8fa5890221afa28c27cf44ee69984c87f634148f5a85b3aa6619d0dfefdef4330
Time:  0.0

9: sm3
36e1c5e3f22ce17579aea6097c11100ad593eac39c34a0c60e88e4734eb80888
Time:  0.0

10: blake2s
a19412119badc26a5b93240b2a55493e274c74ab28ec53469113ca208d4d1509
Time:  0.0

11: blake2b
ef815183acd4bad76c9578d3d4c9c7f021c6ba3cbe3b990b731cc56fbbd925362314cfd5018a28db95a876ee71d10c7f683dc76748f746c63af15d3c62fee0bd
Time:  0.0

12: sha3_384
8b96e231d91b1501bfbff59454e8fc8c9b201475a4f1b909902189526 34c28353f039f1b5a32038773b524aac8b9ec2d
Time:  0.0

13: sha3_512
af2ce826f3dc079a27bd58e8f3f79222eb5788f42aba3473bf6f5f2336cad3867f45ba8fb14d035656f6577bfb5ba521292bf8673024138354bdfe7737ef2b8d
Time:  0.0

14: whirlpool
53bb5ce41aec7984a0b39f7ecac5555555624d1feb8276ddce0debdbf5f937baa74bf3379299ba8bd43ec2f4d5fafb9d286fe9378455b39e1c187d58ce6e8b6b
Time:  0.0

15: sha3_224
f0229c032877d58f44778a0990e87aae7dfa110219a28482b0019913
Time:  0.0

16: sha1
ab86a158b0e3bbe115ed5089984be6ba52735161
Time:  0.0

17: sha512_256
6831c5f4fe2e84de6a310156f4bcdbfbec6f23f330ab5317007ec673ce7c8d65
Time:  0.0

18: shake_128
f8f77c13d2c33382c30e89522afeaf1691969a2133a54c23e5587119cdc9477d
Time:  0.0

19: sha512_224
65eedaa826b1ae066b6670845de440c32dfc01f4126374f2a3ad817b
Time:  0.0

20: md5
3de1680c196ed5fb5c3a9bd130cc4adb
Time:  0.0

21: sha384
bc606b3c4be62396cbb47f7bdf3579c36378af696606ce0207faa844993b2b0b6289de2d7b318b7c198140fa1c3da88f
Time:  0.0
```

In [2]:
```python
#zad 2&3
import hashlib
def hash_file(filename):
    h = hashlib.sha256()
    with open(filename,'rb') as file:
        chunk = 0
        while chunk != b'':
            chunk = file.read(1024)
            h.update(chunk)

    return h.hexdigest()

hashed_message = hash_file(r'E:\Win7\Windows 7 Home Basic SP1 32-Bit.iso') #size ~2.5 GB
```

In [3]:
```python
#zad 2&3
#unit test case
import unittest

class TestStringMethods(unittest.TestCase):
    def test_negative(self):
        firstValue = hashed_message
        secondValue = "D8FA5EA8CF67315FA6CE693EF0C70503DF7E14258301585EBC28EB1C6C8D6216" #taken from Powershell
        message = "First value and second value are not equal !"
        self.assertEqual(firstValue, secondValue, message)

    def test_positive(self):
        firstValue = hashed_message.upper()
        secondValue = "D8FA5EA8CF67315FA6CE693EF0C70503DF7E14258301585EBC28EB1C6C8D6216" #taken from Powershell
        message = "First value and second value are not equal !"
        self.assertEqual(firstValue, secondValue, message)

if __name__ == '__main__':
    unittest.main(argv=['first-arg-is-ignored'], exit=False)
```

```
F.
======================================================================
FAIL: test_negative (__main__.TestStringMethods)
----------------------------------------------------------------------
  File "<ipython-input-3-ead9e75796cd>", line 10, in test_negative
    self.assertEqual(firstValue, secondValue, message)
AssertionError: 'd8fa5ea8cf67315fa6ce693ef0c70503df7e14258301585ebc28eb1c6c8d6216' != 'D8FA5EA8CF67315FA6CE693EF0C70503DF7E14258301585EBC28EB1C6C8D6216'
- d8fa5ea8cf67315fa6ce693ef0c70503df7e14258301585ebc28eb1c6c8d6216
+ D8FA5EA8CF67315FA6CE693EF0C70503DF7E14258301585EBC28EB1C6C8D6216
 : First value and second value are not equal !

----------------------------------------------------------------------
Ran 2 tests in 0.002s

FAILED (failures=1)
```

In [4]:
```python
#zad 4
import hashlib
import timeit
from _md5 import md5 as md5_builtin

bytes_list = []
times_list = []

def time_hash(limit):
    base = b"A"
    number = 1000000
    for counter in range(0,limit):
        factor = 2 ** counter
        text = base * factor
        globals_dict = {"text": text}
        globals_dict["md5"] = md5_builtin
        t = timeit.timeit(stmt = "md5(text)", globals = globals_dict, number = number)
        bytes_list.append(factor)
        times_list.append(t)

time_hash(14)   # 12 is ~8s, 14 is ~30s, 16 is ~2min
```

In [5]:
```python
#zad 4
import pandas as pd
import plotly.express as px

df = pd.DataFrame(
    {'Bytes(b)': bytes_list,
     'Time(s)': times_list})

fig = px.line(df, x="Time(s)", y="Bytes(b)", title = "Time of hashing")
fig.show()
```