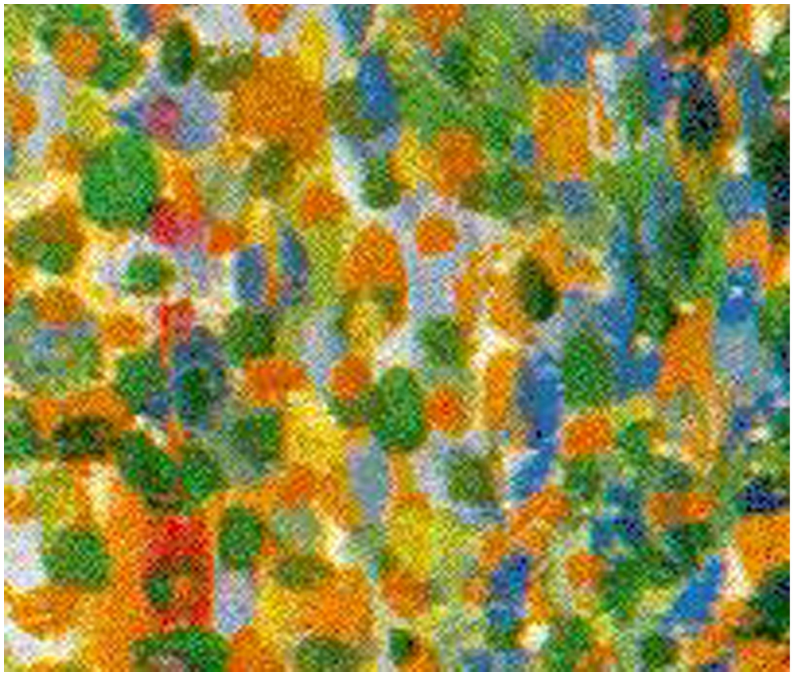
AUBURN
ENGINEERING

# ENGR 1110

## Module 5 Lecture

# Keeping perspective

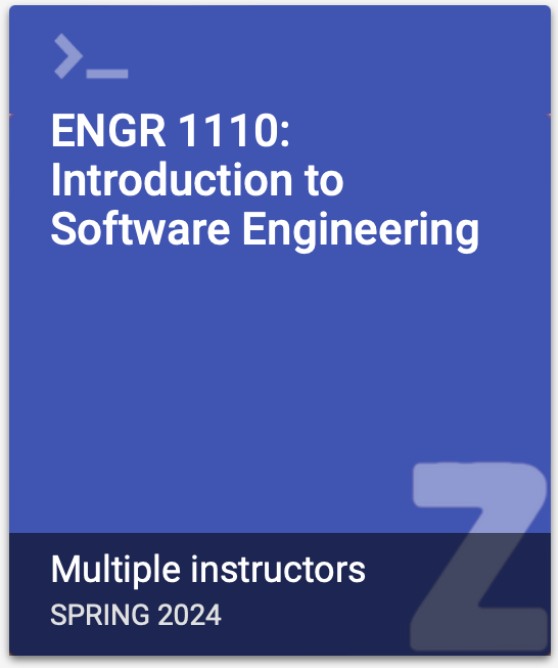


https://www.principlegallery.com/technique-tuesday-pointillism-take-two/

Georges Seurat, *A Sunday Afternoon on the Island of La Grande Jatte*, 1884-86, Public domain, via Wikimedia Commons

# zyBook

zyBooks
A Wiley Brand



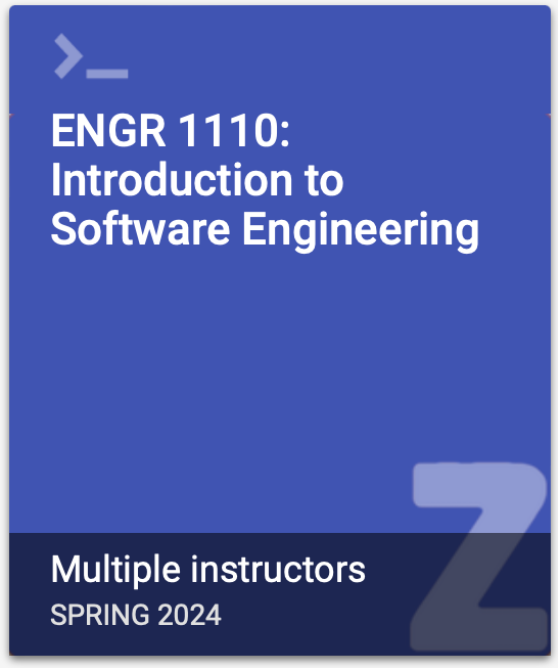

## Chapter 7. Functions


7) Functions

7.1 Function basics
7.2 Print functions
7.3 Dynamic typing
7.4 Reasons for functions
7.5 Writing mathematical functions
7.6 Function stubs
7.7 Functions with branches/loops
7.8 Functions are objects
7.9 Functions: Common errors
7.10 Scope
7.11 Scope resolution
7.12 Function arguments
7.13 Keyword arguments and default parameter values
7.14 Arbitrary arguments
7.15 Multiple function outputs
7.16 help(), docstrings
7.17 Engineering examples
7.18 Lab training: Unit tests to evaluate your program Lab
7.19 LAB: Driving costs - functions Lab
7.20 LAB: Step counter Lab
7.21 LAB: Convert to binary - functions Lab
7.22 LAB: Swapping variables Lab
**7.23 LAB: Fibonacci sequence** Lab

