

# Python Tutorial – Part I: Introduction

Mark A. Austin

University of Maryland

*austin@umd.edu*

*ENCE 688P, Spring Semester 2022*

January 18, 2023

# Overview

- 1 What is Python?
  - Origins, Features, Framework for Scientific Computing
- 2 Program Development with Python
  - Working with the Terminal
  - Integrated Development Environments
- 3 Data Types
- 4 First Program (Evaluate and Plot Sigmoid Function)
- 5 Builtin Collections (Lists, Dictionaries, and Sets)
- 6 Numerical Python (NumPy)
- 7 Data and Dataset Transformation (Pandas)

## Part 3

# Builtin Containers and Collections

(Working with Lists, Dictionaries, Sets)

# Builtin Containers and Collection

## Containers and Collections

A **container** is an object that **stores objects**, and provides a way to **access** and **iterate** over them. **Collections** are **container data types**, namely lists, sets, tuples, dictionary.

### Builtin Collection Data Types:

- **List:** A list is a collection which is ordered and changeable.
- **Dictionary:** A dictionary is a collection which is ordered and changeable. No duplicate members.
- **Set:** A set is a collection which is unordered, unchangeable and unindexed. No duplicate members.
- **Tuple:** A tuple is a collection which is ordered and unchangeable.

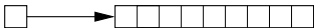
# Working with Lists

## List

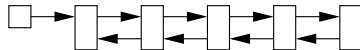
Lists are used to **store multiple items** in a **single variable**. A list may store **multiple types** (heterogeneous) of **elements**.

## Array, List, HashMap, and Queue Structures

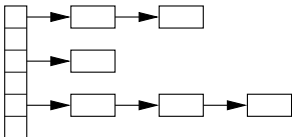
### Arrays



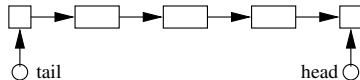
### Linked List



### Hash Map



### Queues



# Working with Lists

## Basic List Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list.
index()	Returns the index of the first element with the specified value.
insert()	Adds an element at the specified position.
remove()	Removes the item with the specified value.
reverse()	Reverses the order of the list.
sort()	Sorts the list.

# Working with Lists

## Example 1: Create working lists ...

```
list01 = [ "apple", "orange", "avocado", "banana", "grape", "watermelon"]
list02 = [ "apple", "avocado", "banana", "banana", "grape", "watermelon"]

print ("--- List01: %s ..." %( list01 ))
print ("--- List02: %s ..." %( list02 ))

# Create list with mix of data types ...

list03 = [ "apple", 40, True, 2.5 ]

print ("--- List03 (with multiple data types): %s ..." %( list03 ))
```

## Output:

```
--- List01: ['apple', 'orange', 'avocado', 'banana', 'grape', 'watermelon'] ...
--- List02: ['apple', 'avocado', 'banana', 'banana', 'grape', 'watermelon'] ...

--- List03 (with multiple data types): ['apple', 40, True, 2.5] ...
```

# Working with Lists

## Example 2: Access list items ...

```
list04 = list( ( "apple", 40, True, 2.5, False ) )

print ( "---- list04[0]: %s ..." %( list04[0] ) )
print ( "---- list04[1]: %s ..." %( list04[1] ) )
print ( "---- list04[2]: %s ..." %( list04[2] ) )
print ( "---- list04[3]: %s ..." %( list04[3] ) )
print ( "---- list04[4]: %s ..." %( list04[4] ) )
```

## Output:

```
---- list04[0]: apple ...
---- list04[1]: 40 ...
---- list04[2]: True ...
---- list04[3]: 2.5 ...
---- list04[4]: False ...
```

**Source Code:** See: [python-code.d/collections/](#)



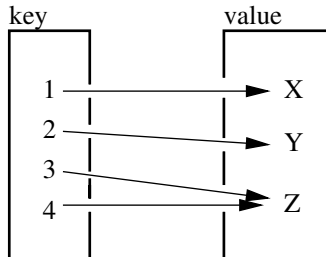
# Working with Dictionaries

## Dictionary

Dictionaries **store data values** as **key:value pairs**. As of Python 3.7, a dictionary is a collection which is ordered, changeable and do not allow duplicates.

