Collections

The Topic: What?

- An introduction to collections in Python
  - What Collections are used for
  - Ordered and Unordered Collections
  - Tuples, Lists, Dictionaries, Sets

Applications: Why?

- To store related data in groups (collections!)
- To know the differences between different types of collection
- To determine which collection is appropriate depending on the scenario.

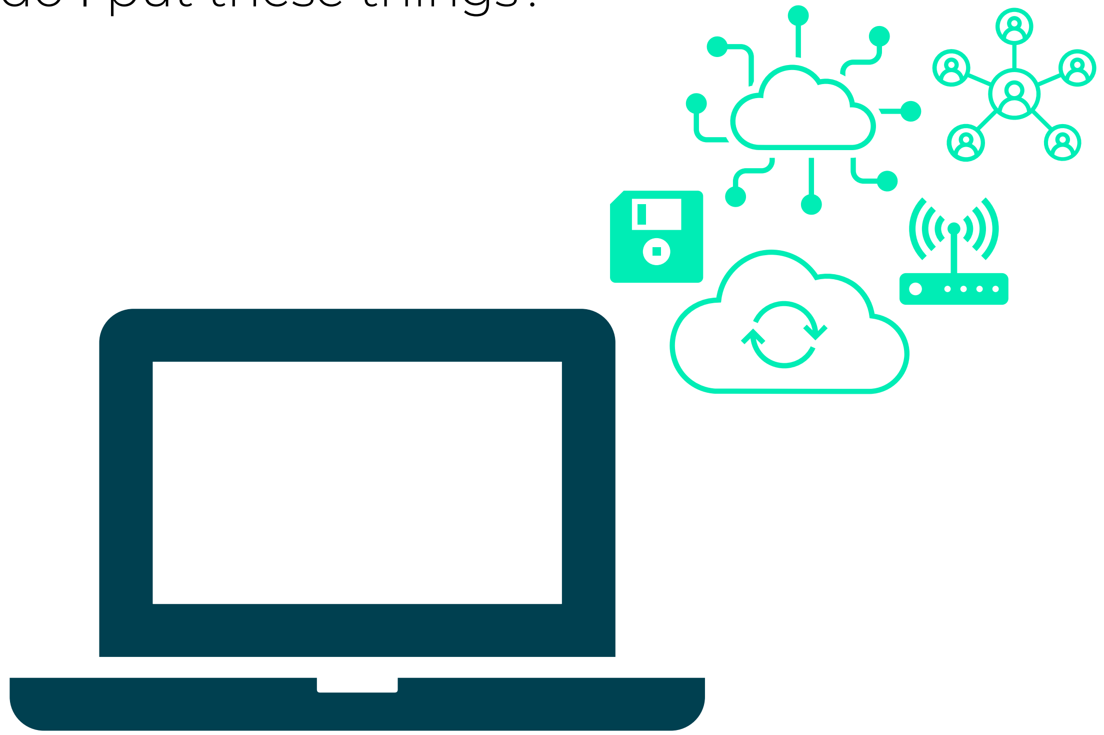
Expectations: Who?

- Learners are expected to have covered fundamental programming in Python previously.

Collections

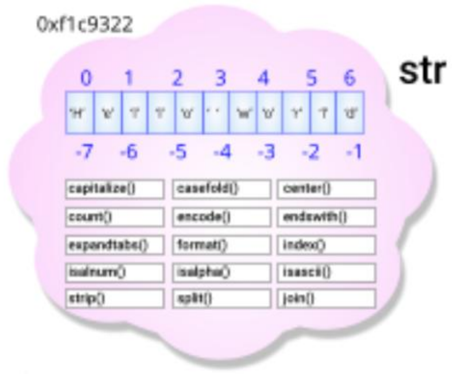
Storing related data in an ordered or unordered way

Where do I put these things?