# zyBook

zyBooks
A Wiley Brand




## Chapter 7. Functions


7) Functions

7.1 Function basics
7.2 Print functions
7.3 Dynamic typing
7.4 Reasons for functions
7.5 Writing mathematical functions
7.6 Function stubs
7.7 Functions with branches/loops
7.8 Functions are objects
7.9 Functions: Common errors
7.10 Scope
7.11 Scope resolution
7.12 Function arguments
7.13 Keyword arguments and default parameter values
7.14 Arbitrary arguments
7.15 Multiple function outputs
7.16 help(), docstrings
7.17 Engineering examples
7.18 Lab training: Unit tests to evaluate your program Lab
7.19 LAB: Driving costs - functions Lab
7.20 LAB: Step counter Lab
7.21 LAB: Convert to binary - functions Lab
7.22 LAB: Swapping variables Lab
**7.23 LAB: Fibonacci sequence** Lab

## Incremental programming

```
# initialize and input list of values

# calculate maximum value

# normalize list with maximum

# output list
```

## Incremental programming

```
# initialize and input list of values
def input_list():
    pass

# calculate maximum value
def get_maximum(values):
    pass

# normalize list with maximum
def normalize_list(values, max_value):
    pass

# output list
def print_list(values):
    pass
```

## Incremental programming

```
# initialize and input list of values
def input_list():
    pass

# calculate maximum value
def get_maximum(values):
    pass

# normalize list with maximum
def normalize_list(values, max_value):
    pass

# output list
def print_list(values):
    pass
```

```
values = input_list()
max_value = get_maximum(values)
normalize_list(values, max_value)
print_list(values)
```

## Incremental programming

```
# initialize and input list of values
def input_list():
```

## Incremental programming

```
# initialize and input list of values
def input_list():
    values = []
    num_values = int(input())
    for i in range(num_values):
        values.append(float(input()))
    return values
```

## Incremental programming

```
# calculate maximum
def get_maximum(values):
```

## Incremental programming

```
# calculate maximum
def get_maximum(values):
     return max(values)
```

***OR, we could write the logic ourselves…***

```
# calculate maximum
def get_maximum(values):
    max_value = values[0]
    for value in values:
        if value > max_value:
            max_value = value
    return max_value
```

## Incremental programming

```
# normalize list with maximum
def normalize_list(values, max_value):
```

## Incremental programming

```
# normalize list with maximum
def normalize_list(values, max_value):
    for index, value in enumerate(values):
        values[index] = value / max_value
```

## Incremental programming

```
# output list
def print_list(values):
```

## Incremental programming

```
# output list
def print_list(values):
    for value in values:
        print(f'{value:.2f}')
```

## Incremental programming

```
# initialize and input list of values
def input_list():
    values = []
    num_values = int(input())
    for i in range(num_values):
        values.append(float(input()))
    return values


# calculate maximum
def get_maximum(values):
    max_value = values[0]
    for value in values:
        if value > max_value:
            max_value = value
    return max_value
```

