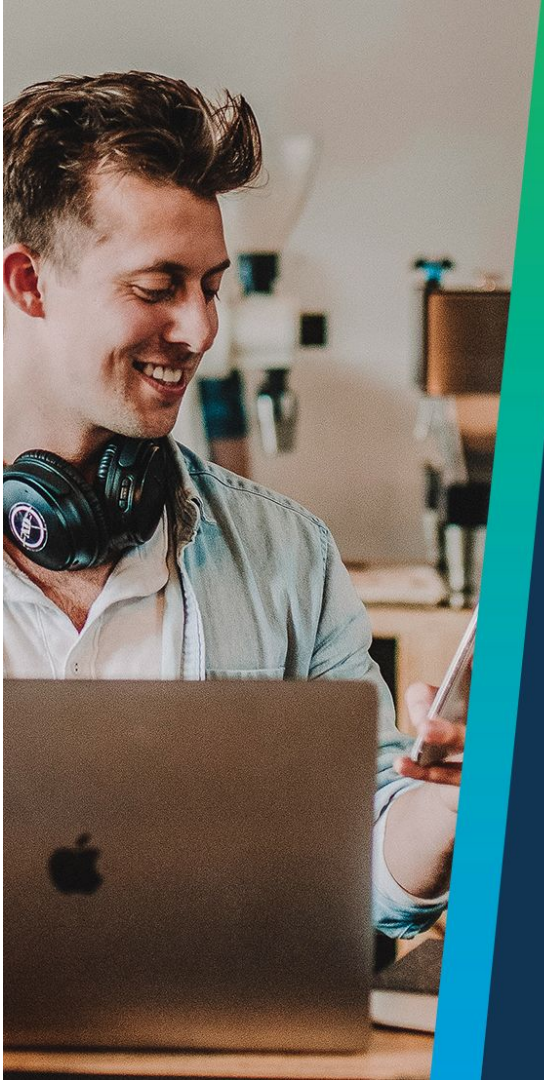


Software Engineering Bootcamp

Hyperiondev

Iteration & Sequences

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

Objectives

1. Recall the syntax and basic purpose of iteration in Python.
2. Describe how iteration works in Python and its applications.
3. Use loops to perform operations with sequences.
4. Recall the definitions and basic operations of strings, lists and dictionaries.
5. Explain the characteristics and uses of strings, lists and dictionaries.
6. Use strings, lists and dictionaries to solve simple problems.

Polls

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

A. A for loop is used to iterate over items of a sequence.
B. A while loop executes only once regardless of the condition.
C. Iteration is only possible with numeric data types.
D. A loop cannot be used to iterate over a string.

Polls

2. **Which of the following statements accurately describes the use of loops with lists in Python?**

A. Loops can be used to access and modify each element of a list individually.
B. Lists cannot be iterated over using loops in Python
C. A loop can only be used to iterate over numeric values in a list.
D. Loops are only used to access the first element of a list.

Iteration

- Iterations refer to the process of repeatedly executing a set of instructions or a block of code. In programming, iterations are commonly associated with loops, where the same code block is executed multiple times, either for a specified number of times or until a certain condition is met.

Iteration in Coding

- Each execution of the code block within a loop is called an iteration. Iterations are essential for automating repetitive tasks and processing collections of data efficiently.

- *“ Loops are like magic tricks in programming that help us avoid doing the same thing over and over again. Instead of writing the same code again, we use loops to make the computer do it for us. This saves time and makes our programs neat and tidy. “*

- In coding, there are two kinds of loops: for loops, which we use when we know exactly how many times we want to repeat something, and while loops, which we use when we want to keep doing something until a certain condition is true.

For Loops


DEFINITE
FOR
1 to 10
BREAK
CONTINUE
1... 2... 3... 4... 5... 6... 18
1
2
3
4
5
6
7
8
9
10
END
Real Python

For Loops

- For loops are control flow structures used to iterate over a sequence (such as a list, tuple, string, etc.) and execute a block of code for each element in the sequence.