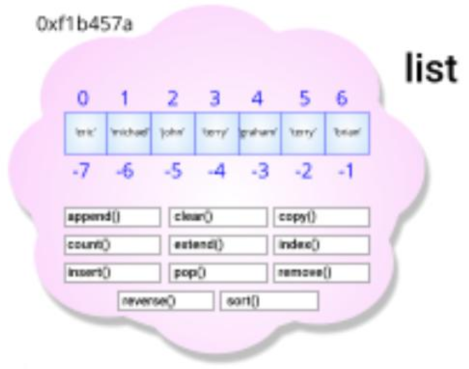


Tuples, Lists, Dictionaries, Sets

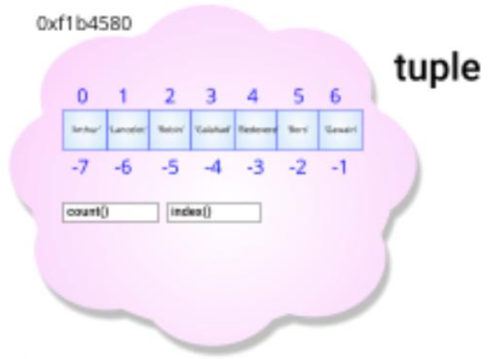
Ordered collections


0xf1c9322
str
0 1 2 3 4 5 6
'H' 'e' 'l' 'l' 'o' ' ' 'w' 'o' 'r' 'l' 'd'
-7 -6 -5 -4 -3 -2 -1
capitalize() casefold() center()
count() encode() endswith()
expandtabs() format() index()
isalnum() isalpha() isascii()
strip() split() join()

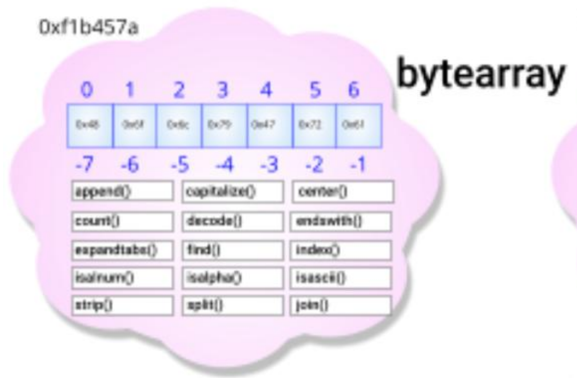
Ordered collection of characters
Immutable = Read Only
Index using [n] and [-n]
Slicing using [start:end]
message = "hello world"
Convert to string using str()



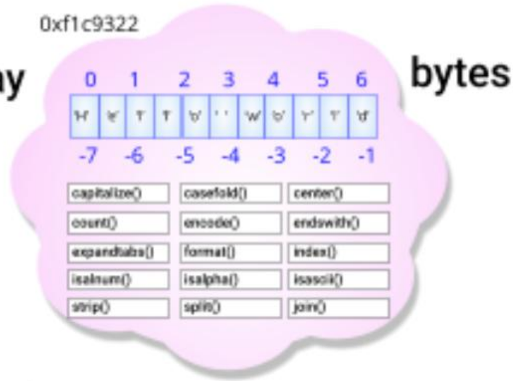
Ordered collection of objects
Mutable = Read/Write
Index using [n] and [-n]
Slicing using [start:end]
knights = ['eric', 'john', 'terry']
Dynamic and flexible (many methods)
Convert to list using list()



Ordered collection of objects
Immutable = Read Only
Index using [n] and [-n]
Slicing using [start:end]
knights = ;eric', 'john', 'terry'
Simple and fast
Convert to tuple using tuple()

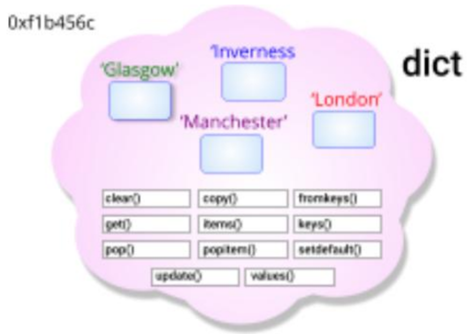

0xf1b457a
bytearray
0 1 2 3 4 5 6
-7 -6 -5 -4 -3 -2 -1
append() capitalize() center()
count() decode() endswith()
expandtabs() find() index()
isalnum() isalpha() isascii()
strip() split() join()

Ordered collection of single bytes/raw binary
Mutable = Read/Write
Index using [n] and [-n]
Slicing using [start:end]
ba_message= bytearray([0x680a, 0x6f, 0x6c])
Dynamic and flexible (many methods)
Convert to array of bytes using bytearray()

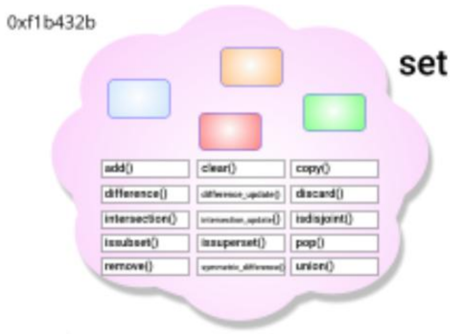

0xf1c9322
bytes
0 1 2 3 4 5 6
'H' 'e' 'l' 'l' 'o' ' ' 'W' 'o' 'r' 'l' 'd'
-7 -6 -5 -4 -3 -2 -1
capitalize() casefold() center()
count() encode() endswith()
expandtabs() format() index()
isalnum() isalpha() isascii()
strip() split() join()

