11. Modules

October 19, 2022

1 Introduction

- A module is a file containing Python definitions and statements.
- A module can define functions, classes, and variables.
- A module can also include runnable code.
- Grouping related code into a module makes the code easier to understand and use.
- It also makes the code logically organized.

2 Creating modules

- To create a module put required lines of code in a python script (text file with .py extension)

3 The `import` statement

- We can use any Python source file as a module by executing an import statement in some other Python source file.
- When the interpreter encounters an import statement, it imports the module if the module is present in the search path.
- If not specified default path is current working directory
- A search path is a list of directories that the interpreter searches for importing a module.

[ ]:
```
import my_module
```

[ ]:
```
my_module.factorial(3)
```

[ ]:
```
my_module.armstrong_number3(370)
```

[ ]:
```
my_module.OUTPUT_PATH
```

4 The `from...import` statement

- `from` statement lets you import specific attributes from a module.
- Syntax: `from <module_name> import <object_name>`

[ ]:
```
from my_module import factorial, armstrong_number3
```

[ ]:
```
factorial(3)
```

5 The `from...import *` statement

- Imports all th objects from module.
- The use of * has it's advantages and disadvantages. If you know exactly what you will be needing from the module, it is not recommended to use *, else do so.

[ ]:
```
from my_module import *
```

[ ]:
```
factorial(3)
```

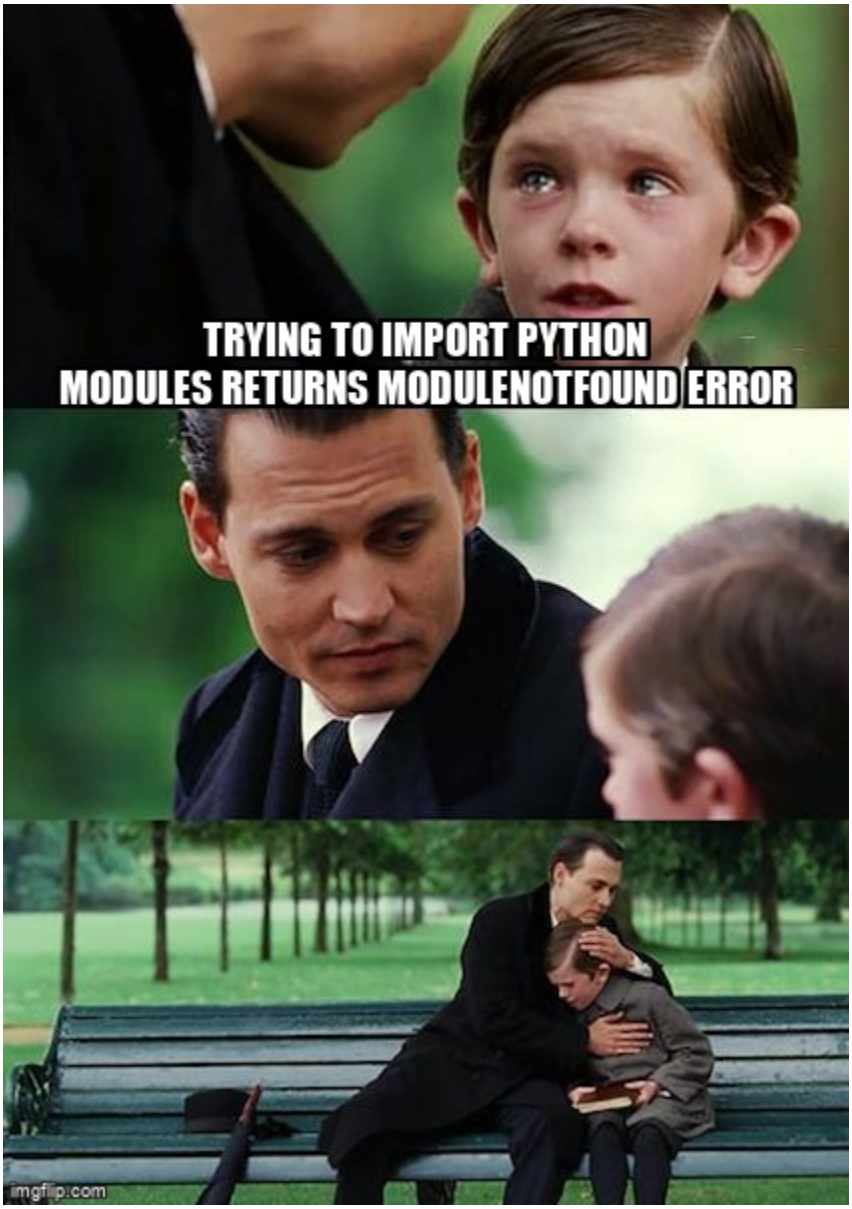
[ ]:
```
armstrong_number3(389)
```

[ ]:
```
OUTPUT_PATH
```

[1]:
```
import themodule
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In [1], line 1
----> 1 import themodule

ModuleNotFoundError: No module named 'themodule'
```



6 Python built-in modules

6.1 Math

[ ]:
```
# importing built-in module math
import math
```

[ ]:
```
# using square root(sqrt) function contained
# in math module
print(math.sqrt(25))
```

[ ]:
```
# using pi function contained in math module
print(math.pi)
```

[ ]:
```
# 2 radians = 114.59 degreees
print(math.degrees(2))
```

[ ]:
```
# 60 degrees = 1.04 radians
print(math.radians(60))
```

[ ]:
```
# Sine of 2 radians
print(math.sin(2))
```

[ ]:
```
# Cosine of 0.5 radians
print(math.cos(0.5))
```

[ ]:
```
# Tangent of 0.23 radians
print(math.tan(0.23))
```

[ ]:
```
# 1 * 2 * 3 * 4 = 24
print(math.factorial(4))
```

6.2 Random

[ ]:
```
 # importing built in module random
import random
```

[ ]:
```
# printing random integer between 0 and 5
print(random.randint(0, 5))