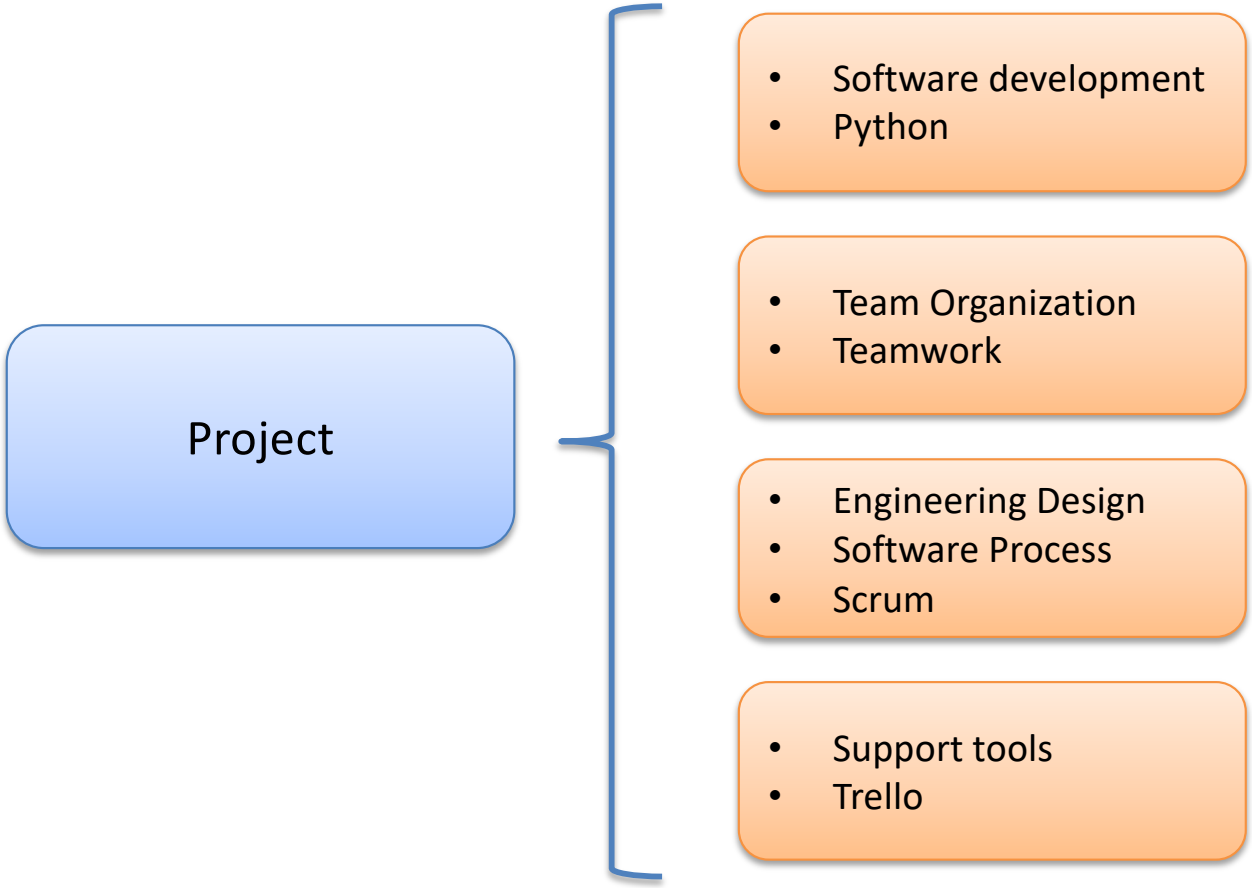
```
# normalize list with maximum
def normalize_list(values, max_value):
    for index, value in enumerate(values):
        values[index] = value / max_value


# output list
def print_list(values):
    for value in values:
        print(f'{value:.2f}')


values = input_list()
max_value = get_maximum(values)
normalize_list(values, max_value)
print_list(values)
```

## Learning to develop programs

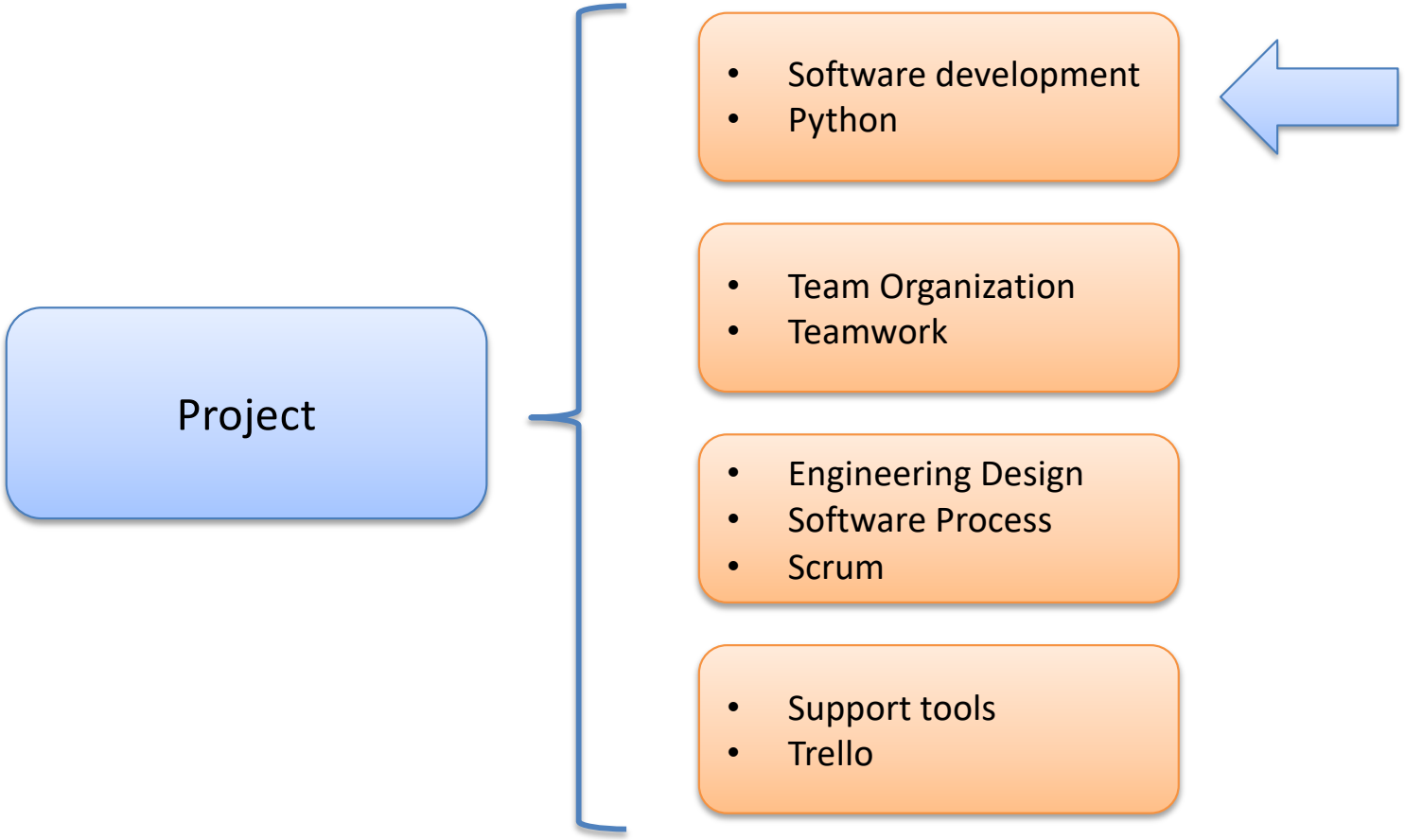
Don’t get too focused on **WHAT** you develop.

