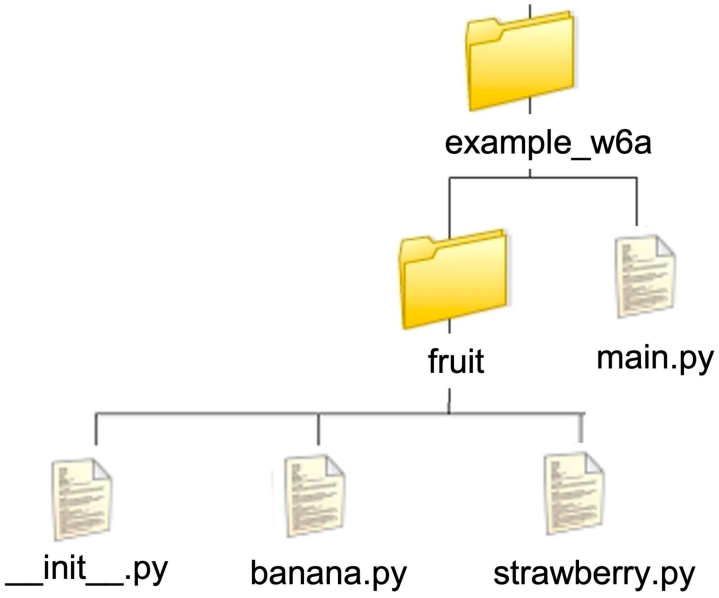
Introduction to Computer Programming

Week 6.2: Importing files from different locations

Importing a file from a different directory

Downstream file location

Example: Importing from a downstream sub-directory (a **package**)



Example: Importing from a downstream sub-directory (a **package**)

```
example_w6a/
│
├── main.py
└── fruit/
    ├── __init__.py
    ├── banana.py
    └── strawberry.py
```

`.` is used to indicate a sub-directory downstream of the current location:

```
import subfolder.file
import folder.subfolder.file
```

Everything after `import` is stored in the local namespace.
It and must be used to prepend any variables, functions etc from the imported module.

File contents

`main.py`

```
import fruit
print(fruit.strawberry.word)
```

`fruit/___init__.py`

```
# (empty file)
```

`fruit/_banana.py`

```
word = 'banana'
```

`fruit/_strawberry.py`

```
word = 'strawberry'
```

The longer namespace when packages are imported can make code long and difficult to read.

Example Renaming `fruit.strawberry --> strawb` to make code shorter and neater

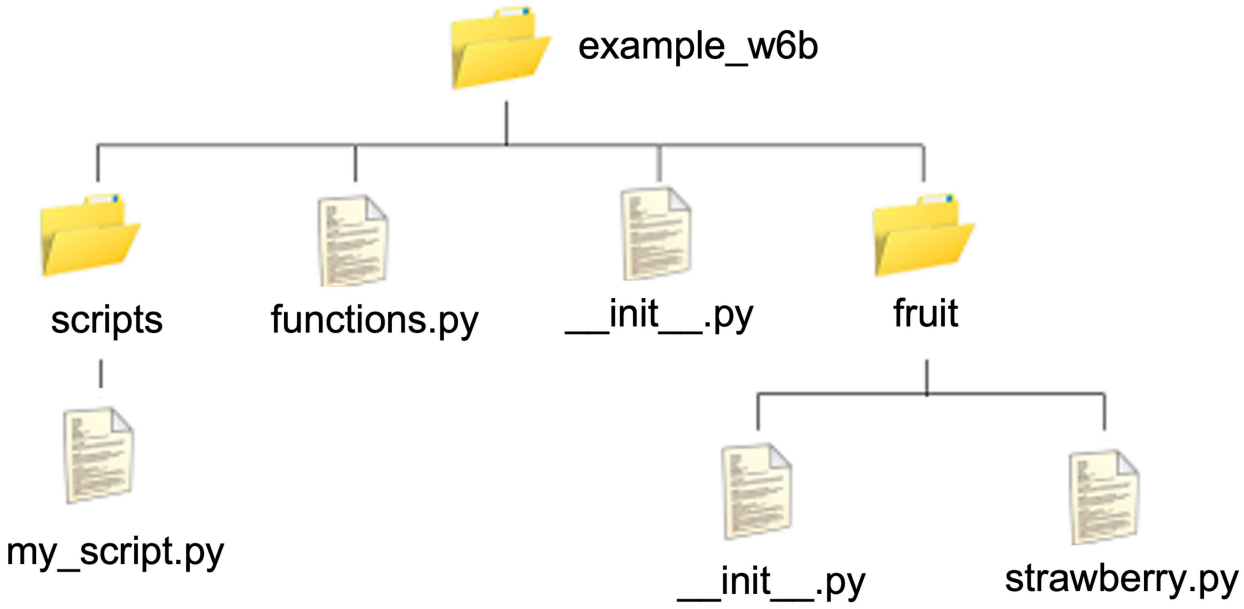
main.py

```
import fruit.strawberry as strawb # OR from fruit import strawberry as strawb
print(strawb.word)
```

