Introduction to

PYTHON PROGRAMING

MODULES AND PACKAGES

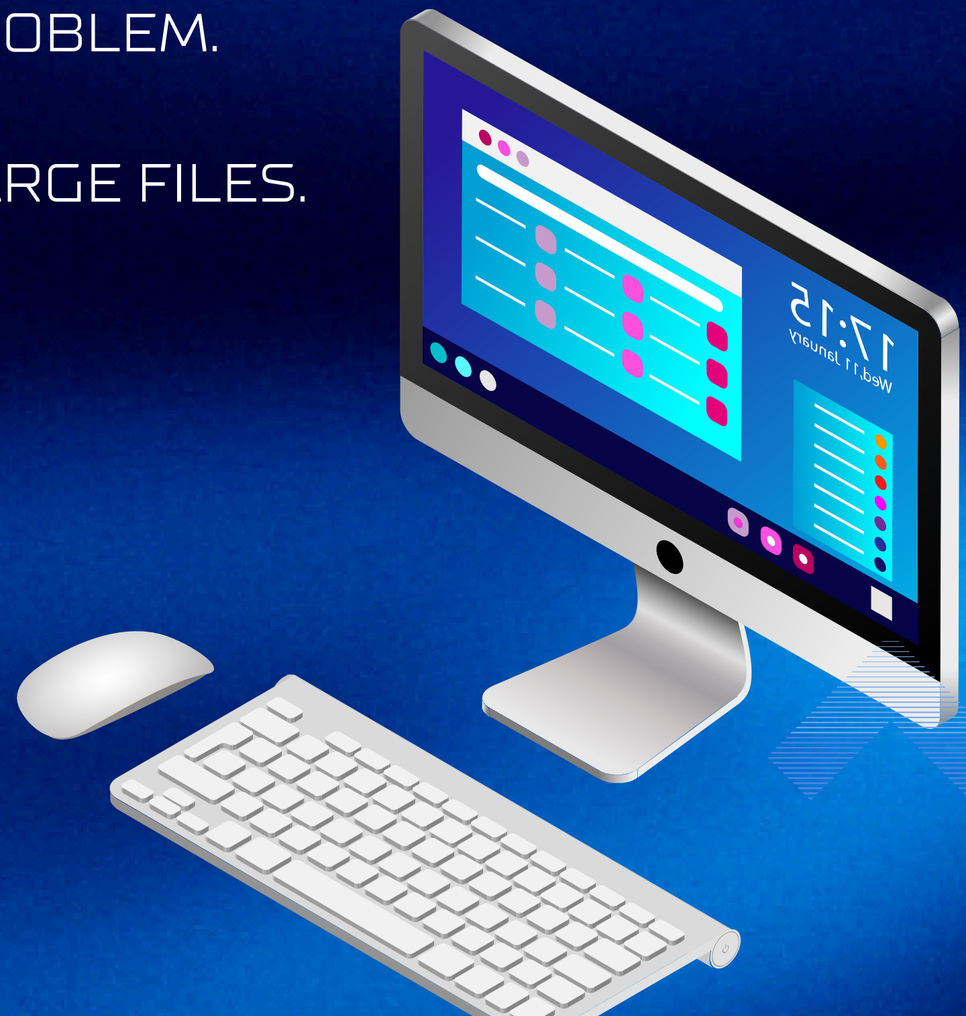
THERE ARE FOUR MAIN ADVANTAGES TO BREAKING A PROGRAM INTO MODULES:

1. SIMPLICITY: MODULES ARE FOCUSED ON A SINGLE PROBLEM.

2. MAINTAINABILITY: SMALL FILES ARE BETTER THAN LARGE FILES.

3. REUSABILITY: MODULES REDUCE DUPLICATE CODE.

4. SCOPING: MODULES HAVE THEIR OWN NAMESPACES.



CREATING MODULES

```python
# adder.py


def add(x, y):
    return x + y
```

```python
# main.py


value = add(2, 2)

print(value)
```

```
Traceback (most recent call last):
  File "//Documents/myproject/main.py", line 1, in <module>
    value = add(2, 2)
NameError: name 'add' is not defined
```

IMPORTING ONE MODULE INTO ANOTHER

```python
# main.py

import adder  # <-- Add this line

# Leave the code below unchanged
value = add(2, 2)
print(value)
```

Important

The name used to import a module is the same as the module's file name.