- For loops are used when you know the number of times you want to execute a block of code.

```
for item in sequence:
    # code block to be executed
```

For Loops Example

```python
# Define a list of fruits
fruits = ["apple", "banana", "cherry", "date"]

# Use a for loop to iterate over the list
for fruit in fruits:
    print(fruit)
```

```
# Result
apple
banana
cherry
date

# This will print each fruit in the list
```

For Loops – Range Function

- Range is a built-in Python function used to generate a sequence of numbers. It is commonly used with for loops.

- Ranges in for loops are a way to specify a sequence of numbers that you want to iterate over. The range() function generates this sequence of numbers based on the arguments you provide.

```
range(start, stop, step)
```

For Loops – Range Function

- Range() takes three arguments: start, stop, and step.
- start: The starting value of the sequence (inclusive). If not provided, it defaults to 0.
- stop: The ending value of the sequence (exclusive). This is a required argument.
- step: The increment between each value in the sequence. If not provided, it defaults to 1.

Range Function Example

```
for i in range(start, stop, step):
    # code block to be executed
```

```
for i in range(1, 6):  # This will iterate from 1 to 5
    print(i)
```

- This loop will print numbers 1 through 5. Remember, the stop value is exclusive, so the loop stops before reaching 6.

While Loops


INDEFINITE
WHILE
n > 0
BREAK
CONTINUE
TRUE! TRUE! TRUE! TRUE!
FALSE
5
4
3
2
1
0
Real Python

While Loops

- While loops are control flow structures that repeatedly execute a block of code as long as a specified condition is true.

- These are used when you want to execute a block of code repeatedly as long as a specified condition is true. They continue iterating until the condition becomes false.

```
while condition:
    # code block to be executed
```

While Loops Example

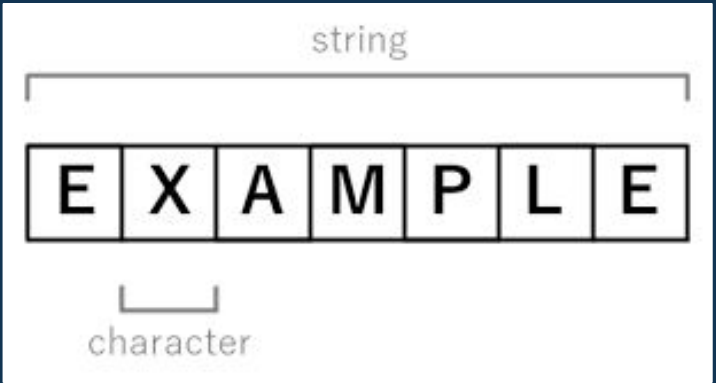
```
count = 0
while count < 5:
    print("Count is:", count)
    count += 1
# This will print numbers from 0 to 4.
```

- while count < 5: This is the beginning of a while loop. It checks if the value of count is less than 5. If this condition is true, the code block inside the loop will execute. If the condition is false, the loop will terminate.

- print("Count is:", count): This line prints the current value of count along with the text "Count is:". Since count is initially 0, it will print "Count is: 0".

- count += 1: This line increments the value of count by 1 in each iteration of the loop. So, after the first iteration, count becomes 1, then 2, and so on.

