Homework 2

- A 4*4 image with 16 pixels
- Borders unaltered

A1	A2	A3	A4
B1	B2	B3	B4
C1	C2	C3	C4
D1	D2	D3	D4

Color of B2 = Average color of (B1,A2,B3,C2)

Swap function

Example: (swap_buggy.py)

```
>>> a = 1
... b = 2
... def swap(t1, t2):
...     t2, t1 = t1, t2
...     return
... swap(a, b)
... print "a=",a
... print "b=",b
a=1
b=2
```

Swap function

Example: swap_right.py

```
>>> a = 1
... b = 2
... def swap(t1, t2):
...     return t2, t1
... a, b = swap(a, b)
... print "a=",a
... print "b=",b
a=2
b=1
```

Cryptography

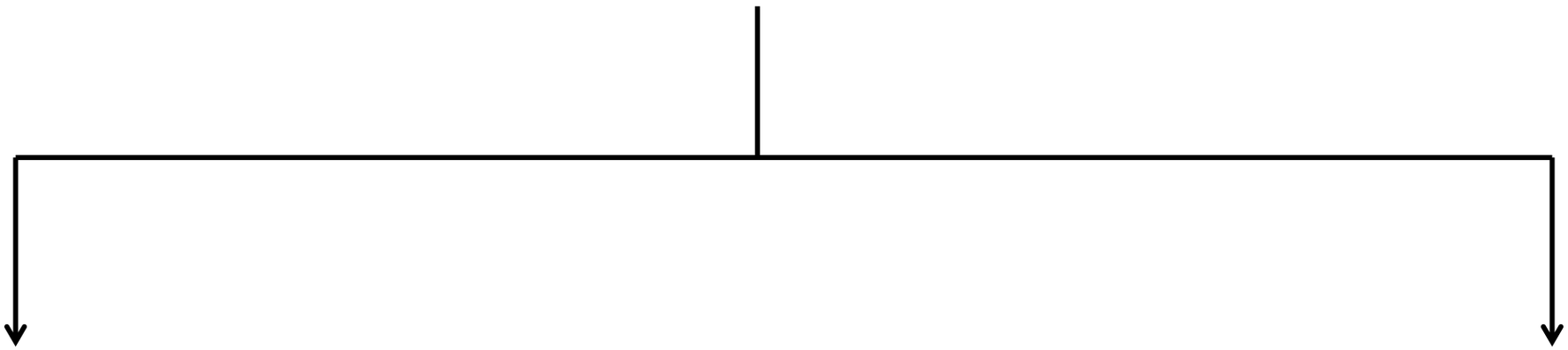
Plaintext – A message in its natural format readable by an attacker

Ciphertext – Message altered to be unreadable by anyone except the intended recipients

Key – Sequence that controls the operation and behavior of the cryptographic algorithm

