Computer Programming Summer Interest Group

Session 6 / Collections

7/23/2025

Luke 12:13-21

The Parable of the Rich Fool

Someone in the crowd said to him, “Teacher, tell my brother to divide the inheritance with me.” Jesus replied, “Man, who appointed me a judge or an arbiter between you?” Then he said to them, “Watch out! Be on your guard against all kinds of greed; life does not consist in an abundance of possessions.”

And he told them this parable: “The ground of a certain rich man yielded an abundant harvest. He thought to himself, ‘What shall I do? I have no place to store my crops.’

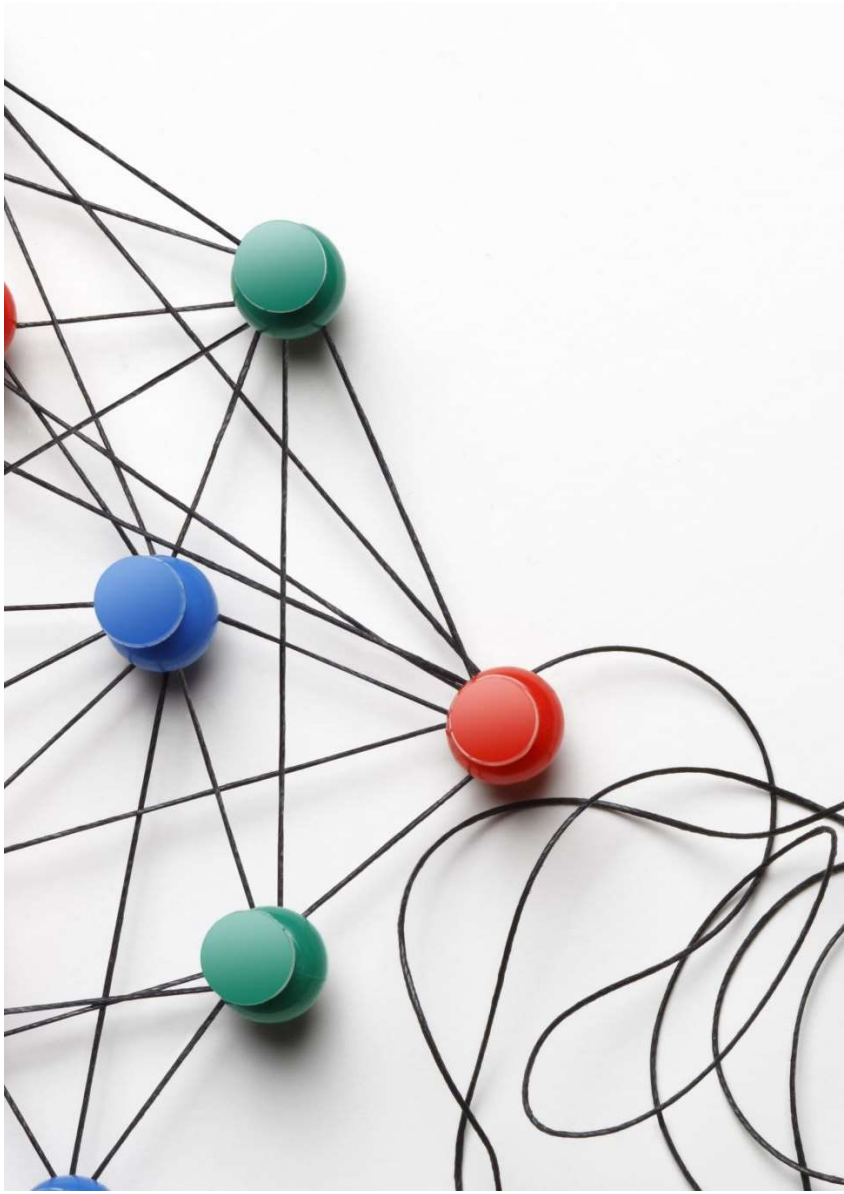
“Then he said, ‘This is what I’ll do. I will tear down my barns and build bigger ones, and there I will store my surplus grain. And I’ll say to myself, “You have plenty of grain laid up for many years. Take life easy; eat, drink and be merry.”’

