



Python Fundamentals, Data Structures, and Algorithms:

Week 3 Workshop
Presentation



Workshop Agenda

Activity	Estimated Duration
Set up and check in	10 mins
Week 3 Review	60 mins
Assignment Tasks 1-3	50 mins
Break	15 mins
Remaining Assignment Tasks	100 mins
Check-Out (Feedback & Wrap-Up)	15 mins



Week 3 Review



Overview

Lists	For Loops
List Index	Strings
Bracket Notation	Dictionaries
Slicing Notation	Tuples
The In Keyword	Sets



Review: Lists

1. `["Charlie", "Alpha", "Delta", "Bravo"]`
2. `[]`
3. `[35, 57, 57, 211, 57, 232]`
4. `["nucamp", 0, 12.5, 'Echo']`

Which of these is *not* a valid list?



Review: Lists

Answer:

They are all valid.



Review: List index

```
["Charlie", "Alpha", "Delta", "Bravo"]
```

What is the index of "Delta"?



Review: List index

```
["Charlie", "Alpha", "Delta", "Bravo"]
```

The index of "Delta" is 2.



Review: Bracket notation

- Used with all indexed Python data structures

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]  
print(my_list[?])
```

- To print "Charlie" to the terminal, what would you put inside the brackets?



Review: Bracket notation

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]  
print(my_list[0])
```

- Answer: This would print **"Charlie"** to the screen.



Review: Bracket notation

- You can also use bracket notation to modify list values

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]  
my_list[0] = "Echo"  
print(my_list)
```

- What is printed to the terminal by the code above?



Review: Bracket notation

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]  
my_list[0] = "Echo"  
print(my_list)
```

ANSWER: ["Echo", "Alpha", "Delta", "Bravo"]



Review: Using lists

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]
```

1. `len(my_list)`
2. `my_list.append("Echo")`
3. `x = my_list.pop()`
4. `x = my_list.pop(2)`



Review: Using lists

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]
```

1. `len(my_list)` Length is 4 (number of items)
2. `my_list.append("Echo")` "Echo" added to end of list
3. `x = my_list.pop()` "Echo" removed from list, x = "Echo"
4. `x = my_list.pop(2)` "Delta" removed from list, x = "Delta"



Review: Slicing notation

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]
```

What part of the list is "sliced" by...

1. `my_list[:3]`
2. `my_list[2:]`
3. `my_list[1:3]`



Review: Slicing notation

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]
```

What part of the list is "sliced" by...

1. `my_list[:3]` `['Charlie', 'Alpha', 'Delta']`
2. `my_list[2:]` `['Delta', 'Bravo']`
3. `my_list[1:3]` `['Alpha', 'Delta']`



Review: The in keyword

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]
```

- 1)
print("Alpha" in my_list)
- 2)
if "Delta" not in list:
 - print("No Delta")
 - else:
 - print("Delta")
- 3)
print("Echo" in my_list)



Review: The in keyword

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]
```

- 1)
print("Alpha" in my_list) True
- 2)
if "Delta" not in list:
• print("No Delta")
• else:
• print("Delta") "Delta"
- 3)
• print("Echo" in my_list) False



Review: For loops

```
my_list = ["Charlie", "Alpha", "Delta", "Bravo"]
```

```
>>> for word in my_list:
...     print(word)
...
Charlie
Alpha
Delta
Bravo
>>> █
```

```
>>> for idx in range(0, len(my_list), 1):
...     print(my_list[idx])
...
Charlie
Alpha
Delta
Bravo
```



Review: Strings

- Only primitive data type also considered a data structure
- Contain ordered sequences of characters
 - Characters can be letters, punctuation, numbers, whitespace.
- Strings are immutable:



```
>>> lang = "Python"  
>>> print(lang[0])  
P
```



```
>>> lang[0] = "M"  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: 'str' object does not support item assignment
```



```
>>> lang.append("s")  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
AttributeError: 'str' object has no attribute 'append'
```



Review: Strings

```
>>> for char in "Bravo":  
...     print(char)  
...  
B  
r  
a  
v  
o
```



Review: Dictionaries

- Dictionaries contain key-value pairs
- Ordered sequence since Python 3.6, previous versions unordered
- Keys must be unique, values do not have to be
- Adding a duplicate key will overwrite existing key
- You can use bracket notation with the key to retrieve a value



Review: Dictionaries

- `ingredients = {"butter": "1 stick", "flour": "2 cups", "salt": "1 tsp"}`
1. How would you retrieve the value associated with the key `"salt"`?
 2. How would you replace the value associated with the key `"flour"` with `"2.5 cups"`?