Strings

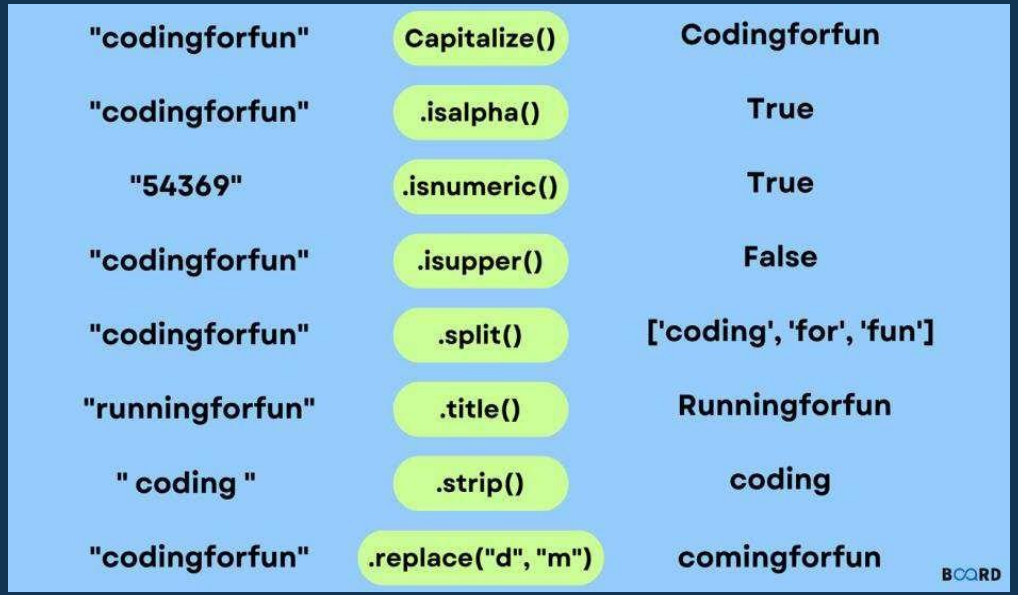


String Creation and Initialisation

- Strings in Python are sequences of characters, enclosed within either single quotes (' '), double quotes (" "), or triple quotes (''' ''')

```
message = "This is a string"
print(message)
```