```

[ ]:
```
# print random floating point number between 0 and 1
print(random.random())
```

[ ]:
```
# random number between 0 and 100
print(random.random() * 100)
```

[ ]:
```
List = [1, 4, True, 800, "python", 27, "hello"]

# using choice function in random module for choosing
# a random element from a set such as a list
print(random.choice(List))
```

6.3 Datetime

[ ]:
```
# importing built in module datetime
import datetime
```

[ ]:
```
datetime.datetime.now()
```

[ ]:
```
x = datetime.datetime.now()
```

[ ]:
```
x.year
```

[ ]:
```
x.strftime("%D")
```

[ ]:
```
help(datetime.datetime.strftime)
```

[ ]:
```
x1 = datetime.datetime(2020, 6, 14)
```

[ ]:
```
x2 = datetime.datetime(2021, 6, 12)
```

[ ]:
```
x1- x2
```

6.4 Time

[ ]:
```
import time
```

[ ]:
```
# Returns the number of seconds since the
# Unix Epoch, January 1st 1970
print(time.time())
```

[ ]:
```
from datetime import date
```

[ ]:
```
# Converts a number of seconds to a date object
y = time.time()
print(date.fromtimestamp(y))
```

6.5 os

- The OS module in Python provides functions for interacting with the operating system.

[ ]:
```
import os
```

[ ]:
```
# Get the current working directory
os.getcwd()
```

[ ]:
```
# creating a directory
os.mkdir("My_directory")
```

[ ]:
```
#Listing out Files and Directories
os.listdir()
```

- `os.remove()` method is used to remove or delete a file
- `os.rmdir()` method is used to remove or delete a empty directory.
- Explore `shutil` package

[ ]:
```
dir(os) # To check all functions and variables available in a module
```

7 pycache

- When you run a program in python, the interpreter compiles it to bytecode first (this is an oversimplification) and stores it in the `__pycache__` folder. If you look in there you will find a bunch of files sharing the names of the .py files in your project's folder, only their extensions will be either .pyc or .pyo. These are bytecode-compiled and optimized bytecode-compiled versions of your program's files, respectively.
- As a programmer, you can largely just ignore it… All it does is make your program start a little faster. When your scripts change, they will be recompiled, and if you delete the files or the whole folder and run your program again, they will reappear (unless you specifically suppress that behavior).
- When you're sending your code to other people, the common practice is to delete that folder, but it doesn't really matter whether you do or don't. When you're using version control (`git`), this folder is typically listed in the ignore file (`.gitignore`) and thus not included.