Review: Dictionaries

- `ingredients = {"butter": "1 stick", "flour": "2 cups", "salt": "1 tsp"}`

1. `ingredients["butter"]`

2. `ingredients["flour"] = "2.5 cups"`



Review: Iterating Dictionaries

- `popcorn_prices = {"small": 1.5, "medium": 3.5, "large": 4}`

- 1) `for size in popcorn_prices.keys():`
 `print(size)`

- 2) `for price in popcorn_prices.values():`
 `print(price)`

- 3) `for size, price in popcorn_prices.items():`
 `print(size, price)`



Review: Iterating Dictionaries

- popcorn_prices = {"small": 1.5, "medium": 3.5, "large": 4}

1) for size in popcorn_prices.keys():
 print(size)

small
medium
large

2) for price in popcorn_prices.values():
 print(price)

1.5
3.5
4

3) for size, price in popcorn_prices.items():
 print(size, price)

small 1.5
medium 3.5
large 4



Review: Tuples

- Tuples are immutable lists
- Immutable means cannot be changed

Given the following tuple:

```
tuple1 = (1, 10, 100, 1000)
```

Will either of these statements work without an error?

1. `tuple1[0] = 2`
2. `tuple1 = (2, 20, 200, 2000)`



Review: Tuples

- Tuples are immutable lists
- Immutable means cannot be changed

Given the following tuple:

```
tuple1 = (1, 10, 100, 1000)
```

Will either of these statements work without an error?

1. `tuple1[0] = 2` **NO**
2. `tuple1 = (2, 20, 200, 2000)` **YES**



Review: Tuples

Which of the following is *not* a valid way to declare a tuple?

1. `tuple1 = ("Charlie", "Alpha", "Delta", "Bravo")`
2. `tuple2 = "Alpha", "Echo", "Bravo"`
3. `tuple3 = ("Delta")`
4. `tuple4 = ()`



Review: Tuples

Which of the following is *not* a valid way to declare a tuple?

1. `tuple1 = ("Charlie", "Alpha", "Delta", "Bravo")`
2. `tuple2 = "Alpha", "Echo", "Bravo"`
3. `tuple3 = ("Delta")` **INVALID**
4. `tuple4 = ()`



Review: Sets

- Unordered collection of values
 - Duplicates are removed
 - Sets are mutable but can only contain immutable data types
-
- Questions:
 1. Which of Python's built-in data types can sets **not** contain?
 2. Which is the correct way to create an empty set?
 - a. `my_set = {}`
 - b. `my_set = set()`



Review: Sets

Example: `my_set = {4, 23, 67, 1}`

Use the method `add()` to add a new item to a set:

`my_set.add(55)`

Use the method `discard()` to remove an item from a set:

`my_set.discard(23)`

You cannot use bracket notation with sets as it is unordered and unindexed, has no keys nor indices

To access values in a set, you can loop through it with a **for** loop, or test if specific values are present using the **in** keyword



Review: Sets

Example: `my_set = {4, 23, 67, 1}`

What would be the result from the following code?

```
for x in my_set:  
    print(x)
```



Review: Sets

Example: `my_set = {4, 23, 67, 1}`

```
for x in my_set:  
    print(x)
```

The answer is **not**

4

23

67

1

```
>>> my_set = {4, 23, 67, 1}  
>>> for x in my_set:  
...     print(x)  
...  
1  
67  
4  
23  
>>> 
```

Though there's a chance it could be, it's not guaranteed.

The same 4 numbers will be printed, but the print order will not be the same as the order in which the set items were declared.

The screenshot above is one potential order it could be in



Workshop 3 Assignment

Goal: Code a text-based donations website!

Tasks 1-2

Set up files and folders, create homepage, initiate variables

Task 3

Handle user input, add exit functionality

Tasks 4-7

Add login, register, donations, and show_donations functionality.

You will be split up into groups to work on the assignment together.
Talk through each step out loud with each other, code collaboratively.
If your team spends more than 10 minutes trying to solve one problem,
ask your instructor for help!