Keyspace – Total number of possible values of keys in a crypto algorithm

Substitution Ciphers



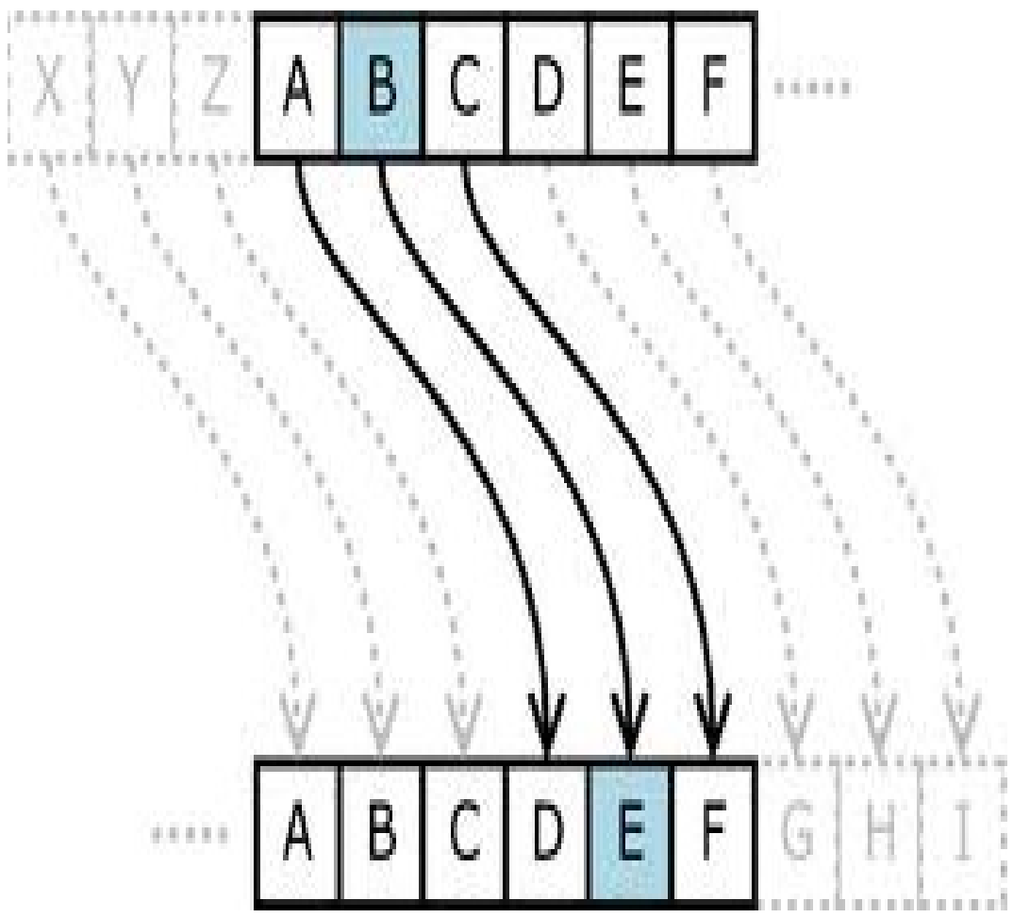
Monoalphabetic cipher

- Caesar cipher

Polyalphabetic cipher

- Vigenère cipher

Caesar Cipher



Example:

Before: RETURN TO ROME

After: UHWXUA WR URPH

Vigenère Cipher

- Example:

Message = SEE ME IN MALL

- Take keyword as INFOSEC
- Vigenère cipher works as follows:

S	E	E	M	E	I	N	M	A	L	L
I	N	F	O	S	E	C	I	N	F	O
A	R	J	A	W	M	P	U	N	Q	Z

Cryptanalysis

- The study of methods to break cryptosystems
- Often targeted at obtaining a key
- Attacks may be passive or active

Cryptanalysis

- **Kerckhoff's Principle**:
  The only secrecy involved with a cryptosystem should be the key
- **Cryptosystem Strength**:
  How hard is it to determine the secret associated with the system?

Cryptanalysis attacks

- **Brute force**
  Trying all key values in the keyspace
- **Frequency Analysis**
  Guess values based on frequency of occurrence
- **Dictionary Attack**
  Find plaintext based on common words

Cryptanalysis attacks

- **Replay Attack**
  Repeating previous known values
- **Factoring Attacks**
  Find keys through prime factorization
- **Ciphertext-Only**
- **Known Plaintext**
  Format or content of plaintext available

Cryptanalysis attacks

- **Chosen Plaintext**
  Attack can encrypt chosen plaintext
- **Chosen Ciphertext**
  Decrypt known ciphertext to discover key
- **Social Engineering**
  Humans are the weakest link

Network Security

- **SSL/TLS**
  Supports mutual authentication
  Secures a number of popular network services
- **IPSec**
  Security extensions for TCP/IP protocols
  Supports encryption and authentication
  Used for VPNs

to be continued...

Strings in Python

- Strings in Python can be created using single quotes, double quotes and triple quotes.

```
>>> a = "Alert"
```

```
>>> b = 'Alert'
```

```
>>> c = """Alert"""
```

String Functions

Finding substrings:

- **find(str, beg=0, end=len(string))**
- **rfind(str, beg=0, end=len(string))**
- **index(str, beg=0, end=len(string))**
- **rindex(str, beg=0, end=len(string))**

For more string functions

http://zetcode.com/lang/python/strings/

Exception Handling

```
try:
 You do your operations here;
except ExceptionI:
 If there is ExceptionI, then execute this block.
except ExceptionII:
 If there is ExceptionII, then execute this block.
else: If there is no exception then execute this block.
```

Exception Handling

```
try:
    fh = open("testfile", "w")
    fh.write("This is my test file for exception handling!!")
except IOError:
    print "Error: can\'t find file or read data"
else:
    print "Written content in the file successfully"
    fh.close()
```

User Inputs

```
# Ask for the number and store it in user Number
userNumber = raw_input('Give me an integer number: ')

# Make sure the input is an integer number
# What if the input is not an integer???
userNumber = int(userNumber)

# Get the square of the number
userNumber = userNumber**2

# Print square of given number
print 'The square of your number is: ' + str(userNumber)
```

User Inputs

```
# Ask for the number and store it in userNumber
userNumber = raw_input('Give me an integer number: ')

try:
    # Try to convert the user input to an integer
    userNumber = int(userNumber)
# Catch the exception if the input was not a number
except ValueError:
    userNumber = 0
else:
    # Get the square of the number
    userNumber = userNumber**2

# Print square of given number
print 'The square of your number is: ' + str(userNumber)
```

Classes

```
class name:
    "documentation"
     statements
```

-or-

```
class name(base1, base2, ...):
    ...
```

Most, *statements* are method definitions:

```
    def name(self, arg1, arg2, ...):
        ...
```

May also be *class variable* assignments

Classes

```
Example class:
class Stack:
    def __init__(self):                # constructor
        self.items = []
    def push(self, x):
        self.items.append(x)           # the sky is the limit
    def pop(self):
        x = self.items[-1]             # what happens if it's empty?
        del self.items[-1]
        return x
    def empty(self):
        return len(self.items) == 0    # Boolean result
```

Using Classes

- To create an instance, simply call the class object:

```
x = Stack()       # no 'new' operator!
```

- To use methods of the instance, call using dot notation:

```
x.empty()                 # -> 1
x.push(1)                 # [1]
x.empty()                 # -> 0
x.push("hello")           # [1, "hello"]
x.pop()                   # -> "hello"       # [1]
```

- To inspect instance variables, use dot notation:

```
x.items           # -> [1]
```

to be continued...