Python: modules

Week 4.2

Today's goals

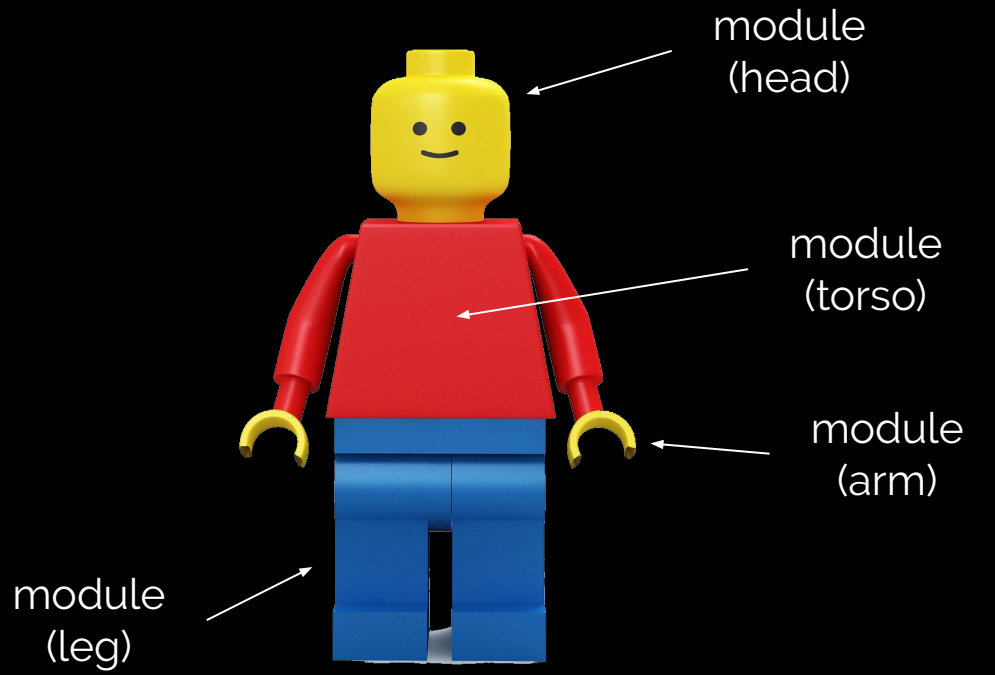
1. Understand the concept of module.
2. How to **import builtin Python modules** (e.g. `string`, `random`, `hashlib`, `..`).
3. Use `csv` module to **read and write** to CSV files.
4. Use `secrets` module to **generate pseudo-random passwords**.
5. Use `os` module to:
    - Generate file paths.
    - Check for specific files.
    - Navigate through a folder system.
6. Perform a **simple attack** on an operating system.

1-2. Understand the concept of module and import builtin modules.

What are modules?

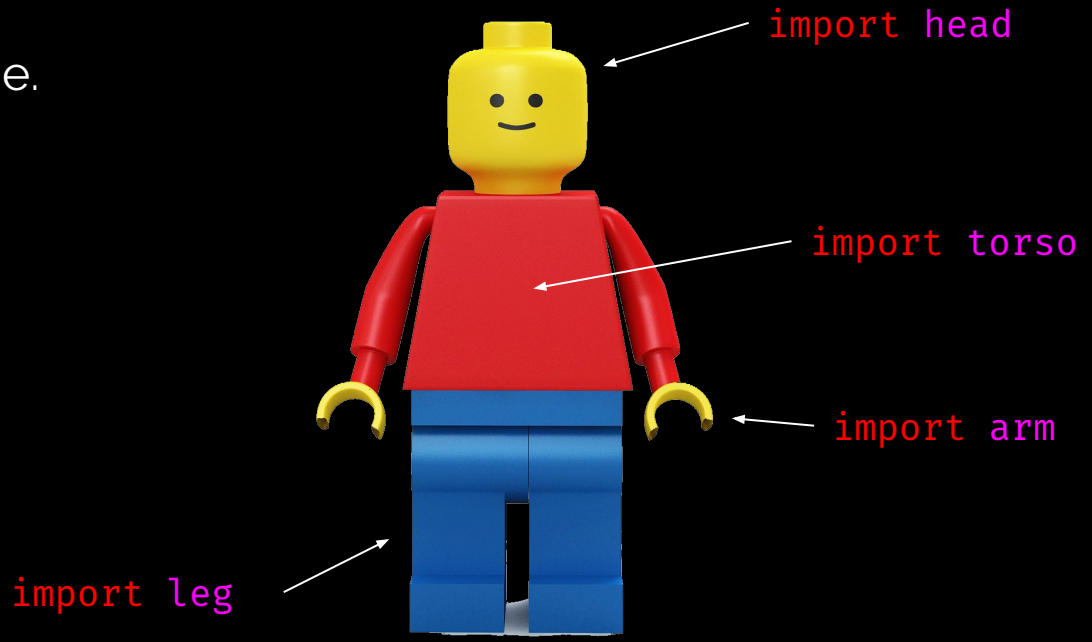
Modules are **distinct blocks of components** (e.g. code) that can be **used together** to build a program.

