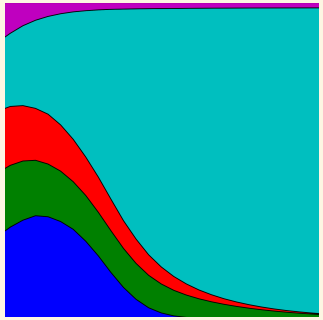
Four stories: four models of learning.

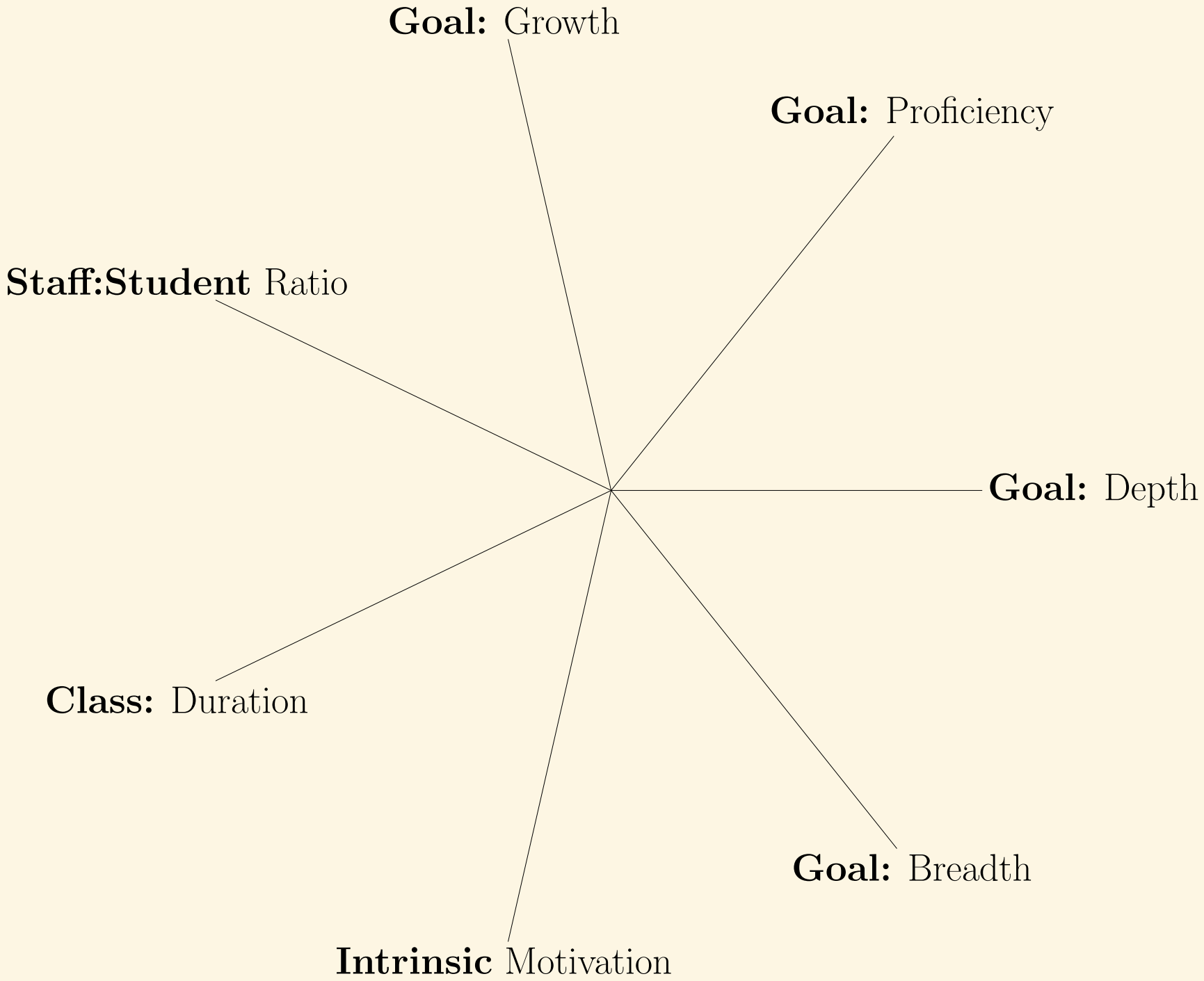
@drvinceknight
vknight.org
knightva@cardiff.ac.uk

