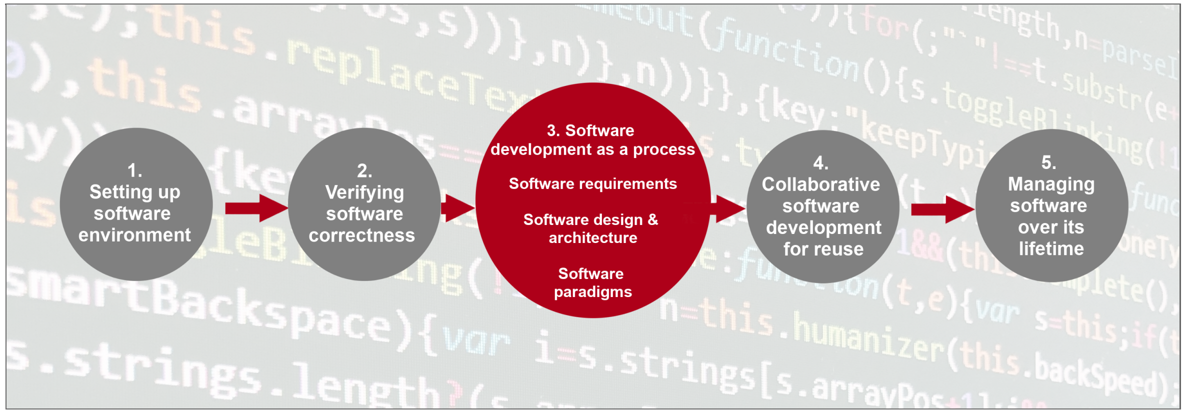
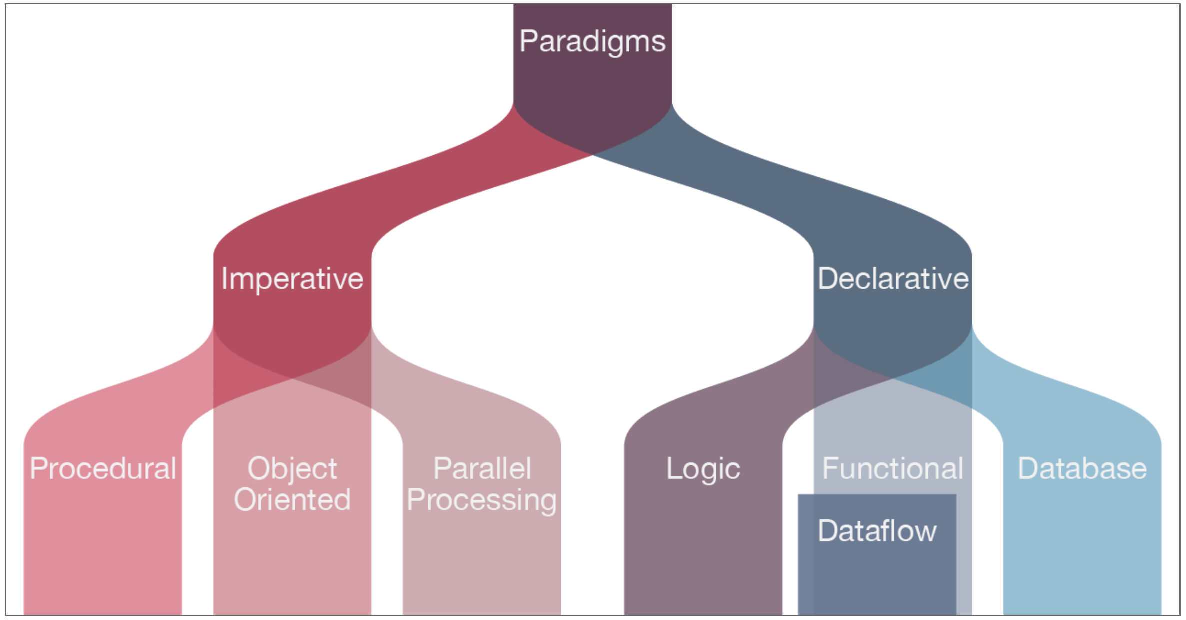
Section 3: Software Architecture and Design


1. Setting up software environment
2. Verifying software correctness
3. Software development as a process
Software requirements
Software design & architecture
Software paradigms
4. Collaborative software development for reuse
5. Managing software over its lifetime

Programming Paradigms



Modified from Davis, Daniel. 2013. "Modelled on Software Engineering: Flexible Parametric Models in the Practice of Architecture." PhD dissertation, RMIT University.

Breakout Exercise: ✏ 1, 2, Fizz, 4, Buzz … FizzBuzz

Start from this section heading and go to the end of the page.

