Cyber Security Bootcamp

Hyperiondev

Lists & Dictionaries

Lecture - Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
- No question is daft or silly - **ask them!**
- There are Q/A sessions at the end of the session, should you wish to ask any follow-up questions.
- For all non-academic questions, please submit a query: www.hyperiondev.com/support
- Report a safeguarding incident: http://hyperiondev.com/safeguardreporting

Outcomes

1. Recall the fundamental characteristics of Lists.
2. Explain the concept of indexing in a list.
3. Apply knowledge of lists to manipulate elements.
4. Distinguish between the functionality of Lists and Dictionaries.
5. Expand on key operations relevant to Dictionaries.
6. Apply the above knowledge to improve data management in programs.

Polls

1. Which of the following statements is true about lists in Python?

A. Lists are immutable and cannot be changed after creation.

B. Lists can store elements of different data types.

C. Lists are accessed using key-value pairs.

D. Lists have a fixed size and cannot grow dynamically.

Polls

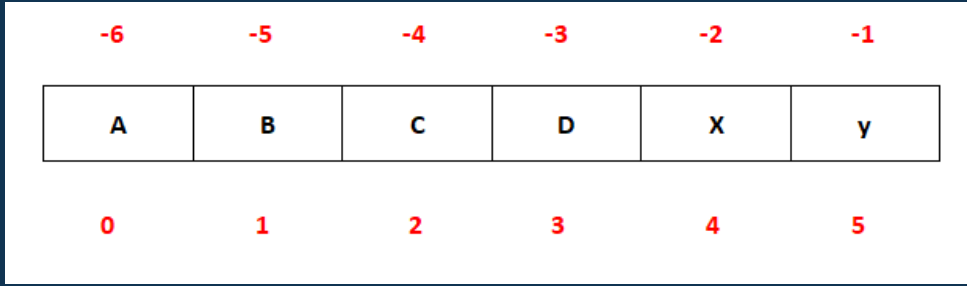
2. Which of the following is a key characteristic of dictionaries in Python?

A. Dictionaries are ordered collections of items.
B. Dictionaries can have duplicate keys.
C. Dictionary keys must be immutable types.
D. Dictionaries are accessed using index positions.

Lists

List Fundamentals

- A list is a data type that allows us to store multiple values of any type together and a list can contain duplicates.
- We can access individual values using indexing and multiple values using slicing.
- We can iterate over lists using a for loop.



List Fundamentals

- Lists are mutable. This means the values inside a list can be changed and unlike a string won't return a new list when changes have been made.
- We can apply methods to our lists without having to store them inside our variables.
- To create a list we can surround comma separated values with square brackets. []
- E.g. my_list = [value1, value2, value3]

List Functions

Key List Functions

Adding Elements	*append(), insert()*
Removing Elements	*remove(), pop() and 'del'*
Manipulating elements	sorting, reversing and slicing

List Examples

Creating Lists

```
# Creating a list of numbers
numbers = [1, 2, 3, 4, 5]

# Creating a list of strings
fruits = ["apple", "banana", "orange"]

# Creating a list of mixed data types
mixed_list = [1, "apple", True, 3.14]
```

List Examples

Adding and Removing Items

```
# Adding a single item
fruits.append("grape")

# Adding multiple items
fruits.extend(["pineapple", "mango"])

# Removing an item by value
fruits.remove("banana")

# Removing an item by index and returning it
removed_item = fruits.pop(2)
```

List Examples

Sorting Lists

```
# Sorting the list in-place
numbers.sort()

# Sorting the list in descending order
fruits.sort(reverse=True)

# Sorting a list without modifying the original list
sorted_numbers = sorted(numbers)
```

List Examples

Copying Lists

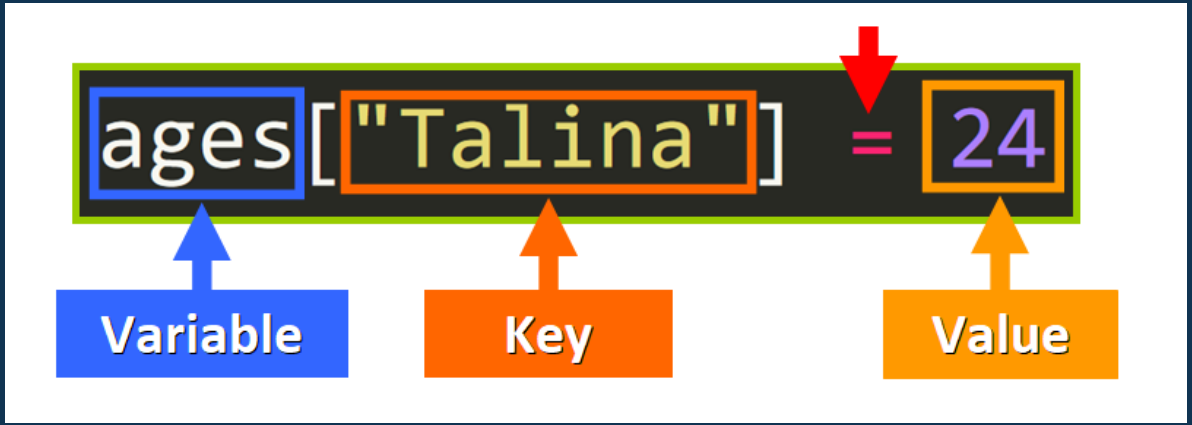
```
num_list = [1,2,3,4,5]
new_num_list = num_list

new_num_list[2] = 200
print(num_list)
```


```
num_list = [1,2,3,4,5]
new_num_list = num_list.copy()

new_num_list[2] = 200
print(num_list)
```