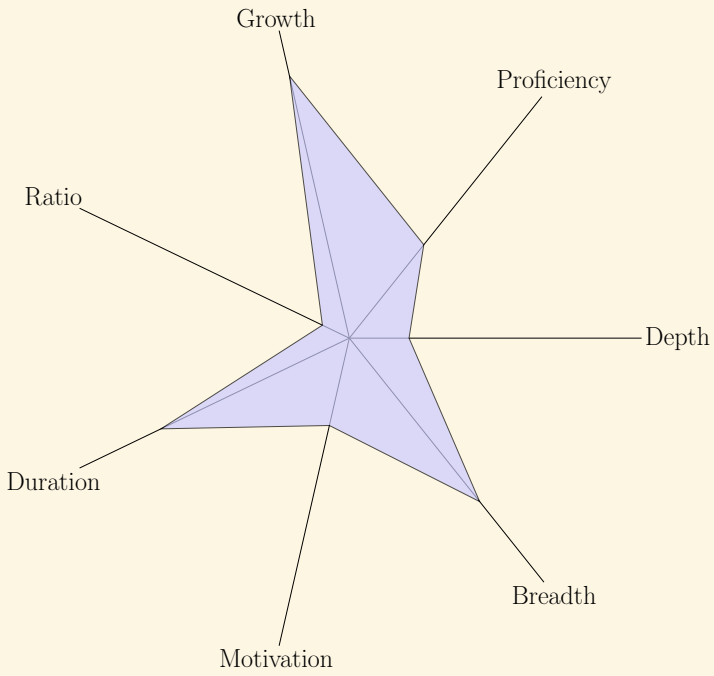


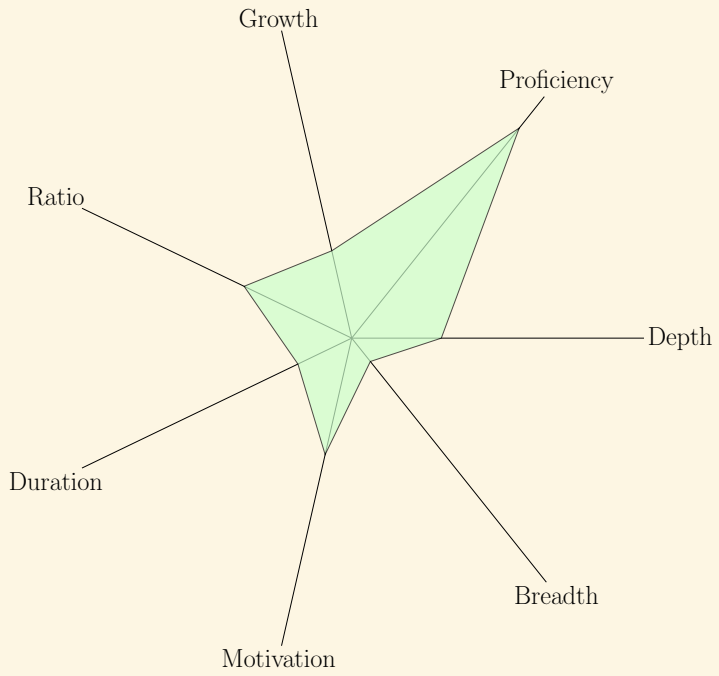
Software
Sustainability
Institute

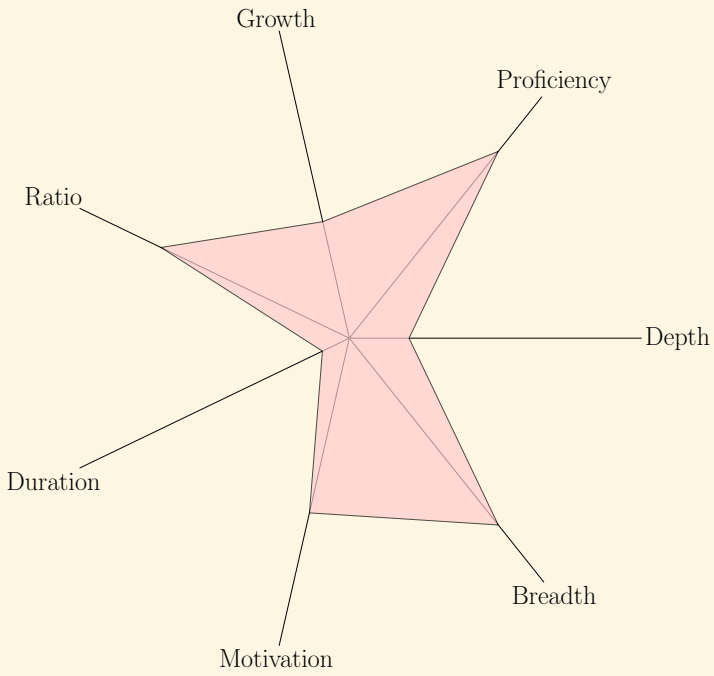
Active learning increases student performance in science, engineering, and mathematics Freeman et al. 2014 (PNAS)

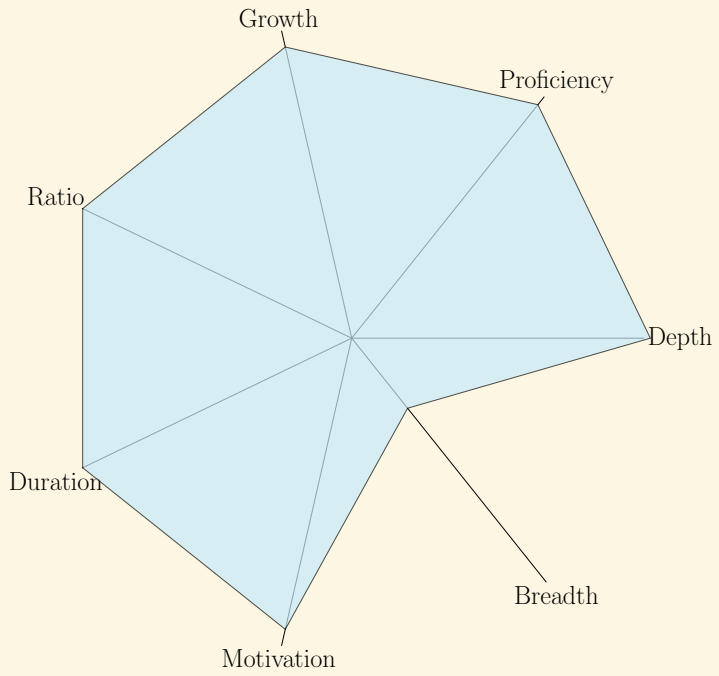
https://vknight.org/tch-phi/


Goal: Growth
Goal: Proficiency
Staff:Student Ratio
Goal: Depth
Class: Duration
Goal: Breadth
Intrinsic Motivation

