One-Way Hash Function

- Goal: to ensure data integrity
- Message -> Hashing Algorithm -> Message Digest
- Message digest: a fingerprint of a piece of data
  - A keyless message digest
- Characteristics of Hash Function H()*
  - One-way: easy to compute but hard to inverse
  - Collision resistant
  - Fixed length:
    - variable-length M -> fixed-length hash
  - Example, MD5,SHA1

The building blocks

- import hashlib
- Syntax for object instantiation

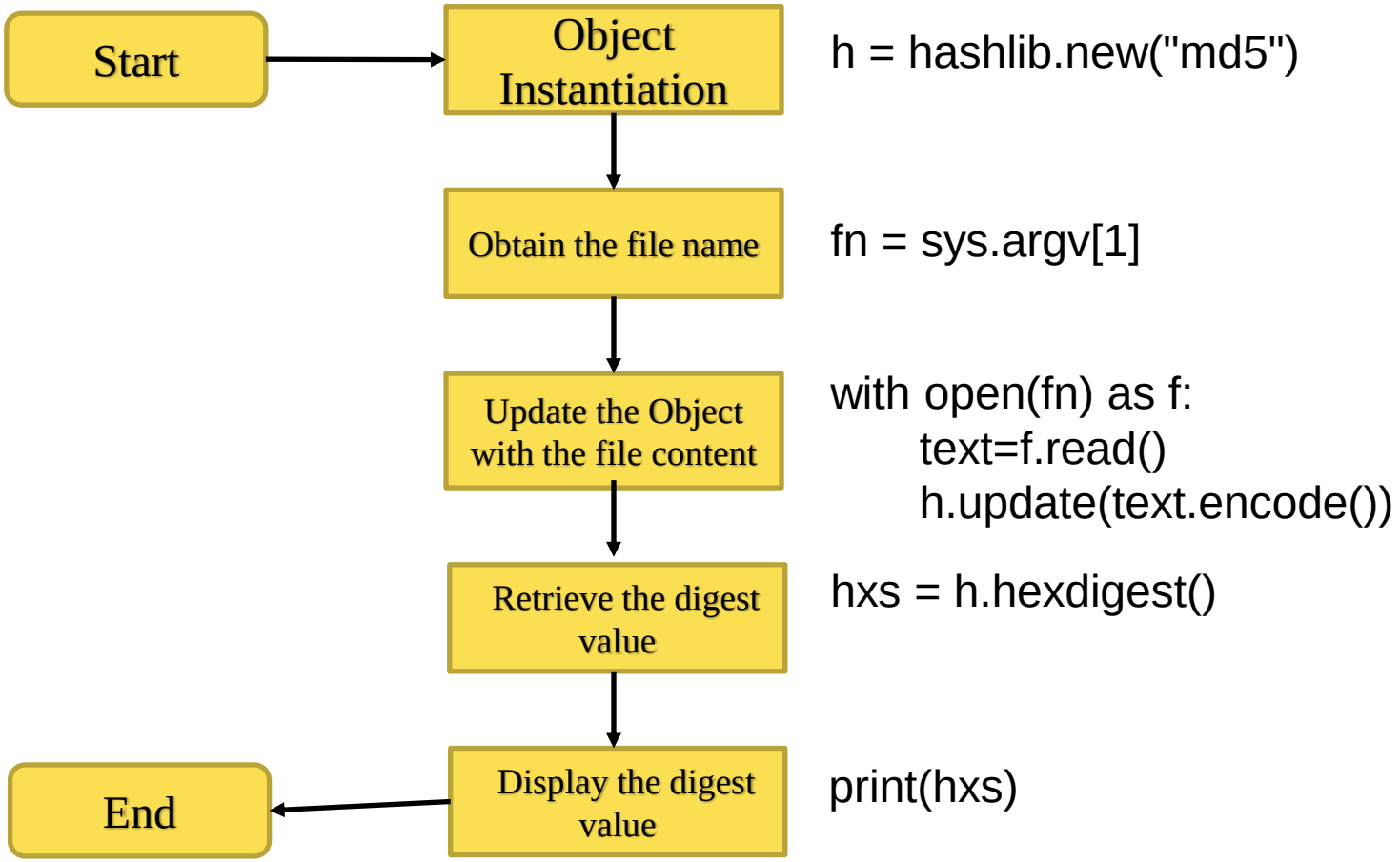
  hobject = hashlib.new(<name of the hash function>)
- Example
  - hash = hashlib.new("md5")
  - hash = hashlib.new("sha1")
- Usage
  - hash.update(message)
    - message is a bytes-like object
    - Repeated calls are equivalent to a single call with the concatenation of all the arguments.
  - hash.digest()
    - Returns message digest in bytes
  - hash.hexdigest()
    - Returns message hexdigest in string type (Representation in hexadecimal digits)
- Reference – for more information
  - https://docs.python.org/3/library/hashlib.html

Demo: One-Way Hash

- Sample output:

```
$md5sum a.txt
82642eccc6df18237de0f228e52de538  a.txt
$./myMd5Stud.py a.txt
A Simple Program on MD5
MD5 Hex => 82642eccc6df18237de0f228e52de538
End of Program
$
```

Flow Chart – Message Digest



Template of myMd5Stud.py

```
#!/usr/bin/env python3
#ST2504 - ACG Practical - myMd5Stud_skel.py
# Template for myMd5Stud.py
import sys
import hashlib
# main program starts here
argc = len(sys.argv)
if argc != 2:
    print("Usage : {0} <file name>".format(sys.argv[0]))
    exit(-1)
try:
    with open(sys.argv[1],"r") as f:
        content = f.read()
        # instantiate your hash object here

        # update the hash object with the file content (in bytes!) here

        # Retrieve and print the hex string of the message digest.
        print("A Simple Program on MD5")
        # insert your code here

        print("End of Program")
    f.close()
except:
    print("Invalid file argument!")
```