Ordered collection of single bytes/raw binary
Immutable = Read Only
Index using [n] and [-n]
Slicing using [start:end]
message = b"hello world"
Convert to bytes using bytes()

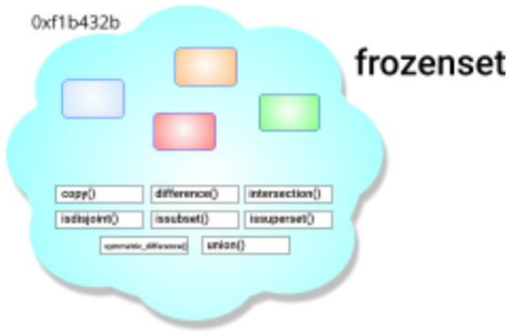
Unordered collections



Unordered/ordered(Py3.6) collection of characters
Mutable = Read/write
Index using ['key']
Keys are unique
Dynamic and faster searching
cities = {'Glasgow': 19, 'Inverness': 21}
Convert to dictionary using dict()

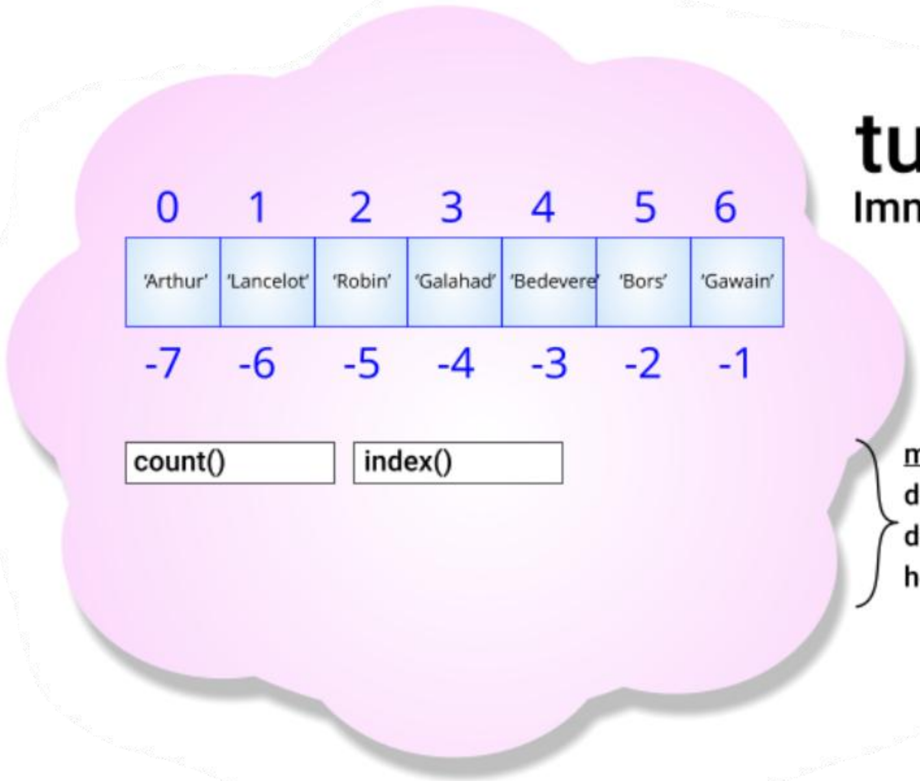


Unordered collection of objects
Mutable = Read/Write
No indexes
Objects are unique (no duplicates)
Can combine with other sets using SET operators
knights = {'eric', 'john', 'terry', 'terry'}
Convert to set using set()



Unordered collection of objects
Immutable = Read Only
No indexes
Objects are unique (no duplicates)
Can combine with other sets using SET operators
knights = frozenset({'eric', 'john', 'terry', 'terry'})
Convert to frozen set using frozenset()

Tuples



tuple

Immutable = Read Only

methods
dir(tuple)
dir(knights)
help(tuple)

