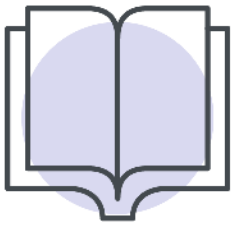
U of A

CMPUT175
MIDTERM EXAM
STUDY GUIDE



Textbook Notes

2.6

- To generate a random number, use *random.random()*.

3.3

- Plaintext are messages that are readable.
- Ciphertext are messages that are unreadable.
- Encryption is the process of turning plaintext into ciphertext.
- Decryption is the process of turning ciphertext into plaintext.

3.4 – 3.6

- A transposition cipher is made by separating the letters into even-numbered characters and odd-numbered characters, then creating one string out of them.

```
def scramble2Encrypt(plainText ):
    evenChars = ""
    oddChars = ""
    charCount = 0
    for ch in plainText :
        if charCount % 2== 0:
            evenChars = evenChars + ch
        else:
            oddChars = oddChars + ch
        charCount = charCount + 1
    cipherText = oddChars + evenChars
    return cipherText
```

- Decrypting a transposed message:

```
    def scramble2Decrypt(cipherText ):
        halfLength = len(cipherText ) // 2
        oddChars = cipherText [: halfLength ]
        evenChars = cipherText[halfLength :]
        plainText = ""

        for i in range(halfLength ):
            plainText = plainText + evenChars[i]
            plainText = plainText + oddChars[i]

        if len(oddChars ) < len(evenChars ):
            plainText = plainText + evenChars [-1]

        return plainText
```

- A substitution cipher is a type of cipher that substitutes one letter for another throughout a message.
  - Substitution ciphers use a ciphertext key, which is the rearranged version of the plaintext alphabet.

```
>>> alphabetString = "abcdefghijklmnopqrstuvwxyz"
>>> key = "zyxwvutsrqponmlkjihgfedcba"
>>> i = alphabetString.index('h')
>>> print(i)
7
>>> print(key[i])
s
```

- Encrypting a message using a substitution cipher:

```
1     def substitutionEncrypt(plainText ,key):
2         alphabet = "abcdefghijklmnopqrstuvwxyz "
3         plainText = plainText.lower ()
4         cipherText = ""
5         for ch in plainText:
6             idx = alphabet.find(ch)
7             cipherText = cipherText + key[idx]
8         return cipherText
```

  - The decrypt function would be nearly the same as the encrypt function.

- To create a random key:

```
def genRandomKey():
        alphabet = “abcdefghijklmnopqrstuvwxyz”
        key = “ “
        for i in range(len(alphabet)):
                j = random.randint(0, 25-i)
                key = key + alphabet[j]
                alphabet = removeChar(alphabet, j)
        return key
```

- To remove one character from a string:

```
def removeChar(string, idx):
        return string[:idx] + string[idx+1:]
```

- Kerckhoff’s principle is the concept that the security of a cipher should only depend on keeping the key secret.

- To remove duplicate characters from a string:

```
1    def removeDupes(myString ):
2        newStr = ""
3        for ch in myString:
4            if ch not in newStr:
5                newStr = newStr + ch
6        return newStr
```

- To remove the characters in one string from another:

```
1    def removeMatches(myString ,removeString):
2        newStr = ""
3        for ch in myString:
4            if ch not in removeString:
5                newStr = newStr + ch
6        return newStr
```

- To create a key, pick a keyword, remove repeated letters, and then add any remaining letters in order, starting after the last keyword letter.
  - If the password is Julius Caesar, the key will be:
    JULISCAERTVWXYZBDFGHKMNOPQ

```
def genKeyFromPass(password):
        key = 'abcdefghijklmnopqrstuvwxyz'
        password = removeDupes(password)
        lastChar = password[-1]
        lastIdx = key.find(lastChar)
        aherString = removeMatches(key[lastIdx+1:],password)
        beforeString = removeMatches(key[:lastIdx],password)
        key = password + aherString + beforeString
        return key
```

4.3

- Some methods for lists are:

append	alist.append(item)	Adds a new item to the end of a list
insert	alist.insert(i,item)	Inserts an item at the ith position in a list
pop	alist.pop()	Removes and returns the last item in a list
pop	alist.pop(i)	Removes and returns the ith item in a list
sort	alist.sort()	Modifies a list to be sorted
reverse	alist.reverse()	Modifies a list to be in reverse order
index	alist.index(item)	Returns the index of the first occurrence of item
count	alist.count(item)	Returns the number of item occurrences of item

`remove`	`alist.remove(item)`	Removes the first occurrence of `item`

1.2

- Computer science is the study of algorithms.
- Algorithms are 'recipes' for solving problems.
- A python identifier:
  - Is a sequence of letters, digits, and underscores.
  - Cannot start with a digit.
  - Cannot be a reserved word.
  - Can be of any length.
- An augmented assignment variable is used to replace a statement where an operator takes a variable as one of its arguments and then assigns the result back to the same variable.

