ENGR 102 PROGRAMMING PRACTICE

WEEK 1

İSTANBUL
ŞEHİR
UNIVERSITY

Files

Persistence

- **Transient**: a program runs for a short time and produce some output, but when it ends, its data disappears. If you run the program again, it starts with a clean state.
- **Persistent**: a program runs for a long time (or all the time); it keeps at least some of their data in permanent storage (a hard drive, for example); and if it shuts down and restarts, it picks up where it left off.

Example

- Write a program that creates/overrides a file with name “data.txt” and writes into file
    - 00, 01, 02
      10, 11, 12

Writing

- To write to a file, you have to open it with mode 'w' as a second parameter:

```
fout = open('output.txt', 'w')
```

- If the file already exists, opening it in write mode clears out the old data and starts fresh, so be careful!
- If the file doesn’t exist, a new one is created.

Writing

- The write method puts data into the file.

```
line1 = "Istanbul Sehir\n"
fout.write(line1)
```

- Again, the file object keeps track of where it is, so if you call write again, it adds the new data to the end.

```
line2 = "University.\n"
fout.write(line2)
```

Closing

- When you are done writing, you have to close the file.

```
fout.close()
```

Example

- Write a program that
  - prints the contents of "data.txt" in a space separated format in one line
  - e.g., 00 01 02 10 11 12

Reading

```
file = open('newfile.txt', 'r')
for line in file:
    print line
file.close()
```

```
for line in open('newfile.txt', 'r'):
    print line
```

```
with open('newfile.txt', 'r') as file:
    for line in file:
        print line
```

Modes

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file. Does not create the file if it does not exist.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and appending.

Format operator

- The argument of write has to be a string, so if we want to put other values in a file, we have to convert them to strings. The easiest way to do that is with str:

```
x = 52

f.write(str(x))
```

- Alternative: Use the format operator: %
  - The first operand is the format string, which specify how the second operand is formatted. The result is a string.

Format operator

- For example, the format sequence '%d' means that the second operand should be formatted as an integer (d stands for “decimal”):

```
camels = 42

print 'I have spotted %d camels.' % camels
```

```
'I have spotted 42 camels.'
```

Format operator

- If there is **more than one** format sequence in the string, the second argument has to be a tuple. Each format sequence is matched with an element of the tuple, in order.
- %d: to format an integer,
- %g: to format a floating-point number
- %s: to format a string

```
print 'In %d years I have spotted %g %s.' % (3, 0.1, 'camels')
```

```
In 3 years I have spotted 0.1 camels.
```

Filenames and paths

- The **os** module provides functions for working with files and directories (“os” stands for “operating system”). os.getcwd returns the name of the current directory:

```
 import os

 cwd = os.getcwd()

print cwd
```

Filenames and paths

- A string that identifies where a file is located is called a **path**.
- A **relative** path starts from the current directory.
- An **absolute** path starts from the topmost directory in the file system.

Filenames and paths

- To find the absolute path to a file, you can use **os.path.abspath**
  - e.g., covert relative path to absolute path
- **os.path.exists** checks whether a file or directory exists.
- **os.path.isdir** checks whether it’s a directory.
- **os.path.isfile** checks whether it’s a file.
- **os.listdir** returns a list of the files (and other directories) in the given directory.

Filenames and paths

```
def walk(dir):
    for name in os.listdir(dir):
        path = os.path.join(dir, name)
        if os.path.isfile(path):
            print path
        else:
            walk(path)
```