Remember that **HOW** you develop is critical.

## Team Project



## Team Project



# Python Resources

**Python Software Foundation**

`https://python.org/`

**Official Python Documentation**

`https://docs.python.org/`

**Python Style Guide (PEP 8)**

`https://peps.python.org/pep-0008/`

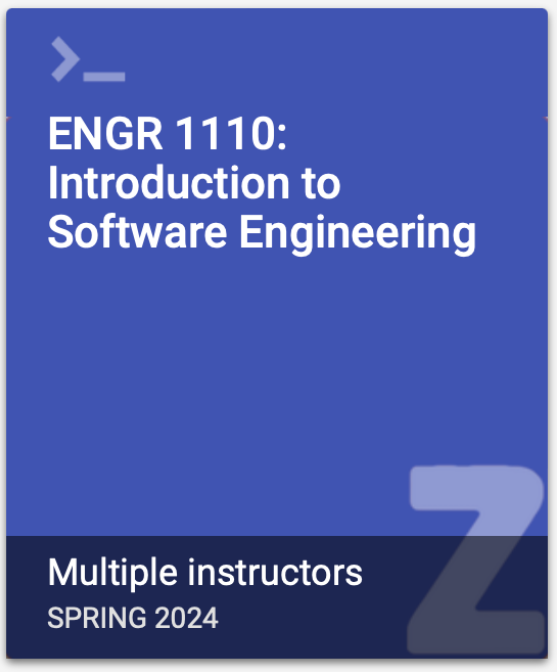
**Python Anywhere**

`https://www.pythonanywhere.com/`

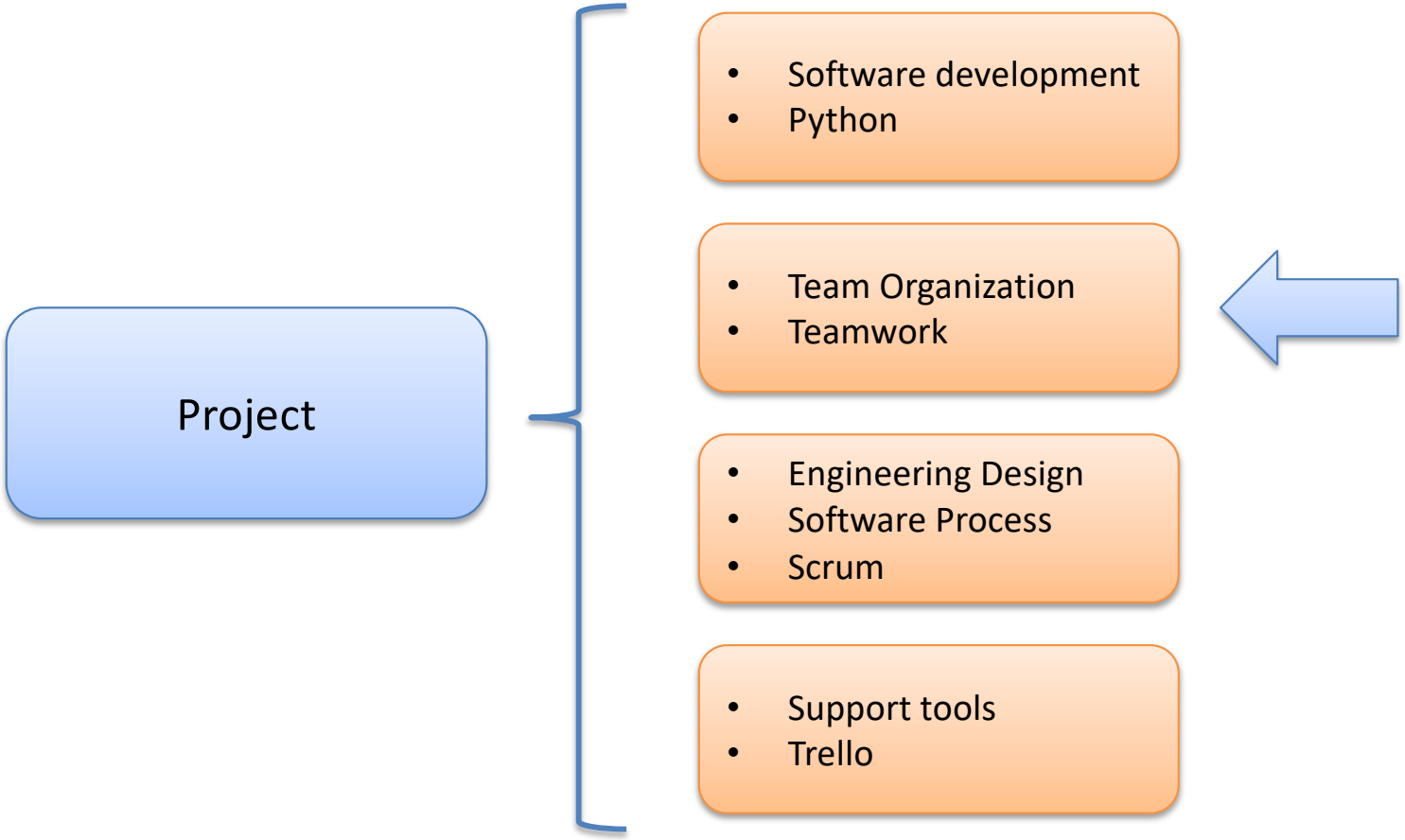
**Python Tutor**

`https://pythontutor.com/`

zyBooks
A Wiley Brand



## Team Project

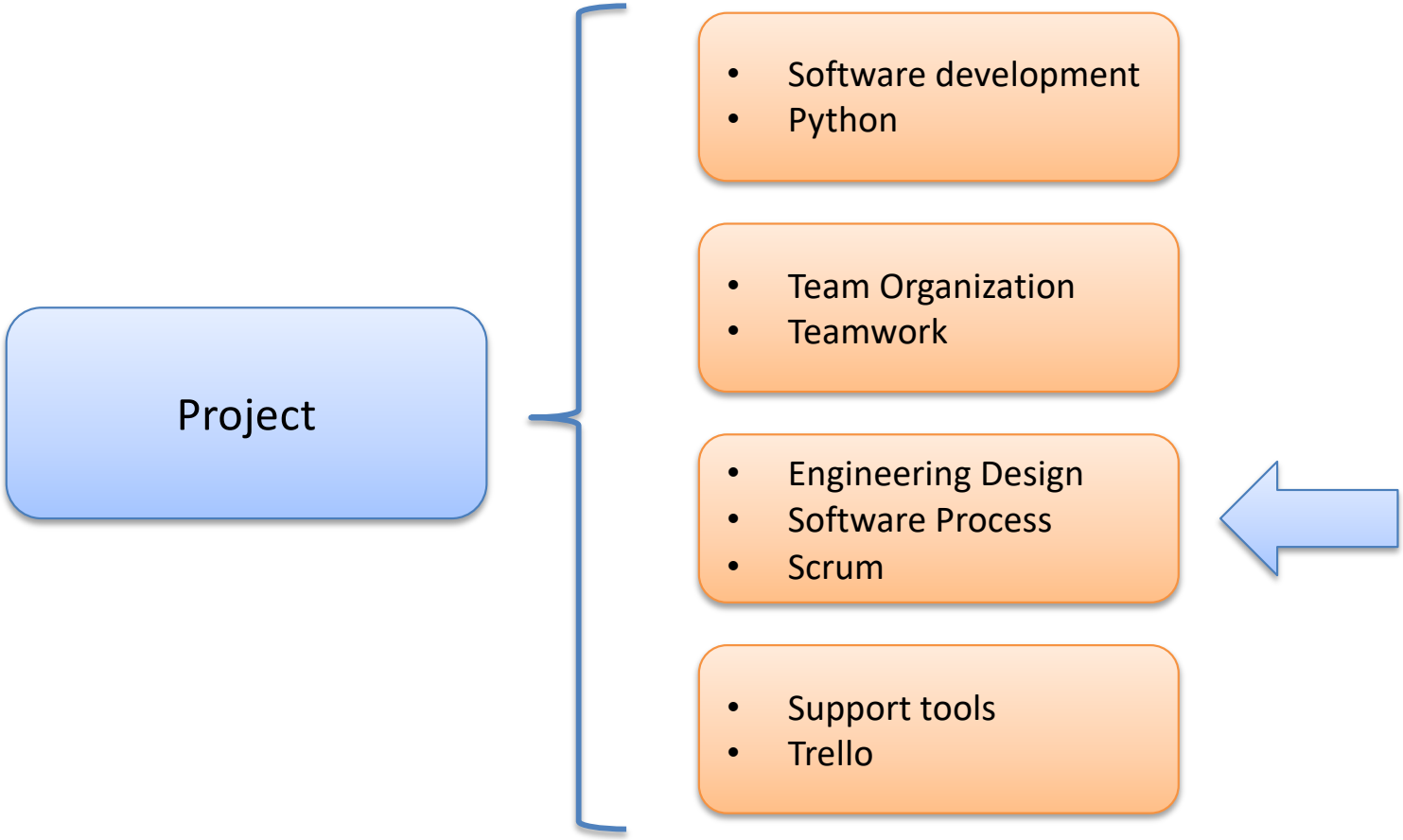


# Teams

All students have been automatically and randomly assigned to a team of 4-5 people.

Find your team and your teammates under “People” in the left menu of Canvas.

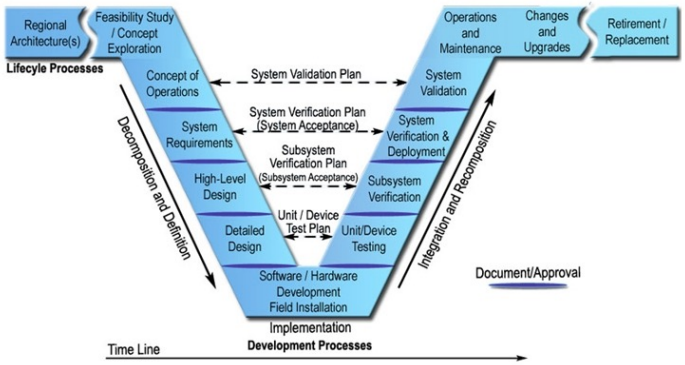