Operator	Example	Equivalent
+=	i += 8	i = i + 8
-=	f -= 8.0	f = f – 8.0
*=	i *= 8	i = i * 8
/=	i /= 8	i = i / 8
%=	i %= 8	i = i % 8

- Comparison operators:
  - Less than: <
  - Less than or equal to: <=
  - Greater than: >
  - Greater than or equal to: >=
  - Equal to: ==
  - Not equal to: !=

- One-way *if* statements:

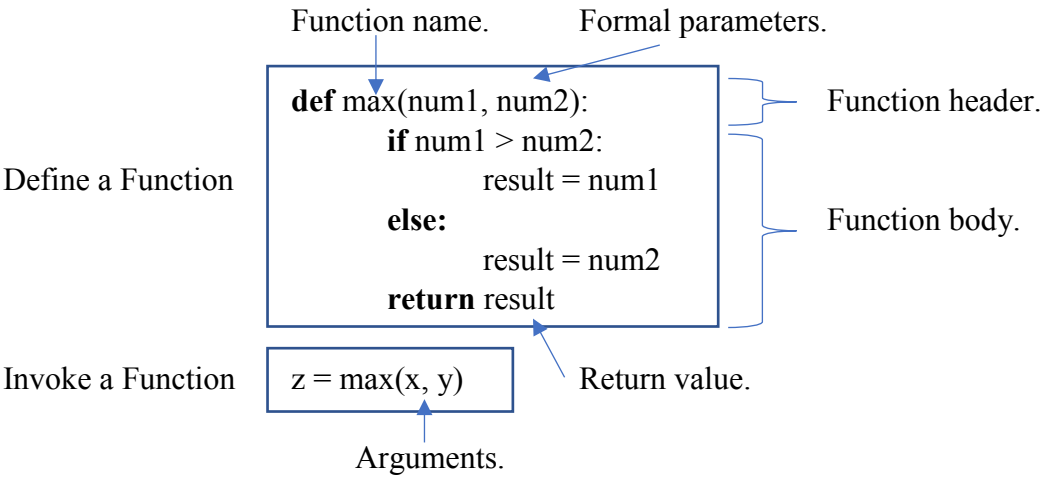
```
if radius >= 0:
        area = radius * radius * 3.14159
        print("the area for the circle of radius",
                radius, "is", area)
```

- Two-way *if* statements:

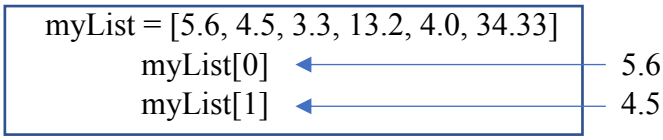
```
if Boolean-expression:
        statement (s) -for-the-true-case
else:
        statement (s) -for-the-false-case
```

- Instead of multiple *else… if* statements, one can write *elif*.

- A function is a collection of statements that are grouped together to perform an operation.



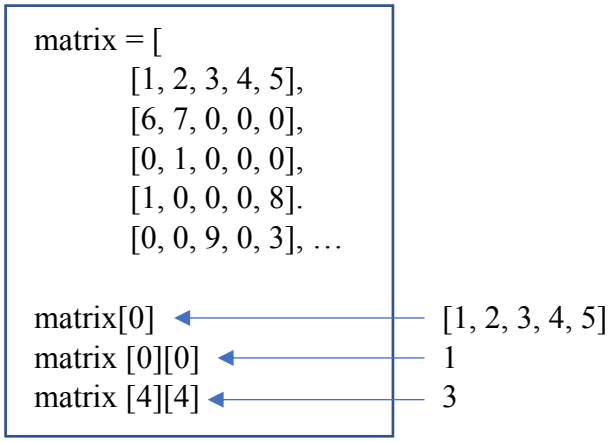
- Operators:
  - Concatenation: +
  - Repetition: *
  - Indexing: [ ]
  - Slicing: [:]



- Strings and lists are sequential collections. Dictionaries are nonsequential collections.
  - Strings are lists of characters in order. They are immutable.
  - Lists are a collection of references.
  - Dictionaries are container objects that store the elements along with their keys.
    - Keys are like an index operator, and a dictionary cannot contain duplicate keys.
    - Key + value = item
    - Dictionary methods:

keys(): tuple	Returns a sequence of keys.
values(): tuple	Returns a sequence of values.
items(): tuple	Returns a sequence of tuples (key, value).
clear(): void	Deletes all entries.
get(key): value	Returns the value for the key.
pop(key): value	Removes the entry for the key and returns its value.
popitem(): tuple	Returns a randomly-selected key/value pair as a tuple and removes the selected entry.