Practice

Estimated time: 10 minutes

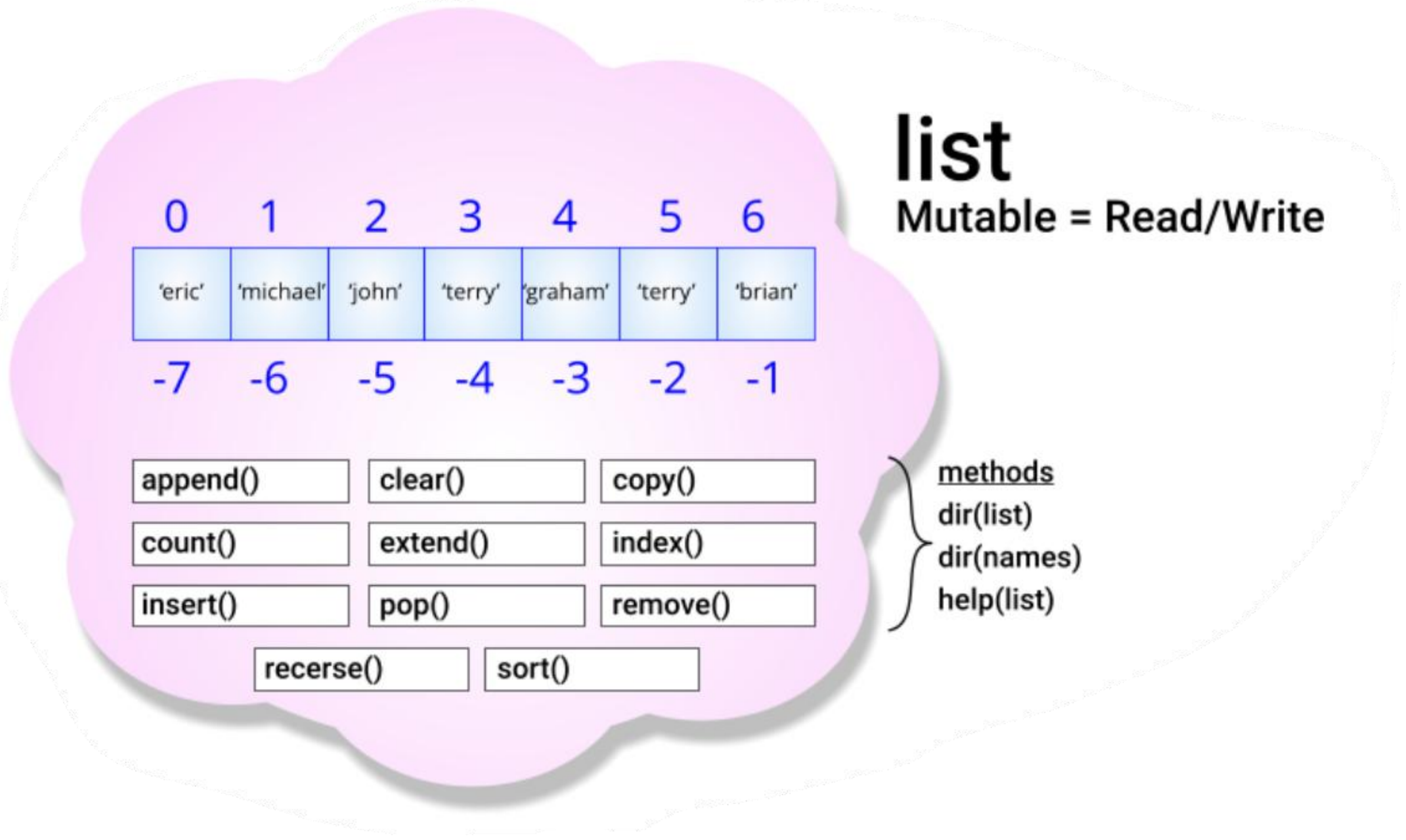
Try these statements at the Python shell and explain what is happening?

```
>>> a = 10
>>> b = a
>>> (a, b) = (b, a)
>>> print(a)
>>> print(b)
>>> (a, b, c) = range(0, 3)
>>> print(a, b, c)
>>> (a, b, c) = (10, 20, 30, 40, 50)
>>> (a, b, *c) = (10, 20, 30, 40, 50) # This is called unpacking and is useful when LHS and RHS are not in balance.
>>> print(a, b)
>>> print(c)
>>> comment = ("Tis but a scratch")
>>> print(type(comment))
>>> comment = ("Tis but a scratch", )
>>> print(type(comment))
```