Dictionaries



Dictionary Fundamentals

- In Python, dictionaries function akin to the dictionaries we commonly used in English class, such as those from Oxford.
- Python dictionaries are similar to a list, however each item has two parts, a key and a value.
- To draw a parallel, consider an English dictionary where the key represents a word, and the associated value is its definition.

Dictionary Examples

- Dictionaries are enclosed in curly brackets; key value pairs are separated by colon and each pair is separated by a comma.

```
# Dictionary Example

my_dictionary = {
    "name": "Terry",
    "age": 24,
    "is_funny": False
}
```

- On the left is the key, and on the right is the value.

Dict Functions

- The dict() function in Python is a versatile way to create dictionaries and is basically a casting function.

- Create dictionaries through assigning values to keys by passing in keys and values separated by an = sign.

```
# Creating a dictionary with direct key-value pairs
my_dict = dict(name="Kitty", age=25, city="Belarus")
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus'}
```

Dictionary Access

- To access a value in a dictionary, we simply call the key and Python will return the value paired with said key.

- Similar to indexing, however we provide a key name instead of an index number.

```
my_dict = dict(name="Kitty", age=25, city="Belarus")
name = my_dict["name"]
# Output: 'Kitty'
```

Dictionary Update

- To append or add elements to a dictionary in Python:

  - You can use the update() method

```python
my_dict = dict(name="Kitty", age=25, city="Belarus")
# Adding or updating a key-value pair
my_dict.update({'breed': 'Shorthair'})
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus', 'breed': 'Shorthair'}
```

  - or simply use the square bracket notation.

```python
my_dict = dict(name="Kitty", age=25, city="Belarus")
# Adding or updating a key-value pair
my_dict['breed'] = 'Shorthair'
print(my_dict)
# Output: {'name': 'Kitty', 'age': 25, 'city': 'Belarus', 'breed': 'Shorthair'}
```

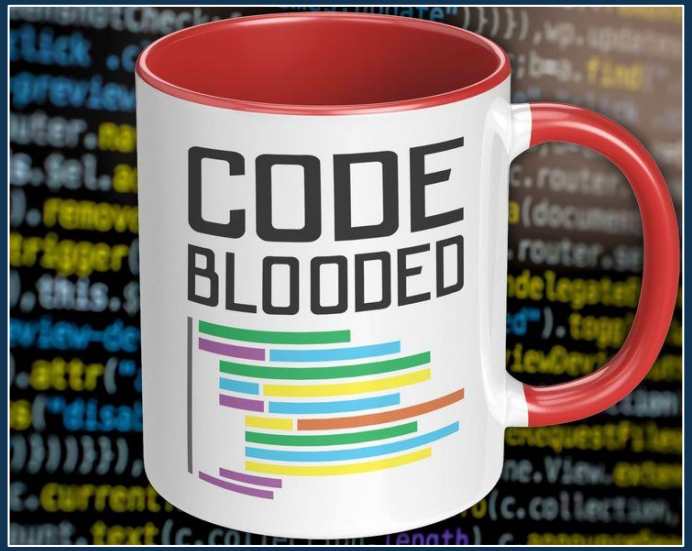
- Value will be updated if key exists, else the key and value will be added.

Dictionary Functions

Key Dictionary Functions

Key-Value Pairs	*items(), keys(), values()*
Fetching	*get()*
Updating	*update()*
Deleting	*pop(), popitem()*

Let’s get coding!



Hyperiondev

Q & A Section

Please use this time to ask any questions relating to the topic explained, should you have any.

Polls

1. Given the following list, what will be the output of the code below?

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

for fruit in fruits:
    print(fruit)
```

A. apple banana cherry

B. apple, banana, cherry

C. apple
   banana
   cherry

D. ["apple", "banana", "cherry"]

Polls

2. Given the following dictionary, what will the following code print?

```
scores = {"Alice": 80, "Bob": 90, "Charlie": 75}

for name, score in scores.items():
    if score > 80:
        print(name)
```

A. Alice, Bob, Charlie

B. Alice

   Bob

C. Bob

D. Charlie

Hyperiondev

Thank you for joining us

Take regular breaks.
Stay hydrated.
Avoid prolonged screen time.
Remember to have fun :)

Some useful links

Python Lists

- https://docs.python.org/3/tutorial/datastructures.html#more-on-lists

Python Dictionaries

- https://docs.python.org/3/tutorial/datastructures.html#dictionaries