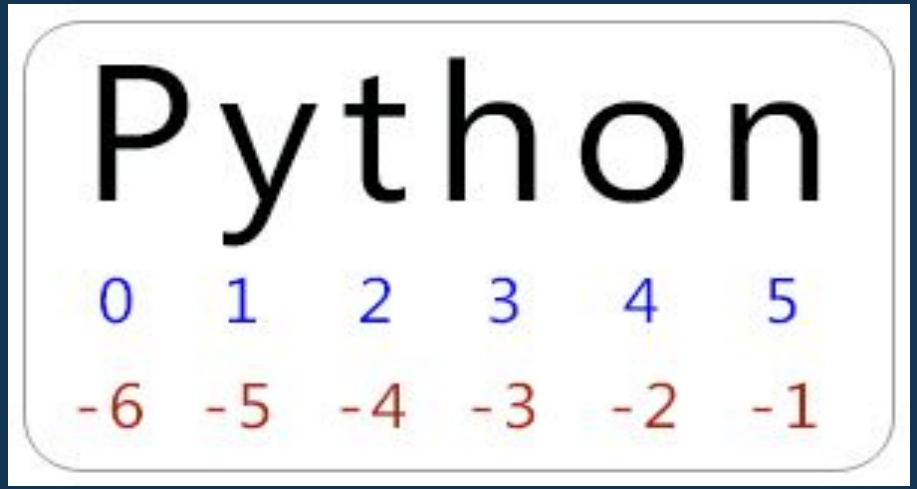
Basic String Methods



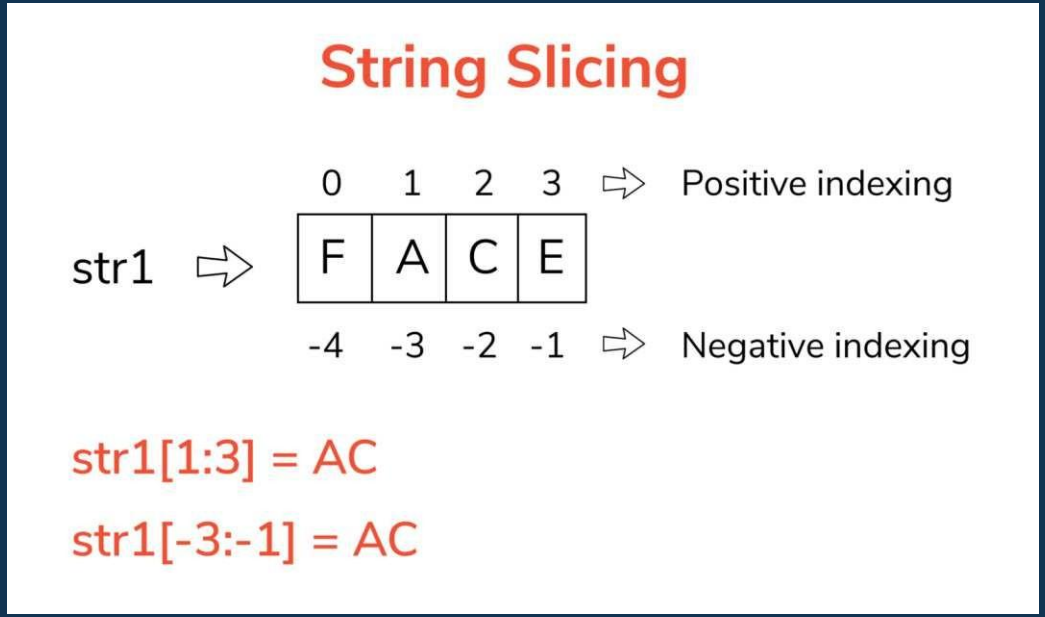
More about Strings

- Strings are Immutable.
- When an object is immutable it means the object cannot be changed.
- When we apply methods to a string that appear to make changes, they are actually creating and returning new string objects.
- This means we have to store the changes we make in a variable to be reused.

String Indexing



String Slicing

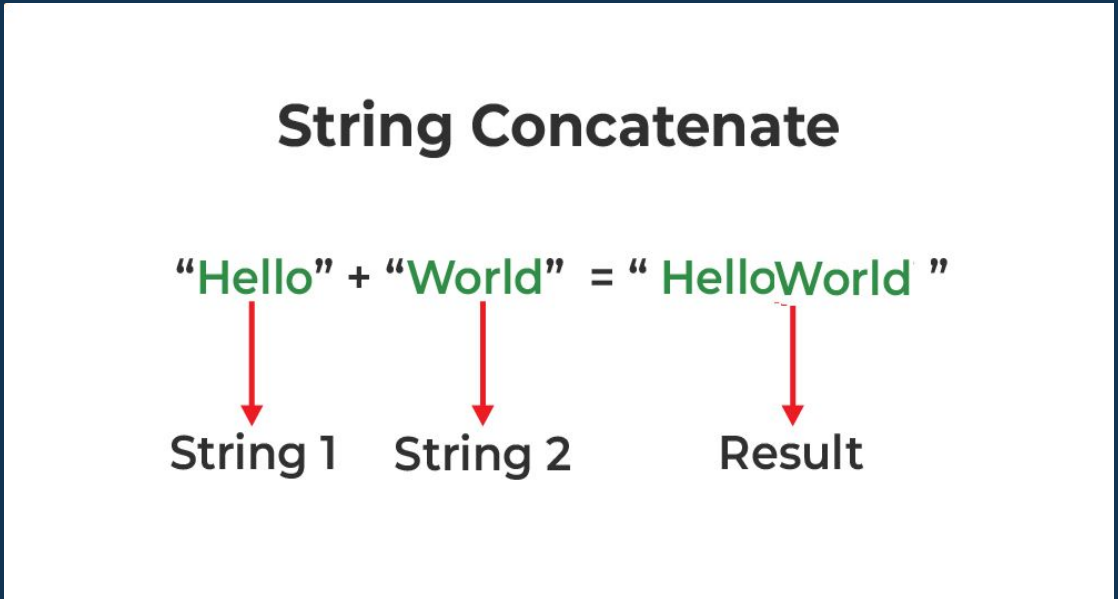


String Concatenation & Format

- String concatenation is the process of joining strings together, while formatting allows you to insert dynamic values into strings.

- String formatting in Python refers to the process of creating strings where dynamic values are inserted into predetermined locations within the string.

