



Welcome to this session: Task Walkthrough - Tasks 6 - 9

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators
answering questions.



Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:



Ian Wyles
Designated Safeguarding
Lead



Simone Botes



Nurhaan Snyman



Rafiq Manan



Ronald Munodawafa



Tevin Pitts

Scan to report a
safeguarding concern



or email the Designated
Safeguarding Lead:
Ian Wyles

safeguarding@hyperiondev.com

Skills Bootcamp Data Science

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: [Questions](#)

Skills Bootcamp Data Science

- For all **non-academic questions**, please submit a query:
www.hyperiondev.com/support
- **Report a safeguarding incident:** **www.hyperiondev.com/safeguardreporting**
- We would love your feedback on lectures: **[Feedback on Lectures](#)**
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Learning Outcomes

- ❖ **Define and use user-defined functions** to perform modular and reusable operations.
- ❖ **Manipulate strings** for formatting, parsing, and extracting information.
- ❖ **Work with lists and dictionaries** to store, access, and manipulate data dynamically.
- ❖ **Perform file I/O operations** to read data from files and save results.
- ❖ **Model real-world entities using OOP principles** with Python classes.

Lecture Overview

- Presentation of the Task
-
- Task Walkthrough



Task Walkthrough

Welcome to your role as the ultimate Survey Data Scientist! 🎉 Imagine working for a nonprofit organization conducting a nationwide survey to understand the needs of communities better. Your job is to create a Survey Data Manager, a Python-powered application that organizes participant data, analyzes responses, and helps make data-driven decisions for impactful change.

- ❖ **Store participant information like their name, email, and responses.**
 - ❖ **Allow updates to responses so you can keep the data up-to-date.**
 - ❖ **Summarize survey results to uncover key insights.**
- ❖ **Save data to a file and reload it anytime, so you can seamlessly manage your survey over multiple sessions.**



What is the main purpose of a user-defined function in Python?

- A. To output data
- B. To group reusable blocks of code
- C. To store multiple values in a single variable
- D. To repeat code multiple times



Which Python data structure allows you to store key-value pairs?

- A. List
- B. Tuple
- C. Dictionary
- D. String

Functions



Functions

- ❖ To declare a function in Python, we use the **def** keyword.
- ❖ We have to provide a **name** for our function (using variable naming conventions), a list of **parameters** (placeholders for function inputs) in brackets, a colon and **body** of the function indented.
- ❖ We also need to add a **return statement** for functions that return a value. This is not necessary for all functions e.g. functions that modify a state.

```
# Syntax of a user-defined function
def functionName(parameter1, parameter2):
    # function block containing statements
    # which accomplishes a specific task
    result = "Output"
    return result
```

Functions

- ❖ After defining a function, we **call or invoke** it to use it in our code.
- ❖ We call a function with its name followed by a list of **arguments** enclosed in brackets, if required by the functions.
- ❖ **Arguments** are the input values provided to the function and take the place of the **parameters** defined in the function in the **same position**.

```
# Function which calculates the sum of two numbers
def calculateSum(a, b):
    return a + b

sum1 = calculateSum(800982390, 247332) # 801229722
sum2 = calculateSum(sum1, 3) # 801229725
```

Sequences



Lists

Ordered, mutable collections of data.

- ❖ Items in a list are known as **elements**.
- ❖ Elements do not have to be unique nor of the same type.
- ❖ Lists are **mutable**, meaning that elements in the list can be changed.
- ❖ Use **square brackets** to create lists and separate values with **commas**:

```
my_list = [1, "two", "buckle", True]
```

- ❖ We can access elements in a list using indexing, which is based on the element's position in the list:

```
print(my_list[0]) # 1  
print(my_list[3]) # True
```

Lists

❖ The most commonly used list functions are:

➤ Adding an element

```
my_list.append("three")  
my_list.insert(0, "zero")
```

➤ Deleting an element

```
my_list.remove("zero")  
my_list.pop(3)
```

➤ Manipulating the list: sorting, reversing etc.

```
my_list.sort()  
my_list.reverse()
```


Strings

- ❖ Strings are considered to be **immutable** collections of sequences in Python.
- ❖ We can access characters in our strings the same way we can access elements in a list:

```
string = "hello"  
print(string[0]) # h
```

- ❖ We can also manipulate strings using the same methods that we use on lists:

```
string.find("h")
```

Dictionaries

Collections of key-value pairs, where each key is unique.

- ❖ Unlike lists, dictionaries distinguish each element in the collection using a **key** instead of an **index**.
- ❖ When we use dictionaries to study languages, we look up definitions of a given word by looking up the word in the dictionary.
- ❖ In the data structure, we can access the **value associated with a key** value by looking up the key in the dictionary.
- ❖ Each element in a dictionary is a **key-value pair**.
- ❖ To create a dictionary, keys and values are **separated by colons (:)** and pairs are **separated by commas** and **enclosed in curly brackets {}**.