Trainer demonstration

demo_tuples.py

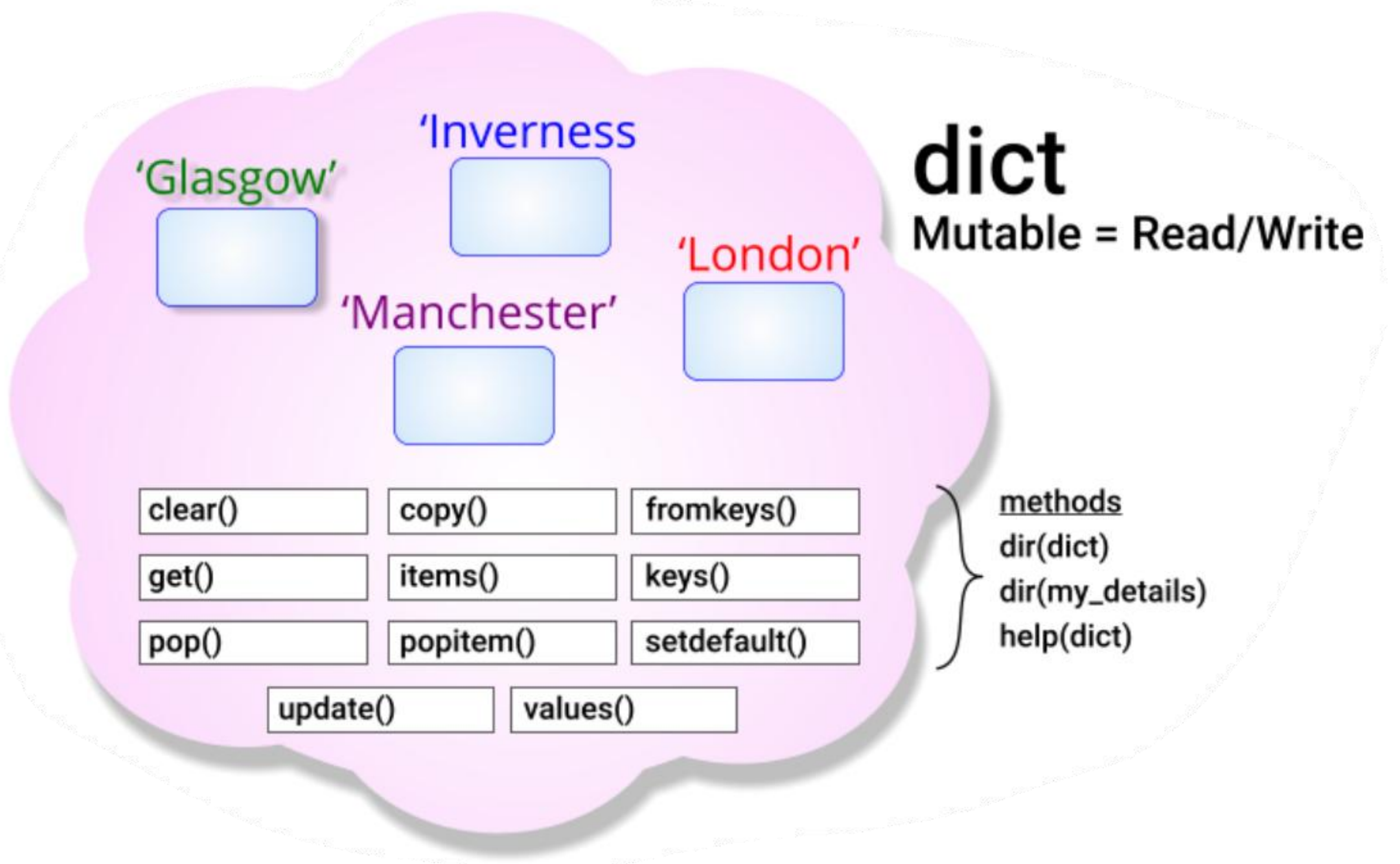
Lists


0
1
2
3
4
5
6
'eric'
'michael'
'john'
'terry'
'graham'
'terry'
'brian'
-7
-6
-5
-4
-3
-2
-1
append()
clear()
copy()
count()
extend()
index()
insert()
pop()
remove()
recerse()
sort()
list
Mutable = Read/Write
methods
dir(list)
dir(names)
help(list)

