**1. Data Structures & functions**

I will start this presentation by introducing the different data structures in python: Lists, Tuples, Sets and Dictionaries. Then I will demonstrate the use of comprehensions, how you can add conditions to your comprehensions, and the advantages of comprehensions.

After that I will talk about functions as first-class objects, how it enables inner functions, make a quick example of a function returning an inner function. If there’s enough time I can talk about how this relates to decorators.

**2. IO & Context Managers**

I will start this presentation off by presenting what a context manager is, why you should it use, and how to implement it. After that I will demonstrate how to read and write json objects to and from a file using a context manager.

**3. Pythonic OOP**

In this presentation i will focus on encapsulation and how it differs from other programming languages.  I will give some pratical examples of  variables and how to get and set them, show that you’re able to add variables to instances on the fly, show how pythonic “private” is only conventional, and how to implement some control of how to set specific variables by using the @property and @x.setter decorators.

**4.\_\_ Protocols \_\_**

In this presentation I will explain what protocols are, and how they enable your custom made objects to extend the intuitive python syntax with practical examples. Time permitting, I would also like to how the implementation of certain protocols enables/is necessary for creating a custom context manager or iterator.

**5. Generators**

In this presentation I will start off by using a simple generator function to demonstrate how the “yield” statement works, and how it differs from the “return statement. Then I will show how a genererator expression returns an iterable object instead of a list, and then I will discuss the merits of this contrasted to a list comprehension.

**6. Decorators**

In this presentation I will explain what a decorator is, how to create them in python, and how to decorate other functions with them. I will also give examples of things you have to keep in mind when decorating functions that takes arguments, returns values or if you want to keep the introspection of the decorated function to remain intact. If there’s time to spare I can also show how you can use decorators to implement a simple plug-in structure.