Dictionaries

- ❖ We can also use the **dict** function to create dictionaries:

```
my_dict = {"name": "Zahra", "age": 24}  
my_dict = dict(name = "Zahra", age = 24)
```

- ❖ To access values in a dictionary:

```
my_name = my_dict["name"]
```

- ❖ To add elements to a dictionary:

```
my_dict["bday"] = "13 November"
```

- ❖ To delete elements in a dictionary:

```
my_dict.pop("bday")
```

IO Operations



File Modes

- ❖ **Read** text from a file with the **mode 'r'**

```
file = open('file.txt', 'r')  
file.read()
```

- ❖ **Write** text to a file with the **mode 'w'**

```
file = open('file.txt', 'w')  
file.write("Hello World!")
```

- ❖ **Append** text to an existing file with the **mode 'a'**

```
file = open('file.txt', 'a')  
file.write("\nThis is a new line.")
```

File Handling (Reading)

Read from a File Python Methods

read()
Reads the entire
contents of the
file and returns it
as a string.

readline()
Reads a single line
from the file and
returns it as a
string.

readlines()
Reads all lines
from the file and
returns them as a
list of strings.

File Handling (Writing)

Write to a File Python Methods

write()

This method is used to write data to the file. It takes a string argument and adds it to the end of the file.

writelines()

This method writes a sequence of strings to the file. It takes a list of strings as an argument and writes each string to the file.

Resource Management

```
# Creating and destroying a file object
# Implicitly using with statement
with open('filename.txt', 'r') as file:
    content = file.read()

# Explicitly using open and close
file = open('filename.txt', 'r')
content = file.read()
file.close()
```

OOP - Classes



Class Properties

- ❖ **Attributes** are **variables** that belong to a class. They represent the **properties or characteristics** of the class that objects can have.
- ❖ **Methods** are **functions** that belong to a class. They define the **behaviors or actions** that an object of the class can perform.

```
class Student:  
    def __init__(self, name, age, mark):  
        self.name = name  
        self.age = age  
        self.mark = mark
```

Class Properties

- ❖ We define methods in our classes the same way that we would define functions.
- ❖ To reference any of the class' attributes we use **self**.

```
def sayMyName (self):  
    print("Hi, my name is " + self.name)
```

Methods

- ❖ In Python, **self** has to be passed into every **instance method** as the first parameter but does not have to be included when the function is actually called. You have to reference **self** when accessing attributes.
- ❖ **Instance methods** are actions or behaviors that specific objects can perform.
- ❖ They have access to the object's data and are defined within the class.

```
def study(self):  
    self.studying = True  
    print("{} is studying. Please do not disturb.".format(self.name))
```



Class Instantiation

- ❖ We use the **class** keyword to create a new class, followed by the name of the class.
- ❖ We use a **constructor function** to define anything that needs to take place when the object is first instantiated.
 - This includes any **attributes** that need to be defined, which we store using the **self** parameter, which is a reference to the object.
 - This function is called the **__init__ function** and it is called when a class is instantiated (created).



Class Instantiation

- ❖ To instantiate a class we call its constructor using the **name of the class**, followed by the **required attributes in brackets**.
- ❖ We usually store the instantiated class in a variable.
- ❖ We can access the instance methods and attributes by referencing the variable which stores the instantiated class followed by a ".".
- ❖ We can access class methods and static methods by referencing the class directly using the class' name.



Class Instantiation

```
class Student:
    def __init__(self, name, age, mark):
        self.name = name
        self.age = age
        self.mark = mark
        self.studying = False

    def study(self):
        self.studying = True
        print("{} is studying. Please do not disturb."
              .format(self.name))

student1 = Student("Zahra", 24, 89)
student1.study()
```

Task Walkthrough

Welcome to your role as the ultimate Survey Data Scientist! 🎉 Imagine working for a nonprofit organization conducting a nationwide survey to understand the needs of communities better. Your job is to create a Survey Data Manager, a Python-powered application that organizes participant data, analyzes responses, and helps make data-driven decisions for impactful change.

- ❖ **Store participant information like their name, email, and responses.**
 - ❖ **Allow updates to responses so you can keep the data up-to-date.**
 - ❖ **Summarize survey results to uncover key insights.**
- ❖ **Save data to a file and reload it anytime, so you can seamlessly manage your survey over multiple sessions.**

How do you read a file in Python?

- A. `file.read()`
- B. `file.open()`
- C. `file.input()`
- D. `file.write()`

Which of these is a correct way to define a class in Python?

- A. `class Person:`
- B. `define class Person()`
- C. `object Person:`
- D. `create class Person:`

Summary

- ★ **User-Defined Functions:**
Modularizing operations like data validation and analysis.
- ★ **Strings:**
Validating email formats and formatting output.
- ★ **Lists and Dictionaries:**
Storing participant objects and managing survey questions and responses.
- ★ **File I/O:**
Reading and saving data to/from files for persistence.
- ★ **OOP Classes:**
Modeling real-world entities and encapsulating behavior with methods.

CoGrammar

Q & A SECTION

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

Thank you for attending



CoGrammar



Department
for Education