The Python Programming Language (Part 2)

CST 205

Today’s Lightening Intro to Python (Part 2)

- Introduce you to more Python goodies:
  - Tuples
  - More lists
  - F-strings
  - Range
  - Loops
  - Dictionaries
  - Python modules
  - Random

Review: Lists and Tuples

- Lists are ordered sequences
    - Denoted by square brackets, **[]**
- Tuples are similar to lists, but they are *immutable*
    - Denoted by parentheses, **()**

Tuple example

```
hot_pink = (255, 105, 180)


# The following won't work:

hot_pink[0] = 250
```

List example

```
dangerous_elements = ['plutonium', 'polonium', 'caesium']

dangerous_elements.append('arsenic')
```

f-strings

- Also known as “literal string interpolation”
- A simple (and computationally efficient) to format strings in Python

```
username = 'joaquín'

print(f'Your username is {username}’)
```

Generate sequences with **range()**

- Generate a list of numbers
- General format is: `range(start, end_exclusive)`
    - If no start is given, **range()** starts at **0**

```
range_1 = range(5)

range_2 = range(4,9)
```

range() step value

- If a third argument is provided, this is used as a step value.
    - The step value can be negative to “go backwards”

```
range_3 = range(40, 56, 4)

range_4 = range(20, 9, -2)

print(f'range_4: {list(range_4)}')
```

Loops

- Loops are used for repeating a single section of code multiple times
- **for** loops are used when we want to perform some action for every thing in some group
    - We *iterate* over every element in a list.

```
dangerous_elements = ['plutonium', 'polonium', 'caesium']

for element in dangerous_elements:
    print(f'{element.capitalize()} is dangerous!')
```

for loop with range()

```
for i in range(5, 0, -1):
    print(f'{i}!')
    if i == 1:
        print('Blast off!')
```

while loops

- **while** loops perform some action **while** some condition is True.
- The loop terminates once the condition becomes False.
- Great way to write infinite loops!

while loop example

```
secret_number = 4
guess = None
guess_counter = 0

while guess != secret_number:
    if guess_counter == 0:
        guess = int(input("Guess the secret number between 0 and 10: "))
    else:
        guess = int(input("Try again: "))
    guess_counter += 1

if guess_counter == 1:
    print("\nGreat job! It took you only one guess!")
elif guess_counter < 6:
    print(f"\nPretty good. It took you {guess_counter} guesses.")
else:
    print(f"\nGet a new crystal ball. It took you {guess_counter} guesses.")
```

Functions

- Reusable code that you can *call* by the name of the function
- Can take parameters
- Can *return* values
    - Python will return **None** if no explicit value is specified.
- Python uses the **def** keyword.

Examples of functions

```
# no parameters
def say_hello():
    print('Hello!')

# one parameter
def hello_you(your_name):
    print(f'Hello, {your_name}!')

# return
def name_year(your_name, birth_year):
    your_age = 2018 - birth_year
    return f'Hi {your_name}, you are {your_age} years old.'

# call the functions
say_hello()

hello_you('Sammy')

print(name_year('Elon', 1971))
```

Scope

- Variables have **scope**, which is the visibility or lifetime of the variable.
- Any variable declared outside of a function can be used inside or outside of that function.
- Any variable declared inside of a function **cannot** be used outside of that function.

Dictionaries

- A data structure consisting of key-value pairs.
    - In other languages, sometimes called associative arrays or hash maps.
- Curly braces, {}, denote a Python dictionary
- Prior to Python 3.6, dictionaries did not preserve order.
- Can create, update, add to, delete, search, and loop through dictionaries.

Dictionary example

```
csumb_dictionary = {
    'year_founded' : 1994,
    'num_students' : 7_200,
    'first_gen_percent' : 65,
    'location' : 'Marina, California'
}

csumb_dictionary['num_students']
```

Nesting

- It is very common to see lists inside lists, or dictionaries inside dictionaries, or dictionaries inside of lists inside of dictionaries, and so on.

```
audio_formats = ['flac', 'm4a', 'mp3']

image_formats = ['jpeg', 'gif', 'png']

image_audio = [audio_formats, image_formats]

print(image_audio[0][2])

for my_format in image_audio:
    for i in my_format:
        print(i)
```

Python 3 Standard Library

- Hundreds of **modules** that provide tools for interacting with the operating system, interpreter, and internet
    - A **module** is a file containing Python definitions and statements.
- We already saw **range()**
- For certain parts of the standard library, we need to import
    - One the next slide, we'll look at the **random** module.

The **random** module

- Pseudo-random number generator
- Designed for modelling and simulation, **not** for security or cryptography.
- Example: Print a random number between 0 and 1

```
import random

print(random.random())
```

random.choice()

- Pseudo-randomly choose a value from a list

```
import random

my_backpack = ['gum', 'pencil', 'dongle', 'book', 'charging cable']

print(random.choice(my_backpack))
```

secrets module

- New in Python 3.6
- Cryptographically strong random numbers
  - Suitable for managing data such as passwords, account authentication, security tokens, and related secrets

```python
import secrets

print(secrets.choice(range(1, 1_000_000)))
print(secrets.randbelow(100))
```