String Concatenation



String Formatting

- format function

```
name = "Inigo Montoya"
quantity = 51
formatted_string = "My name is {} and I would like {} muffins please.".format(name, age)
# Output: My name is Inigo Montoya and I would like 51 muffins please.
```

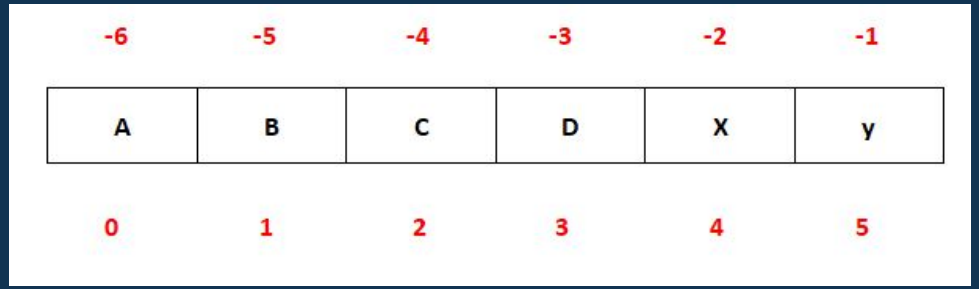
- f-string

```
random_word = "Spanish Inquisition"
formatted_string = f"Nobody expects the {random_word}!"
# Output: Nobody expects the Spanish Inquisition!
```

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

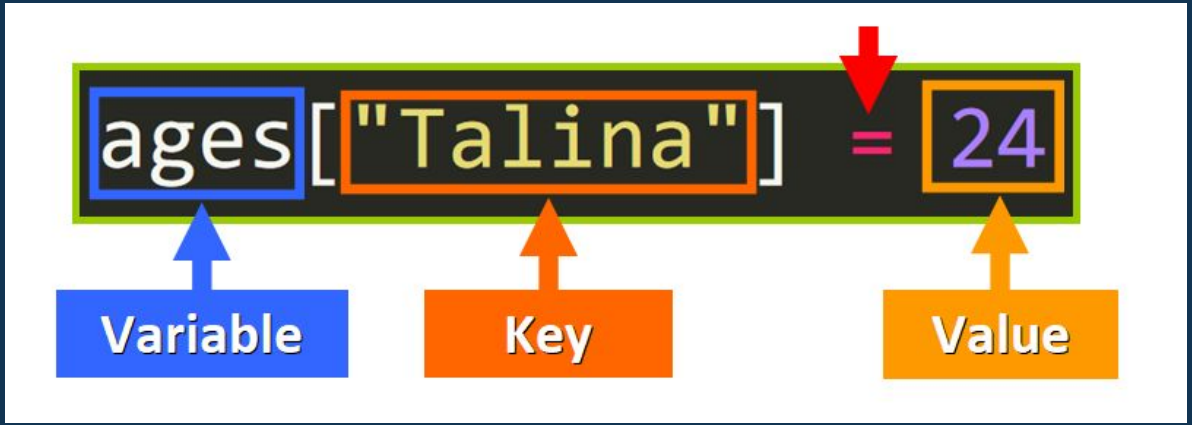
Sorting Lists

```
# 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)
```

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 Update

- To append or add elements to a dictionary in Python:
  - You can use the update() method

```
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.

```
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

Polls

3. **Given the following string, what will the following code print?**

```
word = "iteration"
count = 0

for letter in word:
    if letter in "aeiou":
        count += 1

print(count)
```

A. Counts the number of consonants in the word
B. Prints each vowel in the word
C. Counts the number of vowels in the word
D. Prints the total number of letters in the word

Hyperiondev

Thank you for joining us

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

Some useful links

Iterations
https://docs.python.org/3/library/stdtypes.html#iterator-types

Sequences
https://docs.python.org/3/library/stdtypes.html#sequence-types-str-unicode-list-tuple-bytearray

Python For Loops
https://www.w3schools.com/python/python_for_loops.asp
https://realpython.com/python-for-loop/

Python While Loops
https://www.geeksforgeeks.org/python-while-loop/