FizzBuzz across Paradigms and Languages

R (DECLARATIVE, FUNCTIONAL)

```r
xx <- x <- 1:100
xx[x %% 3 == 0] <- "Fizz"
xx[x %% 5 == 0] <- "Buzz"
xx[x %% 15 == 0] <- "FizzBuzz"
xx
```

C (IMPERATIVE, PROCEDURAL)

```c
#include<stdio.h>

int main (void)
{
    int i;
    for (i = 1; i <= 100; i++)
    {
        if (!(i % 15))
            printf ("FizzBuzz");
        else if (!(i % 3))
            printf ("Fizz");
        else if (!(i % 5))
            printf ("Buzz");
        else
            printf ("%d", i);

        printf("\n");
    }
    return 0;
}
```

Object Oriented Programming

Decorators

Decorators are *syntactic sugar* for the case where we want to wrap a function with another function:

```python
def my_decorator(func):
    def wrapper():
        print("Before wrapped function")
        func()
        print("After wrapped function")
    return wrapper

# we can replace this
def my_function():
    print("Whee!")

my_function = my_decorator(my_function)
my_function()

# with this
@my_decorator
def my_function():
    print("Whee!")
```

Breakout: Start from the Top

Start reading from the top of this page all the way to the end. Complete exercises as you go.

- It is fine to skip the ✏ **Structuring Data** exercise and just read it. Bottom line is that you can achieve a lot with built in data types, so don't jump to more advanced techniques if they aren't needed.
- A note about the Book/Library exercises: create separate files from the existing ones
    1. put a `library.py` under `models/`, and a `test_library.py` under `tests/`
    2. or put both of these files under a separate directory `library/` at the top level of the repo

Class vs Static Methods

One common use case for class methods is to create alternate *constructors* for a class:

```python
class Circle:
    def __init__(self, radius):
        self.radius = radius

    @classmethod
    def from_diameter(cls, diameter):
        return cls(radius=diameter / 2)
```

Then, we can create `Circle` objects using either the radius or diameter:

```python
circle_1 = Circle(radius=1)
circle_2 = Circle.from_diameter(radius=2)
```

☕ 5 Minute Break ☕

Functional Programming

Breakout: Start from the Top ”What is a Function?”

Start from the top of this page and read through, completing any exercises along the way. Like with the Book/Library exercise previously, it is recommended to put your new code into separate files/modules.

THE REDUCE FUNCTION

A `for` loop under the hood: **https://github.com/python/cpython/blob/3.10/Lib/functools.py#L237**

```python
def reduce(function, sequence):
    # some argument checking
    # ...
    it = iterator(sequence)
    for element in it:
        value = function(value, element)
    return value
```

Alternate Sum of Squares

```python
from functools import reduce


def sum_of_squares(sequence):
    squares = [x * x for x in sequence]
    return reduce(lambda a, b: a + b, squares)


def to_integer(sequence):
    return [int(x) for x in sequence]


def remove_comments(sequence):
    return [x for x in sequence if x[0] != "#"]

sequence = ['#100', '1', '2', '4']
sum_of_squares(to_integer(remove_comments(sequence)))
```

☕ 10 Minute Break ☕

Software Design

> ***Software design*** *is the process by which an agent creates a specification of a software artifact intended to accomplish goals, using a set of primitive components and subject to constraints. Software design may refer to either "all the activity involved in conceptualizing, framing, implementing, commissioning, and ultimately modifying complex systems" or "the activity following requirements specification and before programming, as ... [in] a stylized software engineering process."*

- Part of the software development process
- Be careful: the *process* of designing software **VERSUS** *a design* of

some software (sometimes called a *design model*)

> ***Software architecture*** *refers to the fundamental structures of a software system and the discipline of creating such structures and systems. Each structure comprises software elements, relations among them, and properties of both elements and relations. The architecture of a software system is a metaphor, analogous to the architecture of a building. It functions as a blueprint for the system and the developing project, laying out the tasks necessary to be executed by the design teams.*

- This can be thought of as a particular sub-category of the design model that becomes more important the larger your software project gets

Breakout: Start from the Top

Start reading from the top of this page all the way to the end. Complete exercises as you go.
For the ✏ **Types of Software** exercise, please take 5 minutes at the beginning of the session to write down answers to the questions in the shared document, and share your answers with your group.

Some More Thoughts about Software Design

- Requirements collections is the important prerequisite since they determine the design constraints; the questions you ask to collect requirements are similar to the exercise ✏ **Types of Software**, but there can be many more
- Draw it out!
    - Use diagrams and schematics to try and visualise what a software programme/system will look like
    - It doesn't have to be fancy; just having something is far better than nothing
    - Seeing the connections between software components can usually help you further refine the design
- Design it twice

- The more designs you consider, the higher the chance you will move towards a better one

Persistence

```
In [ ]: import json

        patient = {"name": "Alice", "observations": [1, 6, 8, 0, 5]}
        with open("patient_serialized.json", "w") as fp:
            json.dump(patient, fp)
```

```
In [ ]: ! cat patient_serialized.json
```

```
In [ ]: with open("patient_serialized.json", "r") as fp:
            patient_deserialized = json.load(fp)
        print(patient_deserialized)
```

Breakout: Start from the Top

Start reading from the top of this page all the way to the end. Complete exercises as you go.

🕓 End of Section 3 🕓

Please fill out the end-of-section survey!