Upstream file location

Example: Importing from an upstream directory

If a file is located *upstream* of a script being run, it cannot automatically be found by the Python interpreter.




```
example_w6b/
│
├── scripts/
│   └── my_script.py
│
├── fruit/
│   ├── __init__.py
│   └── strawberry.py
│
├── functions.py
└── __init__.py
```

This is because Python only looks for modules and packages in its **import path**.

This is a list of locations:

- current directory
- contents of `PYTHONPATH` variable (a list of user defined directories)
- standard directories automatically set when python installs

To view the path we can use the `sys` module which installs with Python:

In [1]:

```
import sys

print(sys.path)
```

```
['C:\\Users\\hemma\\iCloudDrive\\Documents\\Code\\Jupyter_NBooks\\Teaching
\\UoB\\UoB_ICP_2021', 'C:\\Users\\hemma\\anaconda3\\python38.zip', 'C:\\User
s\\hemma\\anaconda3\\DLLs', 'C:\\Users\\hemma\\anaconda3\\lib', 'C:\\Users
\\hemma\\anaconda3', '', 'C:\\Users\\hemma\\anaconda3\\lib\\site-packages',
'C:\\Users\\hemma\\anaconda3\\lib\\site-packages\\locket-0.2.1-py3.8.egg',
'C:\\Users\\hemma\\anaconda3\\lib\\site-packages\\win32', 'C:\\Users\\hemma
\\anaconda3\\lib\\site-packages\\win32\\lib', 'C:\\Users\\hemma\\anaconda3
\\lib\\site-packages\\Pythonwin', 'C:\\Users\\hemma\\anaconda3\\lib\\site-pa
ckages\\IPython\\extensions', 'C:\\Users\\hemma\\.ipython']
```

To add a *location* to the path from within a python program we can use `sys` .

`../` is used to indicate a location one directory upstream of the current location.

```
example_w6b/
│
├── scripts/
│   └── my_script.py
│
├── fruit/
│   ├── __init__.py
│   └── strawberry.py
│
├── functions.py
└── __init__.py
```

Example: In `my_script.py` :

```
import sys
sys.path.append('../') # appends the python path with the directory one level up
import functions
sys.path.append('../fruit') # appends the python path with a different directory a
t the same level
import strawberry
```

A directory can be removed in a similar way:

```
sys.path.remove('../')
```

Summary

```
example_w6a/
│
├── main.py
└── fruit/
    ├── __init__.py
    ├── banana.py
    └── strawberry.py
```

In `main.py` ...

Importing package from same directory:

```
import
print(fruit.strawberry.word)
```

Importing submodule:

```
from fruit import strawberry
print(strawberry.word)
```

Renaming :

```
import fruit.strawberry as strawb # OR from fruit import strawberry as strawb
print(strawb.word)
```

Importing variable:

```
from fruit.strawberry import word
print(word)
```

Importing and rename variable

```
from fruit.strawberry import word as w
print(w)
```