Trainer demonstration

demo_lists.py

Dictionaries

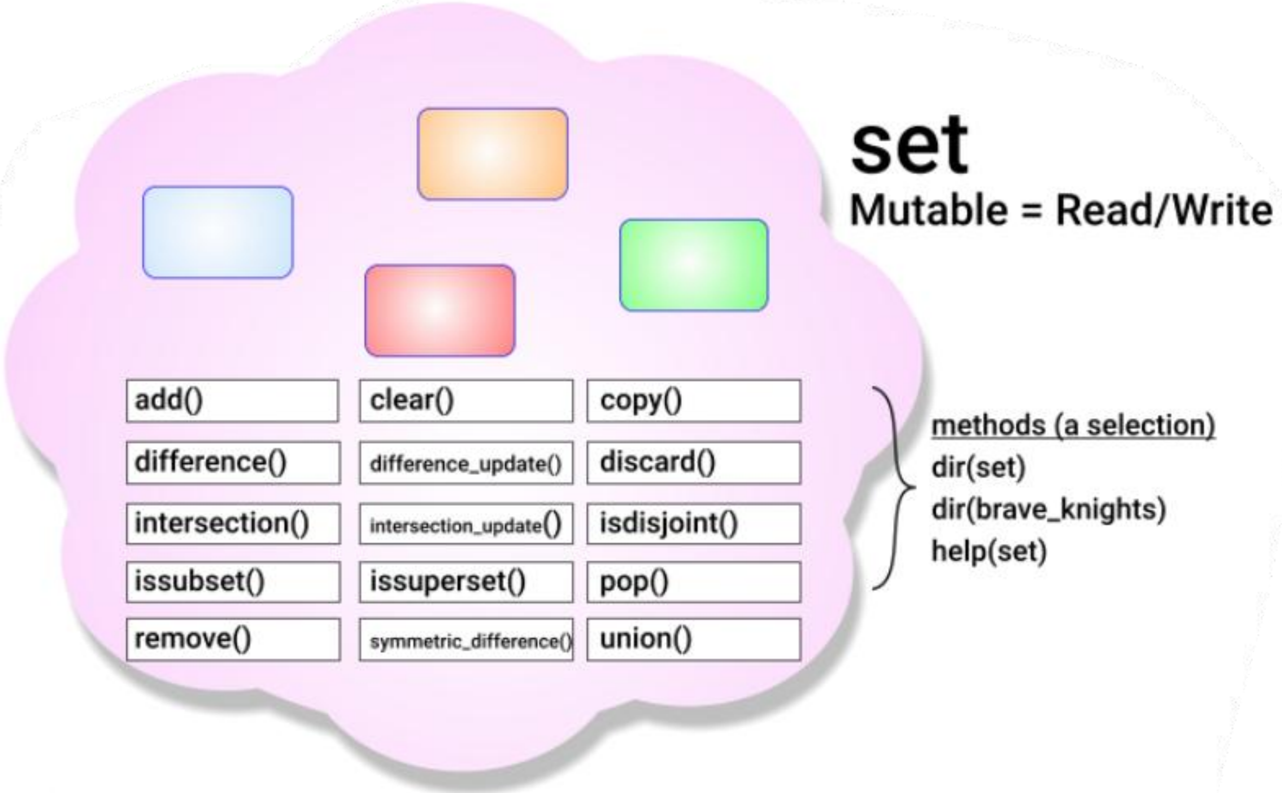


Since 2016, Python 3.6 stores dictionaries as ordered - in the order that the key: objects are assigned, but this may change in a future implementation, so don't assume!

Trainer demonstration

demo_dictionaries.py

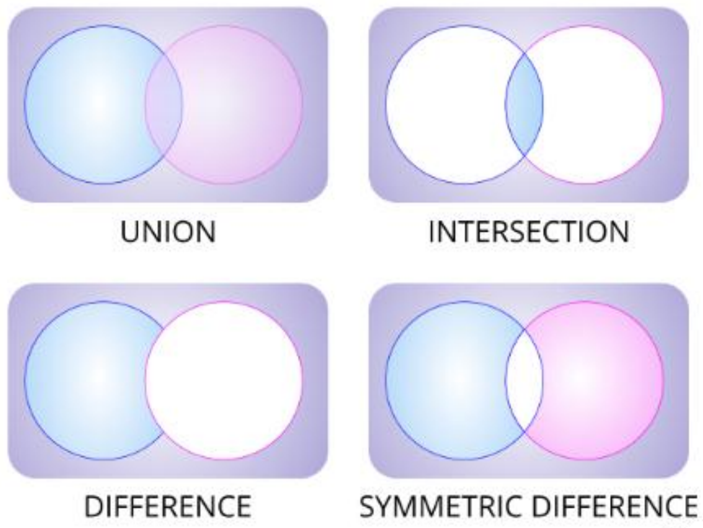
Sets



add()	clear()	copy()
difference()	difference_update()	discard()
intersection()	intersection_update()	isdisjoint()
issubset()	issuperset()	pop()
remove()	symmetric_difference()	union()

methods (a selection)
dir(set)
dir(brave_knights)
help(set)

Venn Diagrams



Trainer demonstration

demo_sets.py

There are many Brave Knights who would also like to be Lumberjacks and vice-versa, so we will attempt to combine sets of Knights and sets of Lumberjacks using common SET operators and methods.