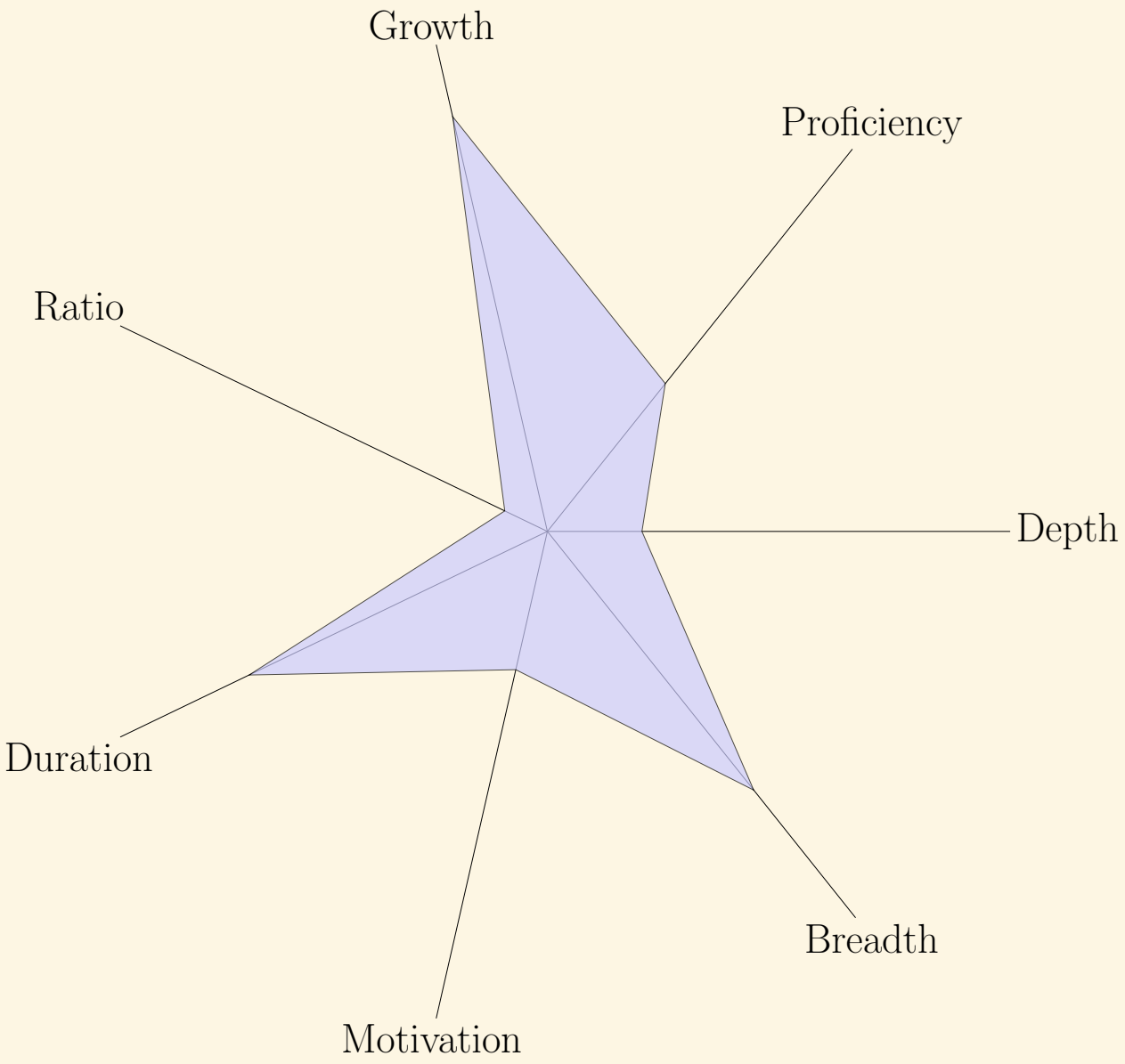

Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation

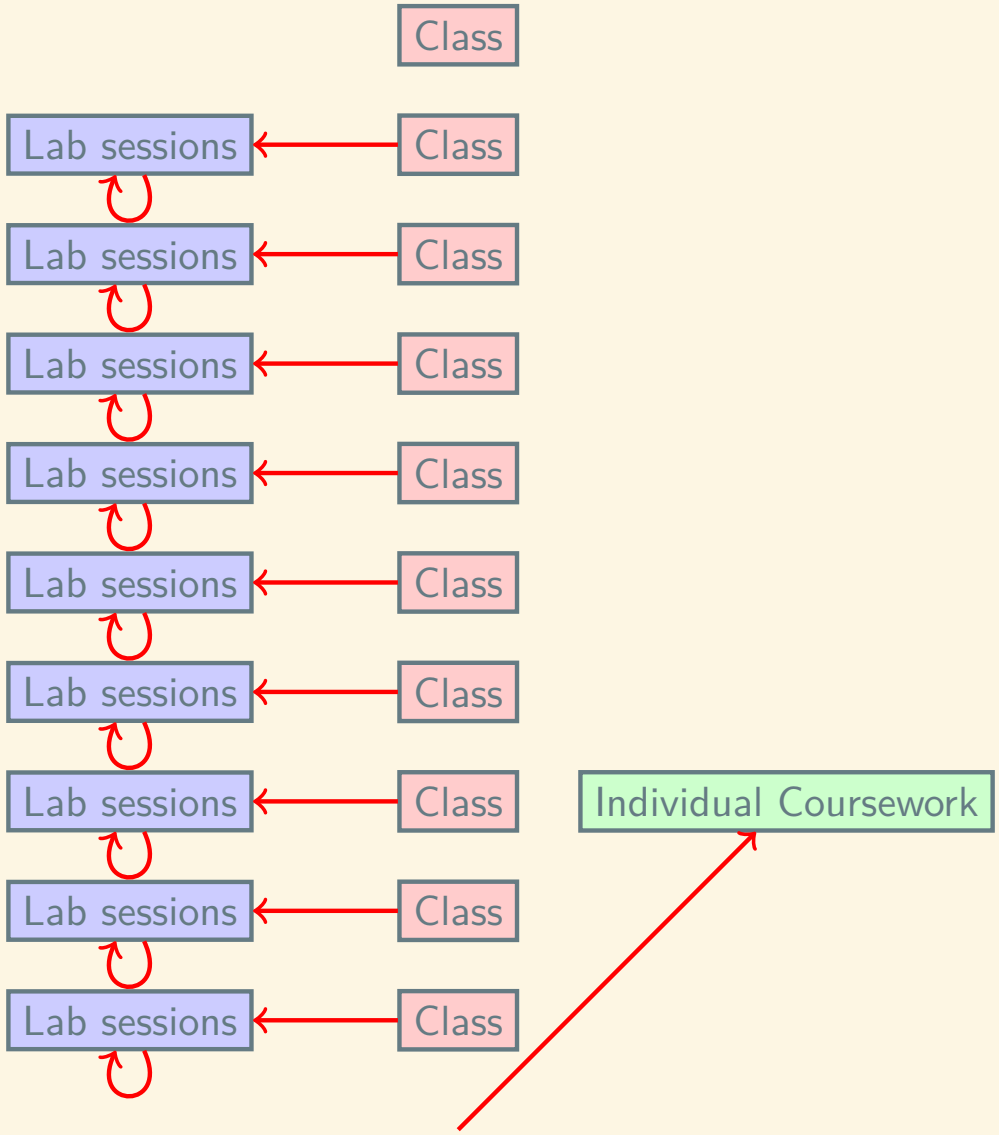

Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation


Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation


Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation

First year undergraduate class




Class
Lab sessions
Class
Lab sessions
Class
Lab sessions
Class
Lab sessions
Class
Lab sessions
Class
Lab sessions
Class
Lab sessions
Class
Lab sessions
Class
Individual Coursework