## Key:Value Map Operations

### Maps



# Working with Dictionaries

## Basic Dictionary Methods

Method	Description
clear()	Removes all the elements from the dictionary.
copy()	Returns a copy of the dictionary.
fromkeys()	Returns a dictionary with the specified keys and value.
get()	Returns the value of the specified key.
items()	Returns a list containing a tuple for each key value pair.
keys()	Returns a list containing the dictionary's keys.
pop()	Removes the element with the specified key.
popitem()	Removes the last inserted key-value pair.
update()	Updates the dictionary with the specified key-value pairs.
values()	Returns a list of all the values in the dictionary.

# Working with Dictionaries

**Example 1:** Create dictionary of car attributes.

```
car01 = { "brand": "Honda",          # <-- Create simple dictionary ....
          "model": "Acura",
          "miles": 25000,
          "new": False,
          "year": 2016
        }

print ("--- Car01: %s ..." %( car01 )) # <-- print dictionary ...
```

**Output:** Print simple dictionary.

```
--- Car01: {'brand': 'Honda', 'model': 'Acura',
           'miles': 25000, 'new': False, 'year': 2016} ...
```

# Working with Dictionaries

## Example 2: Systematically access items in Car01 ...

```
print ("--- Car01: brand --> %s ..." %( car01.get("brand") ))
print ("---          : model --> %s ..." %( car01.get("model") ))
print ("---          : miles --> %d ..." %( car01.get("miles") ))
print ("---          : new   --> %s ..." %( car01.get("new") ))
print ("---          : year  --> %d ..." %( car01.get("year") ))
```

## Output:

```
--- Access items in Car01 ...
--- Car01: brand --> Honda ...
---          : model --> Acura ...
---          : miles --> 25000 ...
---          : new   --> False ...
---          : year  --> 2016 ...
```

**Source Code:** See: [python-code.d/collections/](https://python-code.d/collections/)

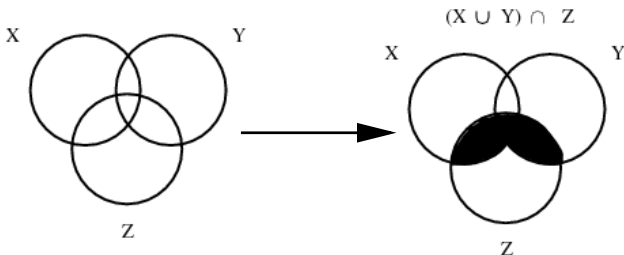
# Working with Sets

## Sets

Sets store **multiple items** in a **single variable**. A set is a collection which is unordered, unchangeable (but you can remove items and add new items) and unindexed.

## Set Operations

Sets



# Working with Sets

## Basic Set Methods

Method	Description
add()	Adds an element to the set.
clear()	Removes all the elements from the set.
copy()	Returns a copy of the set.
discard()	Remove the specified item.
intersection()	Returns a set, that is the intersection of two other sets.
remove()	Removes the specified element.
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others.

# Working with Sets

## Example 1: Create working sets; set operations ...

```
set01 = { 1, 2, 3, 4, 5, 6, 7 }
set02 = { 6, 7, 8, 9, 10 }
set03 = {"apple", "banana", "cherry"}
set04 = {True, False, False}

print ("--- Set01.union(Set02) : %s ..." %( set01.union(set02) ))
print ("--- Set01.intersection(Set02) : %s ..."
      %( set01.intersection(set02) ))
```

## Output:

```
--- Create working sets ...
--- Set01: {1, 2, 3, 4, 5, 6, 7} ...
--- Set02: {6, 7, 8, 9, 10} ...
--- Set03: {'cherry', 'banana', 'apple'} ...
--- Set04: {False, True} ...

--- Set01.union(Set02) : {1, 2, 3, 4, 5, 6, 7, 8, 9, 10} ...
--- Set01.intersection(Set02) : {6, 7} ...
```

# Working with Sets

**Example 2:** Add items to set03, then print ...

```
set03.add("strawberry")
set03.add("kiwi")
print ("--- Set03 (appended): ...")
for x in set03:
    print ("---  %s ..." %(x))
```

**Output:** Set03 appended ...

```
---  cherry ...
---  strawberry ...
---  banana ...
---  kiwi ...
---  apple ...
```

**Source Code:** See: [python-code.d/collections/](https://python-code.d/collections/)