Learning check

5-10 mins

1. **Name three ordered and two unordered collections?**

Ordered?	Unordered

2. **Is a tuple mutable?**

True ☐ False ☐

3. **The joy of sets! What would be printed from the following code?**

```
>>> scottish_movies = {"Gregory's Girl", "Local Hero", "Comfort and Joy", "That Sinking Feeling"}
>>> fav_movies = {"The Hobbit", "Local Hero", "The Lord of the Rings"}
>>> print(scottish_movies & fav_movies)
```

4. **Fill in the blanks:**

Dictionaries have unique ______

Sets have unique ______

5. **What would be printed from the following code?**

```
>>> movies = "Gregory's Girl", "Local Hero", "Comfort and Joy", "That Sinking Feeling"
>>> x, *y, z = movies
>>> print(y)
```

6. **Write the print statement from question 5 using set methods?**

Solutions

Collections quiz

1. Name three ordered and two unordered collections?

Answer: Ordered: str, tuple, list

Unordered: dict, set

NB. Dictionaries are insertion ordered from Python 3.6+.

2. Fill in the blanks:

Answer: Dictionaries have unique keys

Sets have unique objects

3. Is a tuple mutable?

Answer: False

4. What would be printed from the following code?

```
>>> movies = "Gregory's Girl", "Local Hero", "Comfort and Joy", "That Sinking Feeling"
>>> x, *y, z = movies
>>> print(y)
```

Answer: ['Local Hero', 'Comfort and Joy']

5. The joy of sets! What would be printed from the following code?

```
>>> scottish_movies = {"Gregory's Girl", "Local Hero", "Comfort and Joy", "That Sinking Feeling“}
>>> fav_movies = {“The Hobbit”, “The Lord of the Rings”, “Local Hero”}
>>> print(scottish_movies & fav_movies)
```

Answer: {'Local Hero'}

6. Write the print statement from question 5 using set methods (rather than set operators)?

Answer: print(scottish_movies.intersection(fav_movies)

Labs

1. Create a new script in **C:\labs\** called **topTen.py**

a. Read the top 250 movies from the **C:\labs\top250_movies.txt** file and store them in a list called movies.

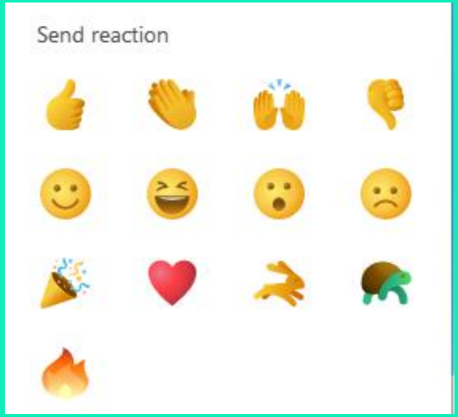
b. Print out the top 10 with rankings (right justified 5 characters) followed by a colon and space followed by name of the film.

c. Print out the first and last movie.

d. Modify the code to allow the user to choose the top N to be displayed.

2. Write a Python script called **C:\labs\lotto.py** that will generate and display 6 unique random lottery numbers between 1 and 50. Think about which Python data structure is best suited to store the numbers! Use the Python help() function to find out which function to use from the python standard library called random.

Stretch Exercises 3-7



END OF SECTION

An introduction to collections in Python

- What Collections are used for
- Ordered and Unordered Collections
- Tuples, Lists, Dictionaries, Sets