For this reason, module file names must be valid Python identifiers. That means they may only contain upper and lower case letters, numbers, and underscores (_), and they may not start with a digit.

```python
# main.py

import adder

value = adder.add(2, 2)  # <-- Change this line
print(value)
```

```python
# adder.py

# Leave this code unchanged
def add(x, y):
    return x + y

def double(x):  # <-- Add this function
    return x + x
```

```python
# main.py

import adder

value = adder.add(2, 2)
double_value = adder.double(value)  # <-- Add this line
print(double_value)  # <-- Change this line
```

HOME ABOUT MORE

IMPORT STATEMENT VARIATIONS

1. IMPORT AS <MODULE> AS <OTHER_NAME>

```
import adder as a  # <-- Change this line

# Leave the code below unchanged
value = adder.add(2, 2)
double_value = adder.double(value)
print(double_value)
```

```
Traceback (most recent call last):
  File "//Mac/Home/Documents/myproject/main.py", line 3, in <module>
    value = adder.add(2, 2)
NameError: name 'adder' is not defined
```

```
import adder as a

value = a.add(2, 2)  # <-- Change this line
```

```
double_value = a.double(value)  # <-- Change this line, too
print(double_value)
```

2. FROM IMPORT <MODULE> IMPORT <NAME>


```
from adder import add  # <-- Change this line

value = adder.add(2, 2)
double_value = adder.double(2, 2)
print(double_value)
```

```
Traceback (most recent call last):
  File "//Documents/myproject/main.py", line 3, in <module>
    value = adder.add(2, 2)
NameError: name 'adder' is not defined
```

```
from adder import add

value = add(2, 2)  # <-- Change this line
double_value = double(value)  # <-- Change this line, too
print(double_value)
```

```
Traceback (most recent call last):
  File "//Documents/myproject/main.py", line 4, in <module>
    double_value = double(value)
NameError: name 'double' is not defined
```

```
from adder import add, double  # <-- Change this line

# Leave the code below unchanged
value = add(2, 2)
double_value = double(value)
print(double_value)
```

SUMMARY OF IMPORT STATEMENTS

Import Statement	Result
`import <module>`	Import all of `<module>`'s namespace into the name `<module>`. Import module names can be accessed from the calling module with `<module>.<name>`.
`import <module> as <other_name>`	Import all of `<modules>`'s namespace into the name `<other_name>`. Import module names can be accessed from the calling module with `<other_name>.<name>`.
`from <module> import <name1>, <name2>, ...`	Import only the names `<name1>`, `<name2>`, etc, from `<module>`. The names are added to the calling modules's local namespace and can be accessed directly.

WHY USE NAMESPACES?

1. THEY GROUP NAMES INTO LOGICAL CONTAINERS

2. THEY PREVENT CLASHES BETWEEN DUPLICATE NAMES

3. THEY PROVIDE CONTEXT TO NAMES

THERE ARE TWO REASONS YOU MIGHT USE THE IMPORT <MODULE> AS <OTHER_NAME> FORMAT:

1. THE MODULE NAME IS LONG AND YOU WISH TO IMPORT AN ABBREVIATED VERSION OF IT

2. THE MODULE NAME CLASHES WITH AN EXISTING NAME IN THE CALLING MODULE

```python
import datetime
```

```python
datetime.datetime(2020, 2, 2)
```

```python
import datetime as dt
```

```python
dt.datetime(2020, 2, 2)
```

```python
from datetime import datetime
```

```python
datetime(2020, 2, 2)
```

REVIEW EXERCISES

1. CREATE A MODULE CALLED *GREETER.PY* THAT CONTAINS A SINGLE FUNCTION GREET(). THIS FUNCTION SHOULD ACCEPT A SINGLE STRING PARAMETER NAME PRINT THE TEXT HELLO {NAME}! TO THE INTERACTIVE WINDOW WITH {NAME} REPLACED WITH THE FUNCTION ARGUMENT.

2. CREATE A MODULE CALLED *MAIN.PY* THAT IMPORTS THE GREET() FUNCTION FROM *GREET.PY* AND CALLS THE FUNCTION WITH THE ARGUMENT "REAL PYTHON".