## Team Project



# Engineering Design

**Engineering design** is a multi-step process that guides an engineer in developing a new product, process, or solution that meets desired needs and requirements.

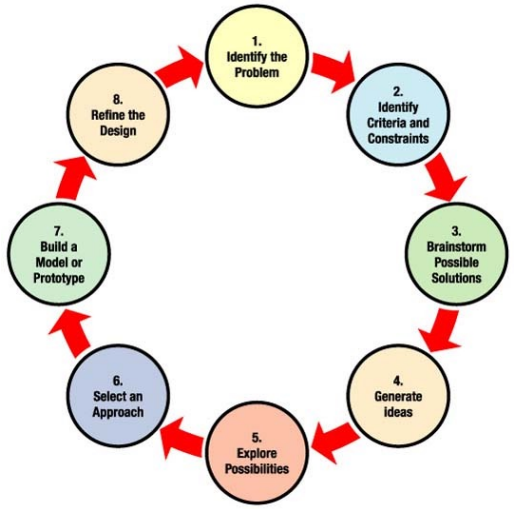
**Step 1.** Identify the problem
**Step 2.** Identify criteria and constraints
**Step 3.** Assess feasibility
**Step 4.** Explore possible solutions
**Step 5.** Select an approach
**Step 6.** Build a model or prototype
**Step 7.** Refine the design
**Step 8.** Develop a product




*U.S. Department of Transportation*

## Engineering Design

An engineering design process is typically **iterative** or at least includes **feedback loops**.




www.nasa.gov