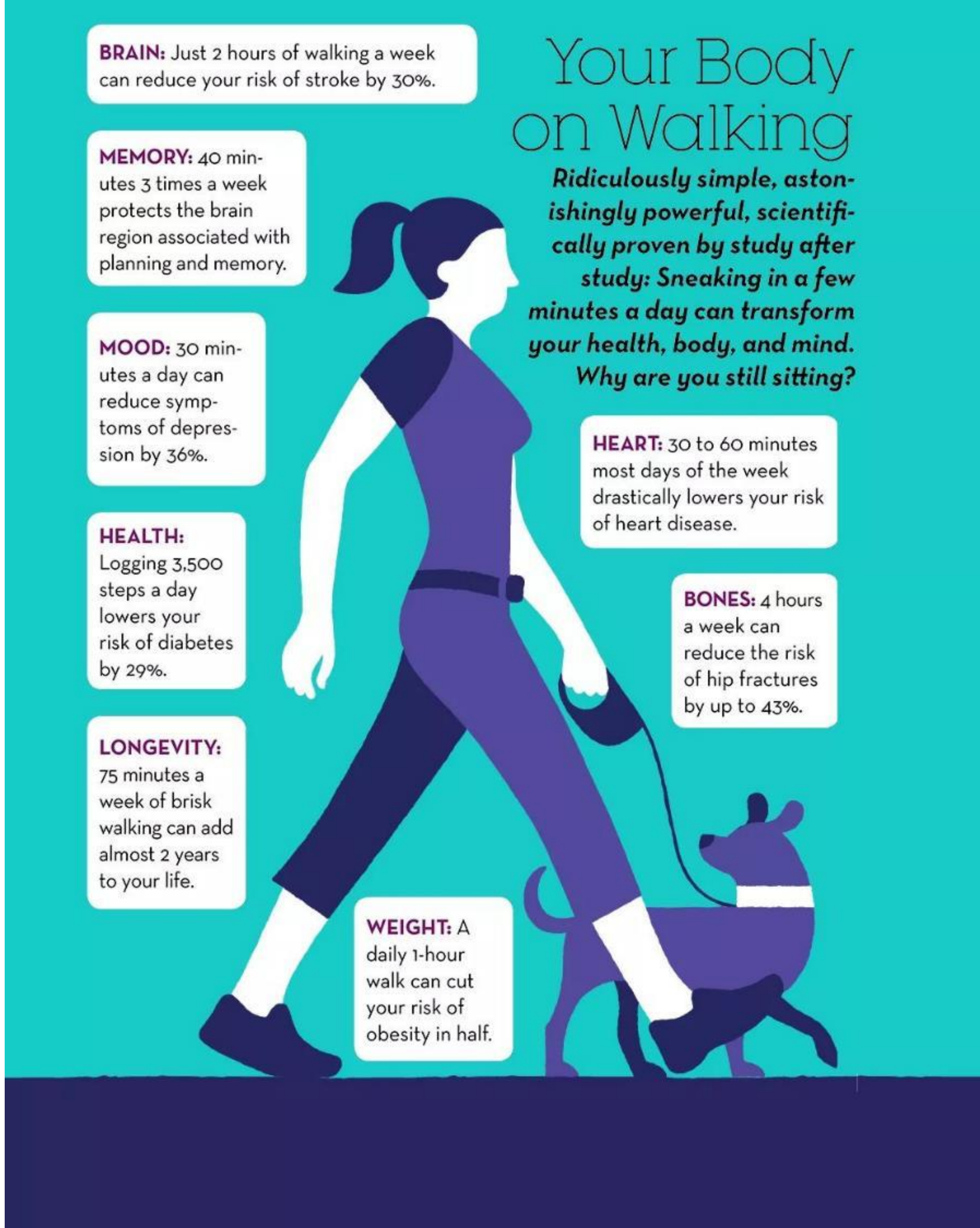
•To store related data in groups (collections!)

•To know the differences between different types of collection

•To determine which collection is appropriate depending on the scenario.

REMINDER: TAKE A BREAK!

10.30 - 10.40
11.40 – 11.50
12.50 – 13.30
14.30 – 14.40
15.40 – 15.50



Your Body on Walking

Ridiculously simple, astonishingly powerful, scientifically proven by study after study: Sneaking in a few minutes a day can transform your health, body, and mind. Why are you still sitting?

BRAIN: Just 2 hours of walking a week can reduce your risk of stroke by 30%.

MEMORY: 40 minutes 3 times a week protects the brain region associated with planning and memory.

MOOD: 30 minutes a day can reduce symptoms of depression by 36%.

HEALTH: Logging 3,500 steps a day lowers your risk of diabetes by 29%.

LONGEVITY: 75 minutes a week of brisk walking can add almost 2 years to your life.

WEIGHT: A daily 1-hour walk can cut your risk of obesity in half.

HEART: 30 to 60 minutes most days of the week drastically lowers your risk of heart disease.

BONES: 4 hours a week can reduce the risk of hip fractures by up to 43%.