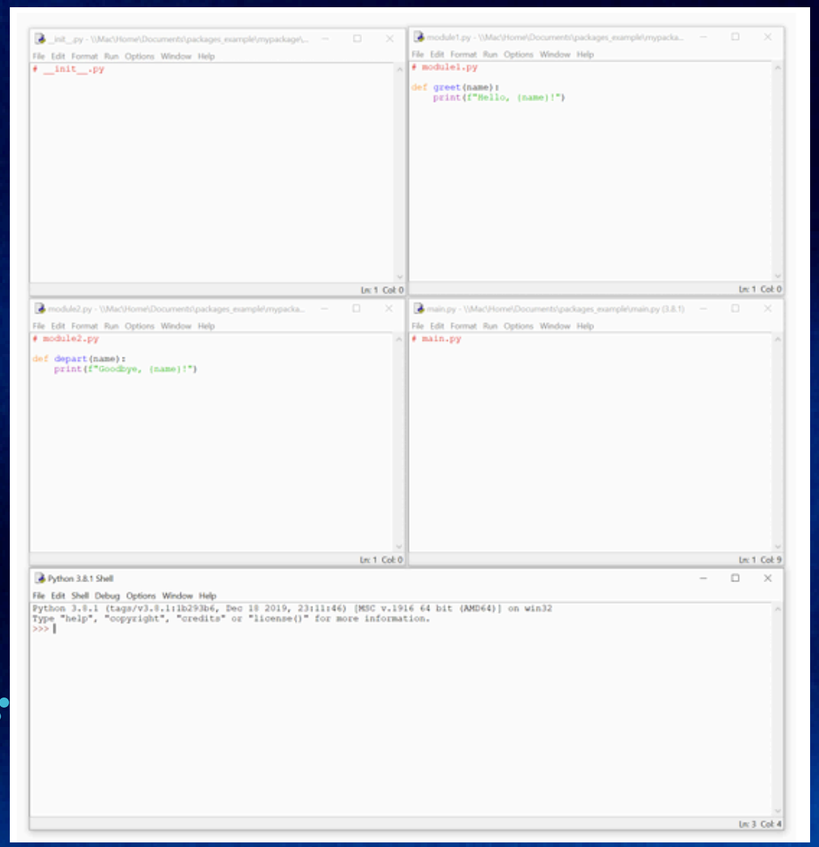
CREATING PACKAGES

USING YOUR COMPUTERS FILE EXPLORER, OR WHATEVER TOOL YOU ARE COMFORT ABLE WITH, CREATE A NEW FOLDER SOMEWHERE ON YOUR COMPUTER CALLED **PACKAGES_EXAMPLE/**.

INSIDE OF THAT FOLDER, CREATE ANOTHER FOLDER CALLED **MYPACKAGE/.** THE *PACKAGES_EXAMPLE/* FOLDER IS CALLED THE **PROJECT FOLDER**, OR **PROJECT ROOT FOLDER,**

```python
# main.py
```

```python
# __init__.py
```



```python
# module1.py

def greet(name):
    print(f"Hello, {name}!")
```

```python
def depart(name):
    print(f"Goodbye, {name}!")
```

IMPORTING MODULES FROM PACKAGES

```python
# main.py

import mypackage

mypackage.module1.greet("Pythonista")
mypackage.module2.depart("Pythonista")
```

```
Traceback (most recent call last):
  File "\MacHomeDocumentspackages_examplemain.py", line 5, in <module>
    mypackage.module1.greet("Pythonista")
AttributeError: module 'mypackage' has no attribute 'module1'
```

```python
# main.py

import mypackage.module1  # <-- Change this line

# Leave the below code unchanged
mypackage.module1.greet("Pythonista")
```

```python
mypackage.module2.depart("Pythonista")
```

```
Hello, Pythonista!
Traceback (most recent call last):
  File "\MacHomeDocumentspackages_examplemain.py", line 6, in <module>
    mypackage.module2.depart("Pythonista")
AttributeError: module 'mypackage' has no attribute 'module2'
```

```python
# main.py

import mypackage.module1
import mypackage.module2  # <-- Add this line

# Leave the below code unchanged
mypackage.module1.greet("Pythonista")
mypackage.module2.depart("Pythonista")
```

```
Hello, Pythonista!
Goodbye, Pythonista!
```

Important

Just like module file names, package folder names must be valid Python identifiers. They may only contain upper and lower case letters, numbers, and underscores (_), and they may not start with a digit.

Import Statement Variations For Packages

1. IMPORT <PACKAGES>

2. IMPORT AS <PACKAGES> AS <OTHER_NAME>

3. FROM <PACKAGE> IMPORT <MODULE>

4. FROM <PACKAGES> IMPORT <MODULE> AS <OTHER_NAME>

```python
# main.py

from mypackage import module1, module2
```

```python
module1.greet("Pythonista")
module2.depart("Pythonista")
```

```python
# main.py

from mypackage import module1 as m1, module2 as m2

m1.greet("Pythonista")
m2.depart("Pythonista")
```

```python
# main.py

from mypackage.module1 import greet
from mypackage.module2 import depart

greet("Pythonista")
depart("Pythonista")
```

GUIDELINES FOR IMPORTING PACKAGES

```
import <package>.<module>
```

```
<package>.<module>.<name>
```

```
from <package> import <module>
```

```
from <package>.<module> import <name>
```

Importing Modules From Subpackages

```python
# module3.py

people = ["John", "Paul", "George", "Ringo"]
```

```python
# main.py

from mypackage.module1 import greet
from mypackage.mysubpackage.module3 import people

for person in people:
    greet(person)
```

```
Hello, John!
Hello, Paul!
Hello, George!
Hello, Ringo!
```

Review Exercises

1. In a new project folder called package_exercises/, create a package called helpers with three modules: __init__.py, string.py, and math.py

   In the string.py module, add a function called shout() that takes a single string parameter and returns a new string with all of the letters in uppercase. In the math.py module, as a function called area() that takes two parameters called length and width and returns their product length * width.

2. In the root project folder, create a module called main.py that im ports the shout() and area() functions. Use the shout() and area() functions to print the following output:

```
THE AREA OF A 5-BY-8 RECTANGLE IS 40
```

THANK YOU