In many traditional engineering disciplines, this process leads to a robust **prototype** that can then be **manufactured** as a **product**.

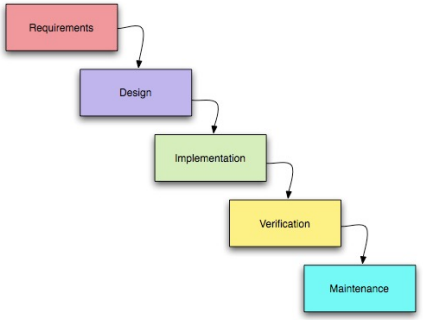
This isn't the case in software engineering since there is no "manufacturing". Although there are different roles and responsibilities, a software engineer or software engineering team is responsible for concept, design, development, and production.

# Software Process

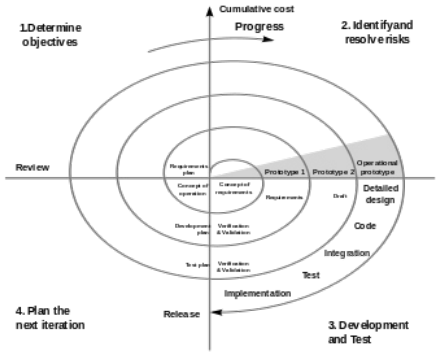
A **software process** is a systematic approach to engineering a software system or product.

There are many different software processes with different characteristics, but all address the basic software development activities: planning, specification, design, implementation, testing, validation, deployment, maintenance.