“But God said to him, ‘You fool! This very night your life will be demanded from you. Then who will get what you have prepared for yourself?’

“This is how it will be with whoever stores up things for themselves but is not rich toward God.”

Objectives

- By the end of this session, you will be able to:
  - Understand common Python collection types
  - Create, modify, and access elements python collections
  - Use common collection operations
  - Iterating over collections
  - Understand mutability of lists and dictionaries
  - Understand immutability in dictionaries and sets



Lists

- A List is an ordered, mutable collection of objects
  - The type of objects in the list is arbitrary
  - The contents of the list can change over time
- Each object in the list can be referenced by its location in the list
- New objects can be inserted anywhere in the list
- Objects in the list can be removed from anywhere in the list

List Manipulation - access

```
fruits = ["apple", "banana", "cherry"]

print(fruits[0])   # apple
print(fruits[-1])  # cherry


lol = [[1, 2], [3, 4]]

print(f"{lol=} / {len(lol)=}")
```

List Manipulation – functions and methods

```
fruits = ["apple", "banana", "cherry"]

fruits.append("orange")  # add orange to the end of the list
print(f'after fruits.append("orange"): {fruits=} - {len(fruits)=}')

fruits.insert(1, "blueberry")  # Insert blueberry after apple
print(f'after fruits.insert(1, "blueberry"): {fruits=} - {len(fruits)=}')

fruits.remove("banana")  # Remove banana from the list
print(f'after fruits.remove("banana"): {fruits=} - {len(fruits)=}')

last = fruits.pop()  # Remove the last element from the list
print(f"after fruits.pop(): {fruits=} - {len(fruits)=}")
print(f"after fruits.pop(): {last=}")
```

List Manipulation – Iteration over lists

```
fruits = ["apple", "blueberry", "banana", "cherry", "orange"]

for fruit in fruits:
    print(fruit)

for index, fruit in enumerate(fruits):
    print(index, fruit)

for index, fruit in enumerate(fruits[::-1]):
    print(index, fruit)
```

Tuples

- A tuple is an ordered, immutable collection of objects
  - The type of objects in the tuple is arbitrary
  - The contents of the tuple cannot change
- Each object in the tuple can be referenced by its location

Tuple Operations

```
fruits = ("apple", "banana", "cherry“)

print(fruits[0])   # apple
print(fruits[-1])  # cherry

for fruit in fruits:
    print(fruit)


x, y, z = fruits
print(f”{x=} {y=} {z=}”)


tot = ((1, 2), (3, 4))
print(f"{lol=} / {len(lol)=}")

tot[0] = “food”  # Will it run?
```

Set

- A set is an unordered, mutable collection of immutable objects
  - The elements of the tuple must individually be immutable
  - The contents of the set can change over time
- A set contains no duplicate elements
- New elements can be added to the set
- Elements in the set can be removed
- Location of elements in the set is not specified

Set Manipulation

```
fruits = {"apple", "banana", "cherry“}

print(fruits[0])   # Not indexable

for fruit in fruits:
    print(fruit)


sos = {{1, 2}, {3, 4}}  # Invalid
sol = {[1, 2], [3, 4]}  # Invalid
sot = {(1, 2), (3, 4)}  # Valid
```

Set Operations

```
fruits = {"apple", "cherry", "banana"}
trees = {"birch", "apple", "maple"}

print(fruits.union(trees))
print(fruits.intersection(trees))
print(fruits.difference(trees))
print(trees.difference(fruits))
print(trees.symmetric_difference(fruits))
```

Dictionary

- A dictionary is a collection of key-value pairs
- Dictionary keys must be comprised of immutable values
- Dictionary values can be of any type
- Key-value pairs can be added and removed
- Values associated with a key can change
- No two distinct entries can have the same key

Dictionary Access

```
student = {
    "name": "Jordan",
    "age": 18,
    "grade": "A"
}

print(student["name"])
print(student)

student["age"] = 19
print(student)

student["major"] = "Math"
print(student)

del student["grade"]
print(student)
```