Imagine a Lego person.



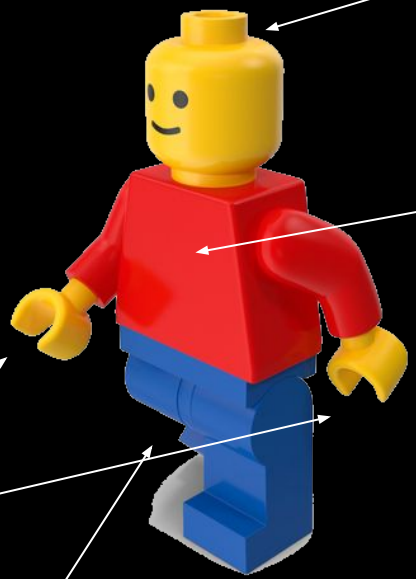
How do we use modules?

1. Use `import` on module name.



How do we use modules? (cont.)

1. Use `import` on module name.
2. Call implementation in a module *namespace*[*].



```
import head
```

```
import torso
```

```
import arm
arm.right.move(8)
arm.left.move(-8)
```

```
import leg
leg.right.move(-5)
leg.left.move(0)
```

Multiple module imports?

```
# we can consolidate multiple
# import statements from:
import a
import b
import c
import d
```

```
# into a single import statement:
import a, b, c, d

# which is same as:
import (a, b, c, d)

# which is also same as having
# newlines and/or spaces
# between commas:
import (a, b,
        c,
        d)
```

Specific attribute/function imports from module

```
# let's say `head` module
# contains `hair`, `eye`, etc.
# attributes. We can import
# everything about `head`:
import head

# and then access `hair`
# attribute through the
# `head` namespace:
head.hair.color = "black"
```

```
# or, we can directly access
# `hair` from `head`:
from head import hair
hair.color = "black"

# and even rename our `hair`
# namespace to something else:
from head import hair as dog_hair
dog_hair.color = "black"
```

Your turn!

1. Import **at least five different** Python modules from **The Python Standard Library** in a new file called **module_practice.py**. The standard library link is https://docs.python.org/3/library/index.html

   Example: `import string, random, hashlib`

2. `./week4/4.2/02-Stu_ModuleHunter/Unsolved/ModuleHunter.py`

[Instructor Review]

3. Use `csv` module to **read and write** to CSV files.

`csv` module review

- CSV is a format that defines “comma-separated values”.

Imagine a Microsoft Excel file that contains these two rows:

Date	Firstname	Lastname
2018/04/04	Herbert,	Shin

transform into CSV format

```
Date,Firstname,Lastname
2018/04/04,Herbert”,”,Shin
```

csv module demo

```python
# import `csv` module
import csv

# open 'WWE-data.csv' file and
# store its' buffer into variable
csv_file = open('./WWE-data.csv')

# parse the buffer into a line reader
# and store the iterator into variable
contents = csv.reader(csv_file)

# print each lines
for line in contents:
    print(line)
```

```
# `WWE-data.csv` for terminal demo:

Wrestler,wins,losses,draws
Dean Ambrose,133,67,4
Kevin Owens,61,130,2
```

Your turn!

1. Rewrite the previous script (below) so that, instead of printing out information to the screen, it instead **writes the lines directly to a new text file** called **PeoplesToKeepEyesOn.txt**.

```
import csv
csv_file = open('./WWE-data.csv')
contents = csv.reader(csv_file)
for line in contents:
    print(line)  # <-- hint
```

Hint: We used `.read()` before to read an opened file. Is there such thing as `.write()`? Incidentally, we used `csv.reader`. Is there a such thing as a "writer"?

[Instructor Review]

4. Use `secrets` module **generate pseudo-random passwords**.

What is `secrets` module?

From Python docs:

The secrets module is used for **generating cryptographically strong* random numbers*** suitable for managing data such as **passwords, account authentication, security tokens, and related secrets**.