## Waterfall




## Spiral




## Cleanroom

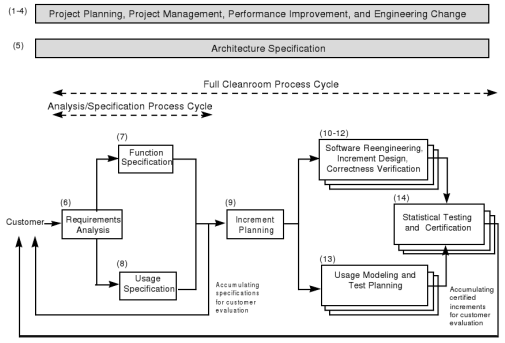



Figure 1. Cleanroom Process Flow

## Software Process

Some software processes (like those shown before) are called **heavyweight** processes, while others are called **lightweight** or **agile** processes.

Agile processes value: **Individuals and interactions** over processes and tools
**Working software** over comprehensive documentation
**Customer collaboration** over contract negotiation
**Responding to change** over following a plan

These values are expressed in the **Agile Manifesto** from the Agile Alliance.

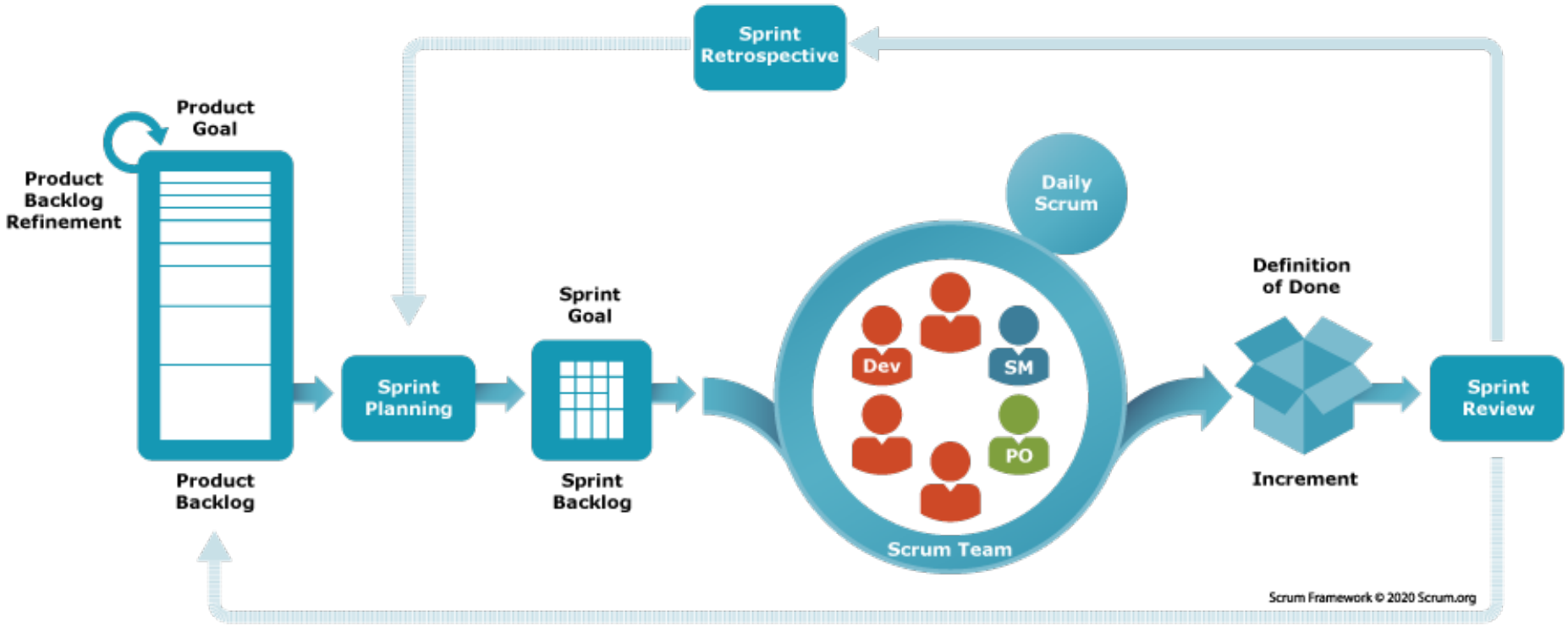
## Scrum

**Scrum** is a process used in agile software development, but is perhaps best described as a **framework** rather than a process, per se.