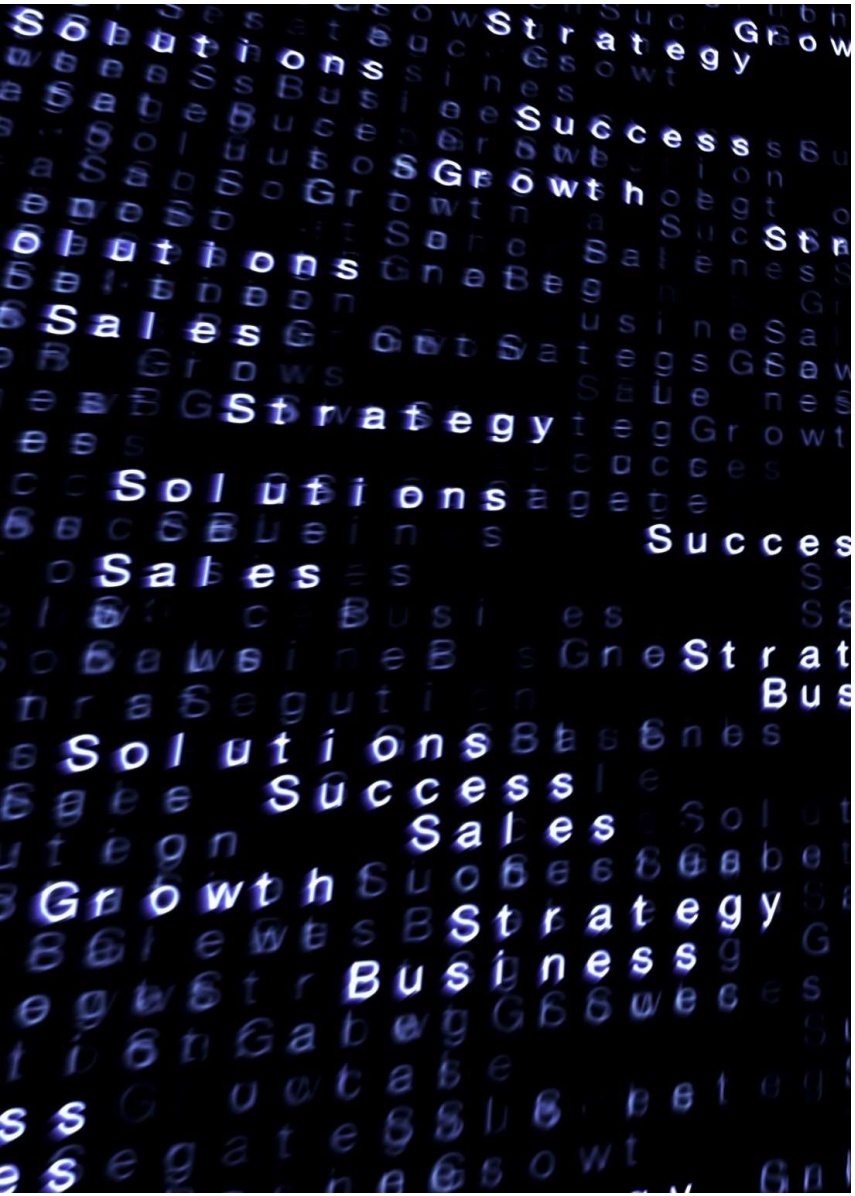
Dictionary Methods

```
student = {
    "name": "Jordan",
    "age": 18,
    "grade": "A"}

print(student.get("name"))
print(student.get("address"))
print(student.get("address“, “dunno”))

print(list(student.keys()))
print(list(student.values()))
print(list(student.items()))

for key, value in student.items():
    print(key, value)
```



Activities

- List & tuple
  - Create a list of 5 of your favorite foods not listed alphabetically
  - Sort them into alphabetic order
  - Place the sorted list into a tuple (hint: use the ‘tuple’ function)
- Set & dictionary
  - Uses a dictionary to store student names and their favorite subjects.
  - Use a set to list all of the subjects without duplication