1. Recursion.

   A video describing the concept.

   A video demo.

   It is possible to define functions recursively. This is similar to inductive proofs in mathematics. To do this we need to consider two things:

   1. A recursive rule;
   2. A base case (that the recursive rule reduces to).

   As an example let us code the following function:

$$f(n) = \sum_{i=0}^{n} i$$

   We can see the **'recursive rule'**:

$$f(n) = f(n-1) + n$$

   We also know that the **'base case'** is:

$$f(0) = 0$$

   Now let us program this function using recursion:

```
>>> def sum_of_integers(n):
...     """Sum of integers from 0 to n"""
...     if n == 0:  # Base case
...         return 0
...     return sum_of_integers(n - 1) + n  # Recursive rule
```

1 Introduction

"Given a 5 x 5 set of lattice points (Figure 1), find a set of five circles which pass through each of the 25 points at least once" is a problem given in [1] (page 208). The aim of this paper is to find such a set of five circles. Additionally, this paper aims to ascertain whether this is the minimal number of circles required to intersect every point of the lattice. This will be done by considering all the possible circles that may be drawn and considering all combinations of circles, starting with one circle and increasing circles until a minimum solution is found.

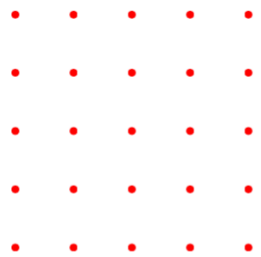


Figure 1: Lattice of 5 x 5 points.

2 Finding every circle and the minimal solution

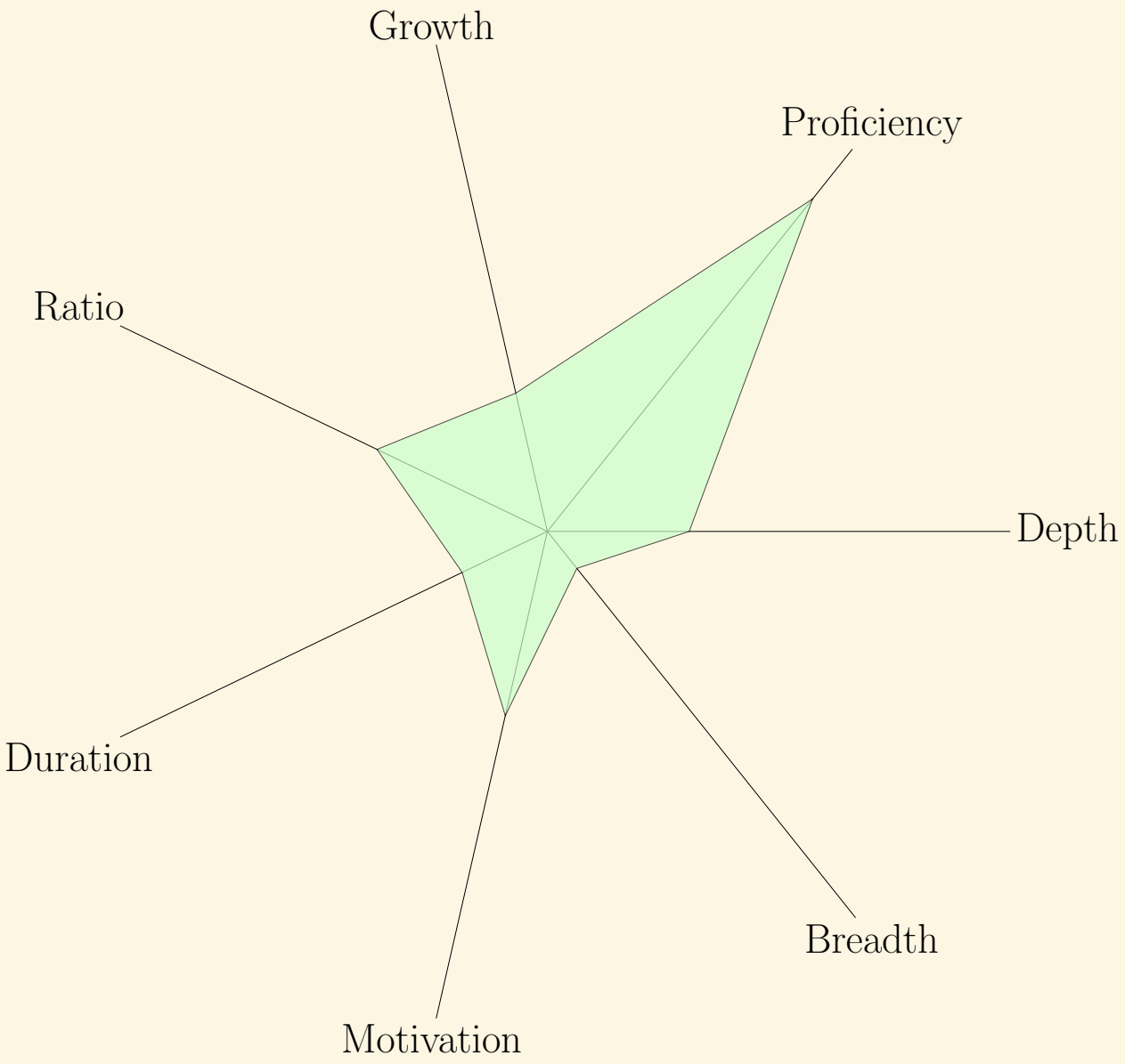
To represent the lattice a set of coordinates is created with each point of the lattice being an element in a list:

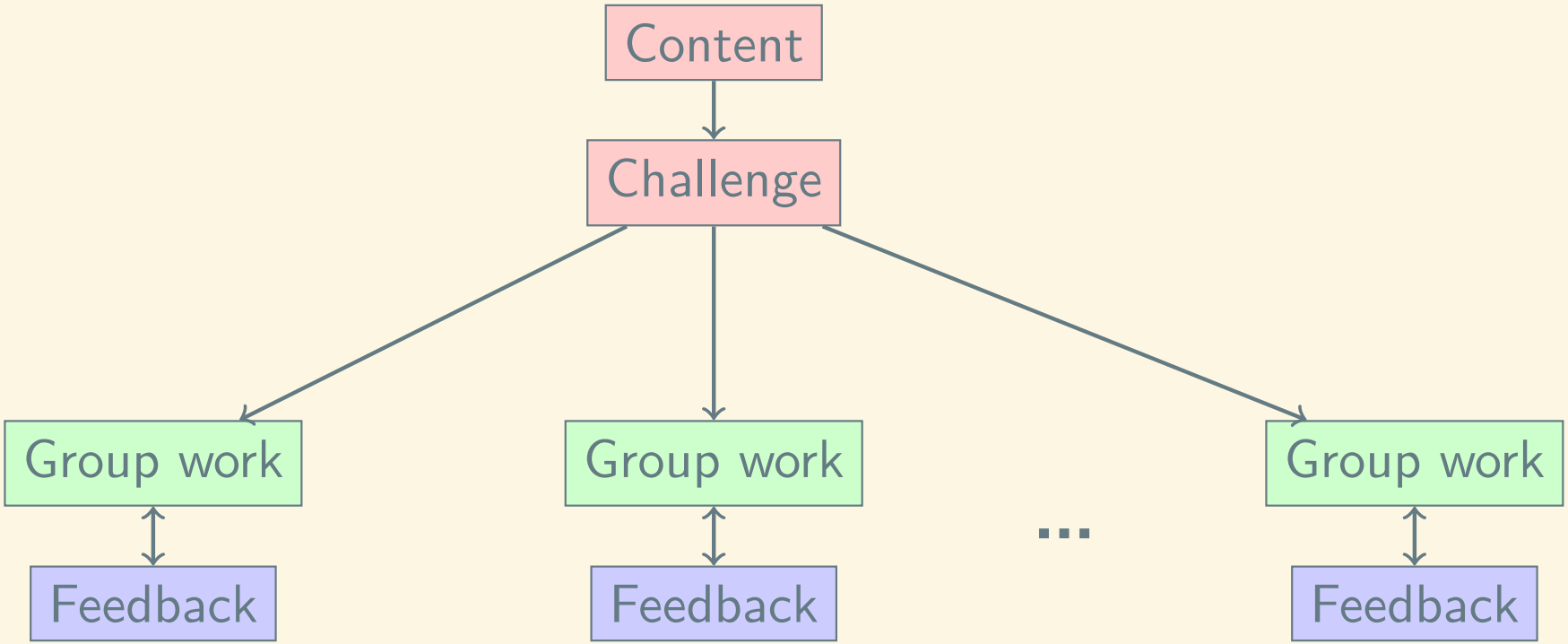
```
>>> n = 5
>>> coords = [[x, y] for x in range(1, n + 1) for y in range(1, n + 1)]
```

The function `set_of_circles` is created to determine the set of non-trivial circles that may be drawn on lattice:

```
>>> from itertools import combinations    # to iterate all possible combinations
>>> from sympy import Circle, Point    # to interpret possible circles
>>>
>>> def set_of_circles(coords):
...     """Returns a unique set of circles that intersects coordinate set coords."""
...     circles = []
...     for i in combinations(coords, 3):
...         if Point.is_collinear(i[0], i[1], i[2]) == False:
...             c = Circle(i[0], i[1], i[2])
...             if c not in circles:
...                 circles.append(c)
...     return circles
```

Masters level 2 day hackathon




Content
Challenge
Group work
Group work
...
Group work
Feedback
Feedback
Feedback

You need to build a tournament that creates a round robin of players using different strategies playing Rock Paper Scissors. Each match between 2 players will be 21 rounds of Rock Paper Scissors.