- Tuples are immutable, while lists are mutable.
  - You can use tuples as dictionary keys, but not lists.
- A two-dimensional list is a list of lists. Each row is a list containing the values.
  - The rows are accessed using the *row index*, and the values in each role are accessed using the *column index*.



- In Python, all data are objects.
  - An object's type is defined by a class.
  - A variable is a reference to an object.
    - The statement ' n = 3 ' assigns '3' to an 'int' object referenced by variable 'n'.

- List methods:
  - Append: list.append()
  - Insert: list.insert(place, object)
  - Pop: list.pop()
  - Sort: list.sort()
  - Reverse: list.reverse()
  - Count: list.count(object)
  - Index: list.index(object)
  - Remove: list.remove(object)

- A text file contains only characters and lines.
- A binary file can be read only by another program.
- To open a file, you must create a file object associated with the physical file.

```
file = open(filename, mode)
```

'r'	Open a file for reading only.
'w'	Open a file for writing only.
'a'	Open a file for appending data to the end of the file.
'rb'	Open a file for reading binary data.
'wb'	Open a file for writing binary data.

- To write to a file:

```
outfile = open("test.txt", "w")
outfile.write("Welcome to Python")
```

read ([number: int]): str	Returns the specified number of characters from the file. If the argument is omitted, the entire remaining contents are read.
read line(): str	Returns the next line of a file as a string.
read lines(): list	Returns a list of the remaining lines in the file.
write(s: str): None	Writes the string to a file.
close(): None	Closes the file.

- To test if a file exists:

```
import os.path
if os.path.isfile("Presidents.txt"):
        print("Presidents.txt exists")
```

4.6

- Using a dictionary to compute frequency distribution:

```
1 def frequencyTable(alist ):
2     countdict = {}
3
4     for item in alist:
5         if item in countdict:
6             countdict[item ] = countdict[item ]+1
7         else:
8             countdict[item ] = 1
9
10    itemlist = list(countdict.keys ())
11    itemlist.sort ()
12
13    print("ITEM","FREQUENCY")
14
15    for item in itemlist:
16        print(item , " ",countdict[item ])
```

5.2

- Program to read data from a file (example):

```
1 rainfile = open("rainfall.txt","r")
2
3 for aline in rainfile:
4     values = aline.split ()
5     print(values [0], "had",values [1],"inches of rain.")
6
7 rainfile.close ()
```

- Creating a file with new data:

```
1 rainfile = open("rainfall.txt","r")
2 outfile = open("rainfallInCM.txt","w")
3
4 for aline in rainfile:
5     values = aline.split ()
6
7     inches = float(values [1])
8     cm = 2.54 * inches
9
10    outfile.write(values [0]+" "+str(cm)+"\n")
11
12 rainfile.close ()
13 outfile.close ()
```

Internet

- The internet is a global network of computers.
- The world wide web contains resources, which are stored on computers.
    - Programs exchange those resources.
- A server is a program that provides resources.
- A client is a program that requires resources.
- Web browsers are considered web clients.

HTML

- Every resource has a unique name, its URL.
    - A URL is a protocol, a host, and a path.
- HTTP: Hyper Text Transfer Protocol
- HTML: Hyper Text Markup Language
    - HTML file: A web page source file.
    - HTML element: A start tag, optional attributes, data, and an end tag.
- Hyperlinks use the HTML anchor element (*a*).
    - The anchor element requires the *href* attribute.
        - Each attribute is assigned a value.
        - Each value of *href* is a URL.
            - Said URL can be absolute or relative.
        - The text in the anchor element is the clickable hyperlink.

Python WWW API

- Any program can act as a web client, but we typically use browsers.
- To act as a web client, use the Python module *urllib.request*.
    - Resources are opened like files.
        - *urlopen()* is similar to *open()*.
    - It takes a URL and sends a web request, then returns an object of the HTTPResponse.
        - The object contains the resource and additional information.

```
import urllib.request
url = 'http://www.weatheroffice.gc.ca/city/pages/ab-50_metric_e.html'
# like the open function, just returns an object representing the webpage
#
webpage = urllib.request.urlopen(url)
# like the file read method, but returns “bytes”
#
contents = webpage.read()
# to get a string from “bytes”, use decode:
#
contents = contents.decode('ISO-8859-1')
print(contents)
```

The HTTPResponse Object

geturl()	Returns the URL.
getheaders()	Returns response headers.
read()	Retrieves the content of the resource.
decode()	Decodes a text file.

- The content is a sequence of bytes.
- A text file must be decoded to use string operators and methods.

Python WWW API

- Module *urllib.request* downloads web pages.
- To process a downloaded HTML file:
    - Use string operators and methods.
    - Use regular expressions.
    - Use an HTML parser.
        - The *html.parser* module provides the HTMLParser class that parses the files.