Note: cryptographically strong means “highly resistant to cryptanalysis”.

`secrets` module demo

```
# Example #1: generate a pseudo-random
# number below N.

# import `secrets` module
import secrets

# print a random number below N=100
print(secrets.rand_below(100))  # 53

# print a random number below N=500
print(secrets.rand_below(500))  # 237
```

```
# Example #2: choose a random element
# from a list.

# import `secrets` module
import secrets

# create a list containing each
# characters of string ‘apple’.
# apple -> [‘a’, ‘p’, ‘p’, ‘l’, ‘e’]
apple = list(‘apple’)

# choose a random element from `apple`
print(secrets.choice(apple))  # ‘a’
```

Your turn!

1. Using secrets module, **generate a list of 100 random password strings** and **write each results into a file** called **PasswordsList.txt**.

Hint: using what we already know about `secrets.randbelow()`, `secrets.choice()`, `for` loop, and writing strings into a file, can we generate multiple strings composed of random characters/symbols/numbers?

Extra: can we do even better job by setting a minimum password length requirement of each generated passwords?

[Instructor Review]

5. os module

What is `os` module?

From Python docs:

This module provides a portable way of using **operating system dependent functionality**.

For example, `os.path.join()` method allows OS-independent interface to concatenate directory paths:

- UNIX: `/root/folder_1/folder_2/`
- Windows: `C:\folder_1\folder_2\`

Your turn!

1. Write a program that **prompts the user for a file name** (e.g. **Dracula.txt**) and then **searches the Books** directory for it.
    a. If the file exists, then **print the file content** to the terminal.
    b. Otherwise, print the string "*Sorry! That book is not in our records! Please try again!*"

Hint #1: you must first unzip **Books.zip** file before implementing the program.

Extra: since it's possible that users forget to put file extensions, can we do better by checking for file extensions (like ".txt") and concatenating to the user input?

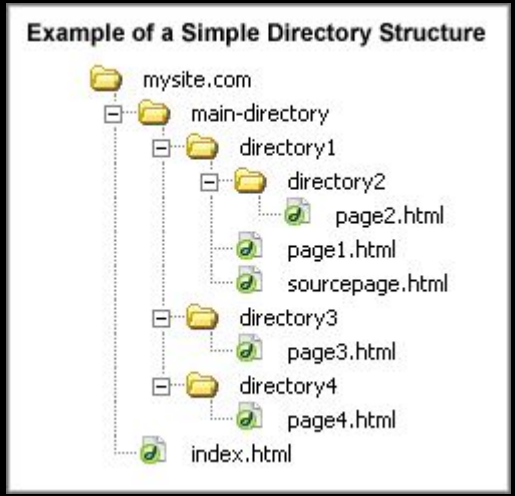
[Instructor Review]

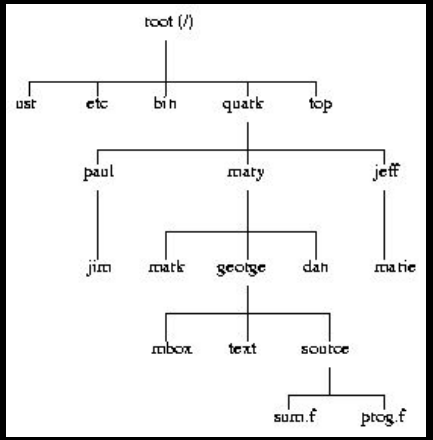
5. Perform **a simple attack** on an operating system.

os.walk

From Python docs:

Generate the file names in a directory tree by walking the tree either top-down or bottom up.





os.walk demo

```
# import `secrets` module
import os

# store file path ‘Resources/DiaryEntries’ into variable
folder_path = os.path.join(‘Resources’, ‘DiaryEntries’)

# traverse the ‘Resources/DiaryEntries’ directory
for root, dirs, files in os.walk(folder_path):
    print(‘Currently inside of ‘   + root)        # tree root node
    print(‘Contains directories: ‘ + str(dirs))   # directory node(s)
    print(‘Contains files: ‘       + str(files))  # file node(s)
```

Your turn!

You will be taking on the role of a hacker. Your task is to completely overwrite the victim's files with the phrase "GET WREKT!".

1. Write an application that will check the **Diaries** directory and automatically navigate through its' subdirectories/files.
2. If the application finds a file, replace the file content with the phrase "GET WRECKT!".

Hint #1: you must first unzip **Diaries.zip** file before implementing the program.

Hint #2: `os.walk` through the **Diaries** directory.