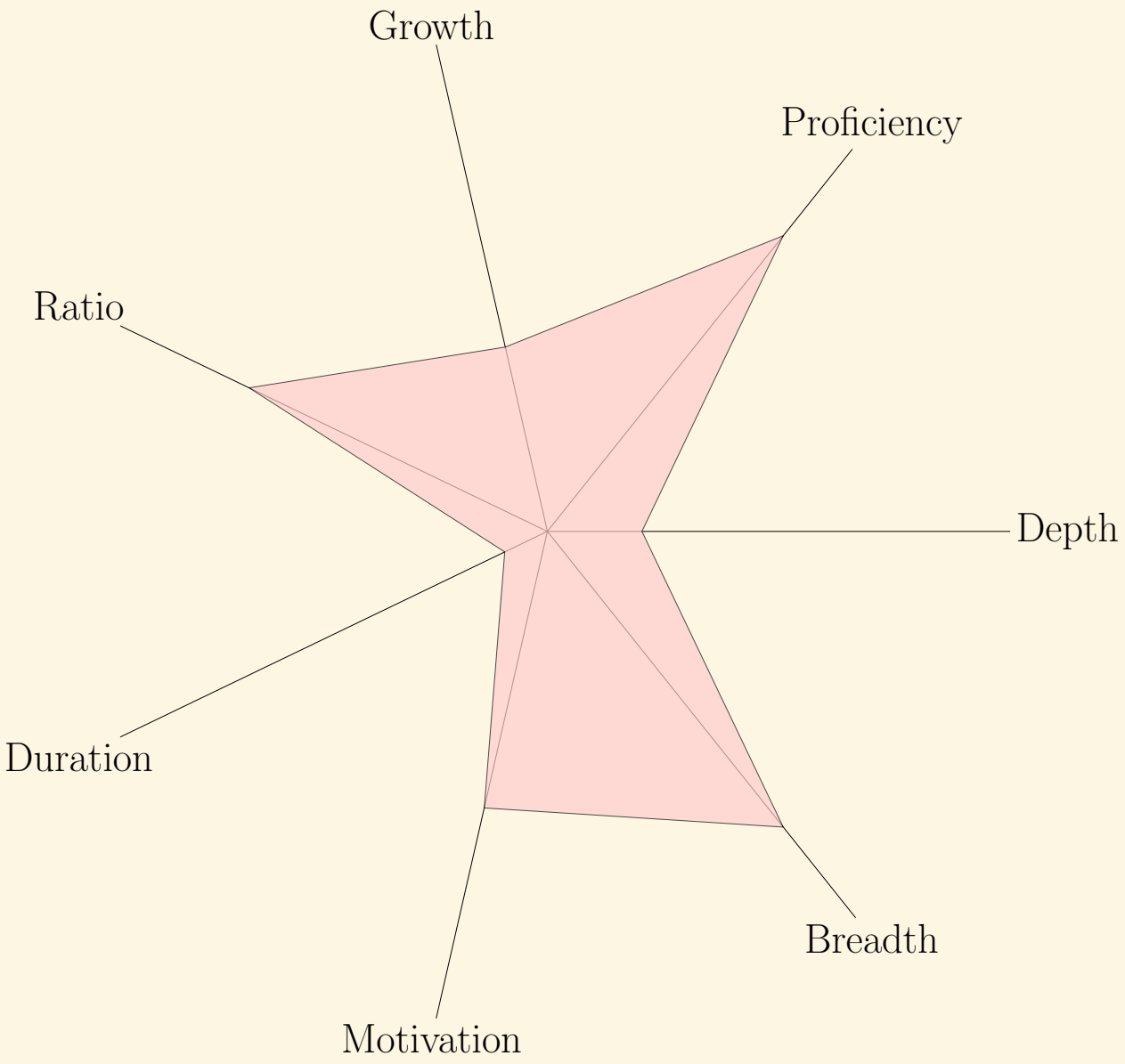
After creating this tournament, create a new strategy that wins the overall tournament.

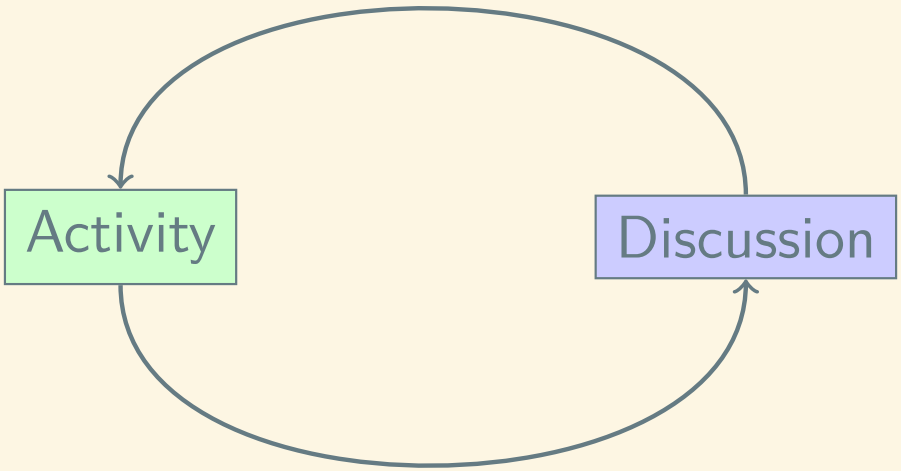
Parameters:

There will be 6 original strategies/players:

- Always play Rock;
- Always play Paper;
- Always play Scissors;
- Alternate: Rock then Paper then Scissors;
- Alternate: Rock then Scissors then Paper;
- Play randomly
- Mirror: Repeats the ‘previous move’ made by the oppponent.

PhD level research practice workshop




Activity
Discussion

We will do this by writing software to find the prime factors of an integer.