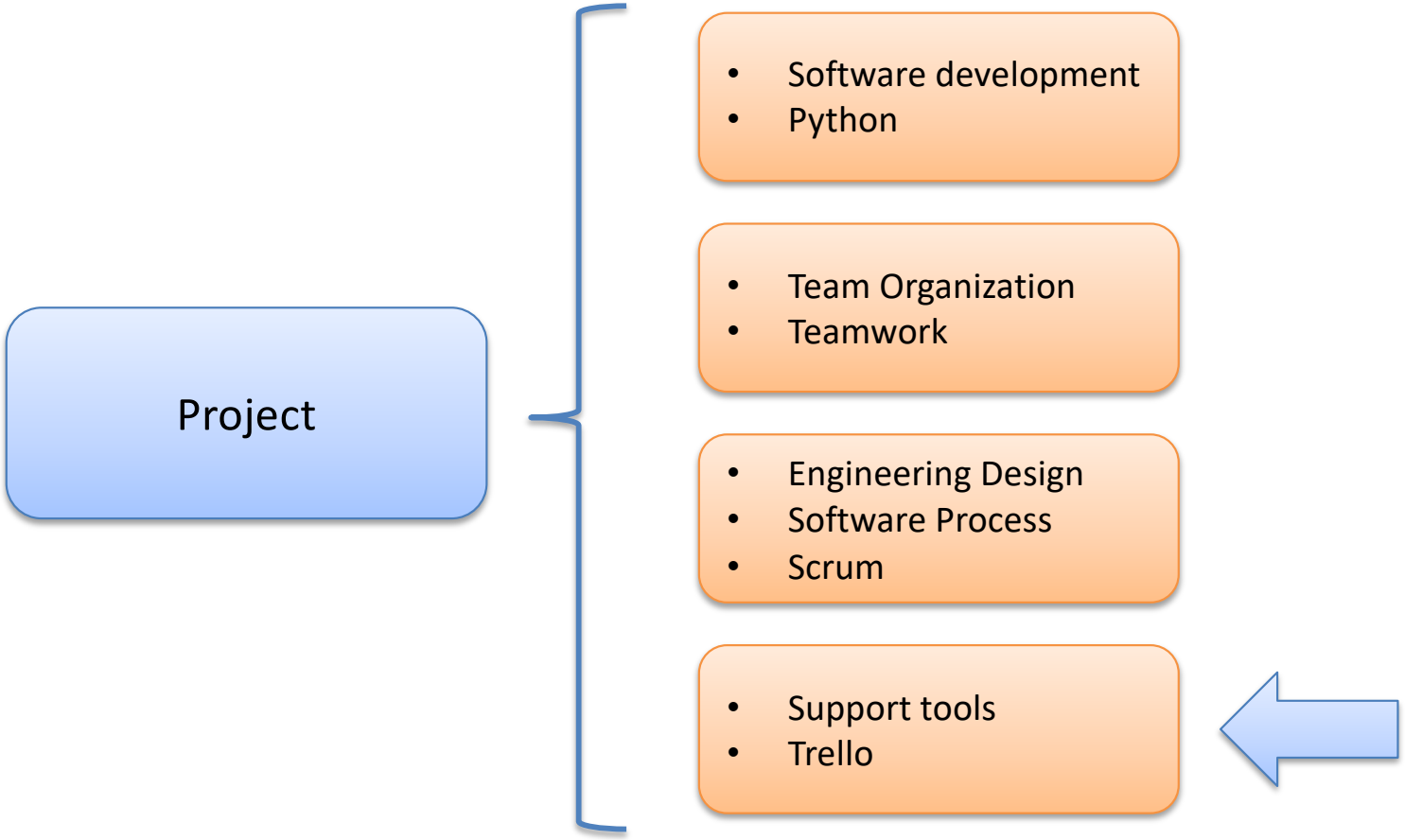
The idea that became the software process was first described as a new approach to product development by Hirotaka Takeuchi and Ikujiro Nonaka in the Harvard Business Review (86116:137-146, 1986).

> "The … 'relay race' approach to product development … may conflict with the goals of maximum speed and flexibility. Instead, a holistic or 'rugby' approach – where a team tries to go the distance as a unit, passing the ball back and forth – may better serve today's competitive requirements."

# Scrum


Sprint Retrospective
Product Goal
Product Backlog Refinement
Product Backlog
Sprint Planning
Sprint Goal
Sprint Backlog
Daily Scrum
Dev
SM
PO
Scrum Team
Definition of Done
Increment
Sprint Review
Scrum Framework © 2020 Scrum.org

## Team Project



## Trello

Trello is a web-based tool that supports work management via **Kanban boards**.

> “In Japanese, kanban literally translates to "visual signal." For kanban teams, every work item is represented as a separate card on the board.
>
> The main purpose of representing work as a card on the kanban board is to allow team members to track the progress of work through its workflow in a highly visual manner.”

https://www.atlassian.com/agile/kanban

Trello


https://www.forbes.com/advisor/business/software/what-is-kanban-board/

## Trello in ENGR 1110


- Focus on **how** you go about development.
- Use Trello to guide, plan, and **document** your work.
- This is important!

**Beginning with the next module, you will be required to use Trello to:**

- **Document your individual work in zyBooks.**
- **Document your individual contribution to a team assignment in zyBooks.**

**This will prepare you and your team to document all project work.**