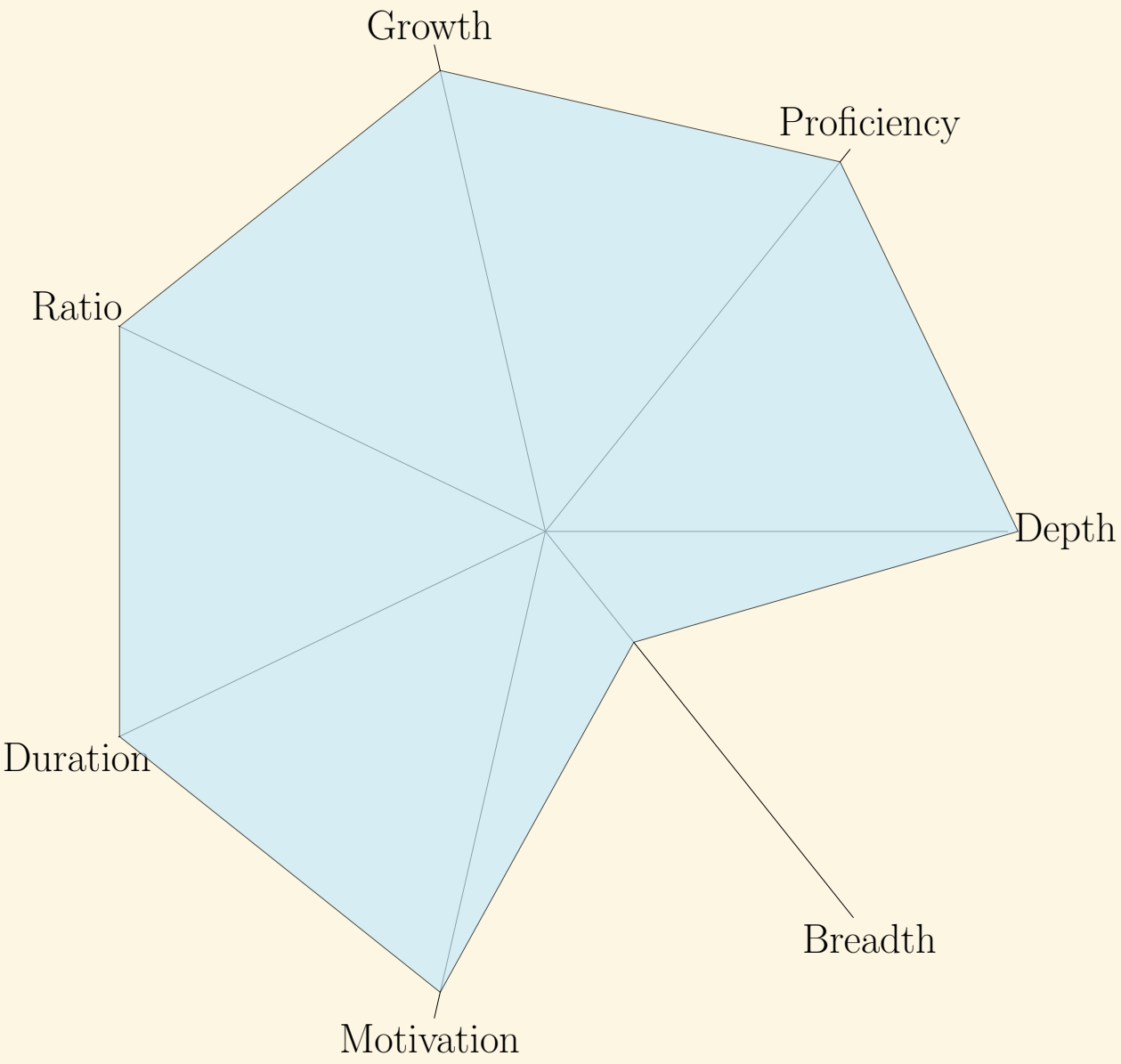
Discussion

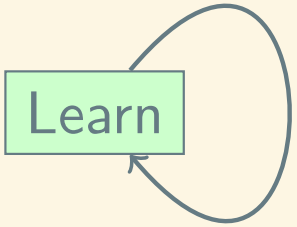
Before looking any further there will be a class discussion.

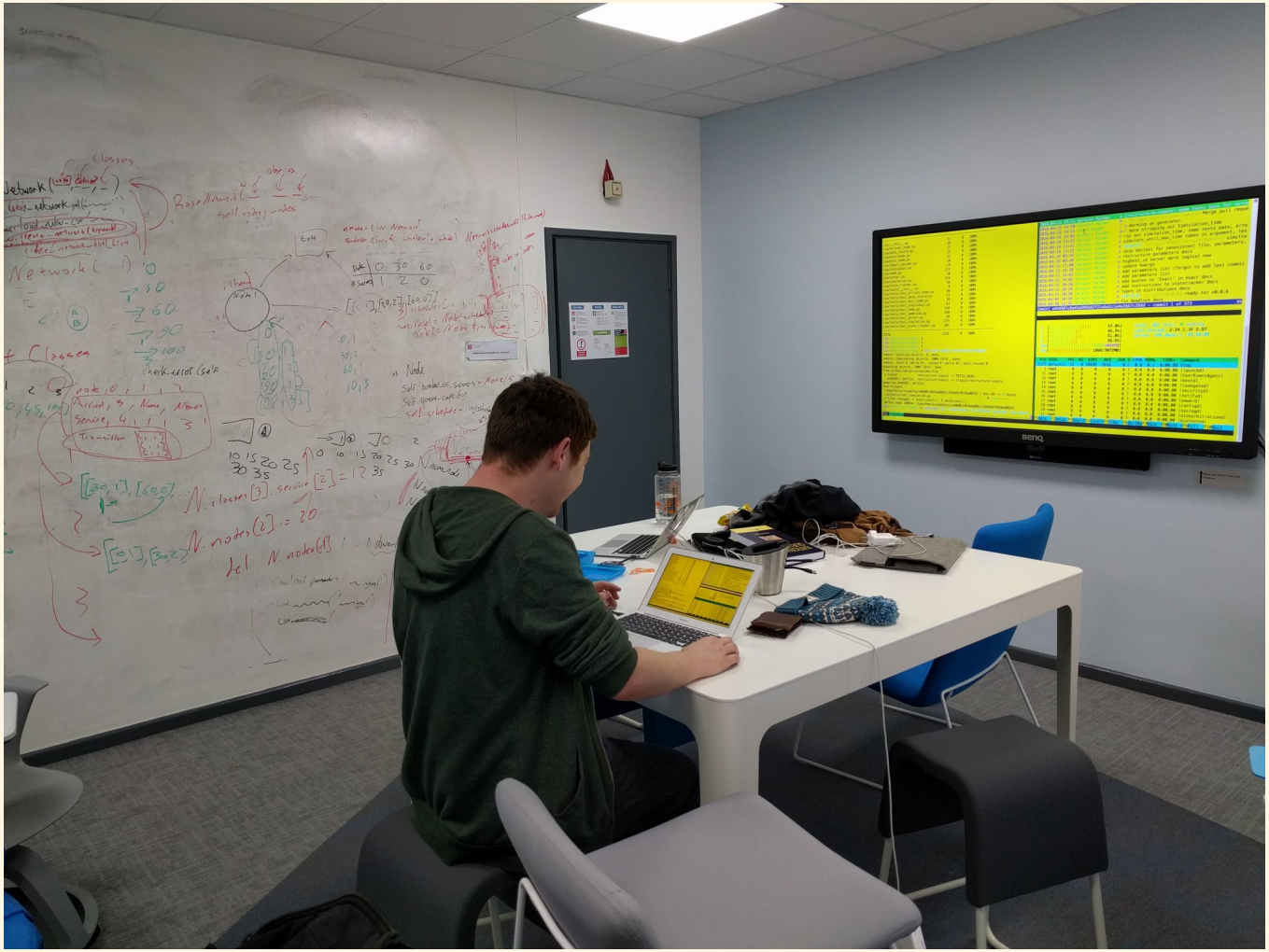
Files from discussion

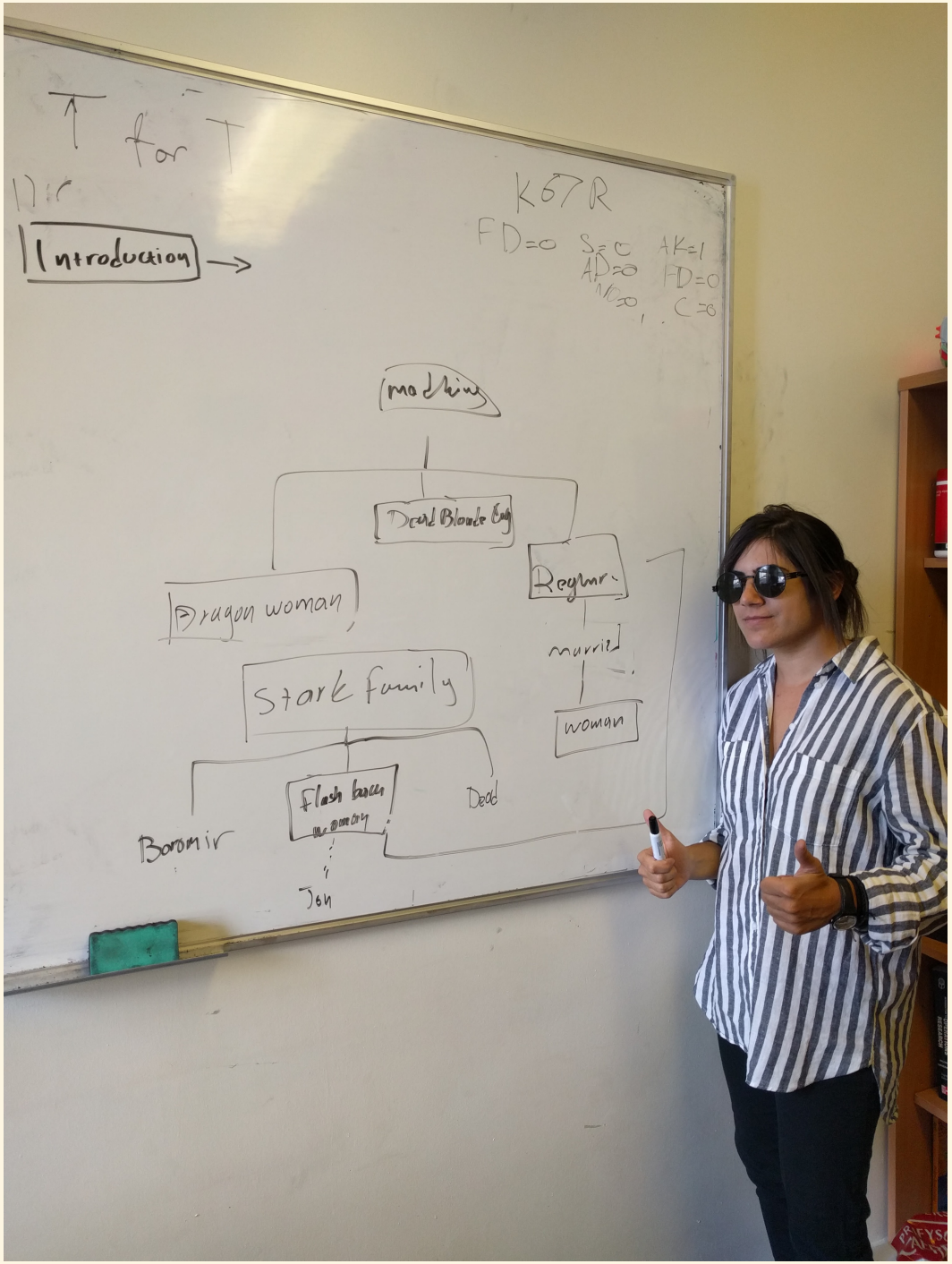
- `find_primes_v1.py`.
- `find_primes_v2.py`.
- `find_primes_v3.py`.
- `is_prime.py`
- `integer_division`
- `prime_factors.py`
- `test_is_prime.py`
- `test_integer_division.py`
- `test_prime_factors.py`

PhD supervision

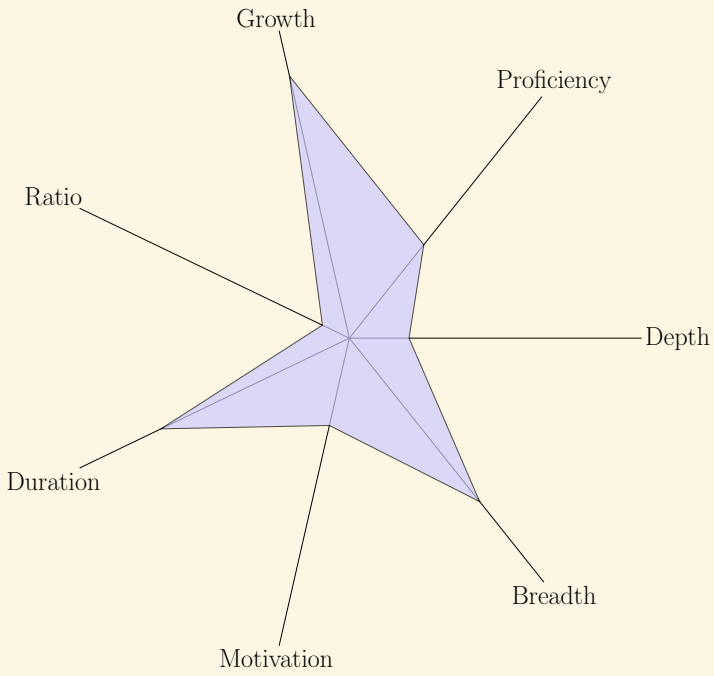


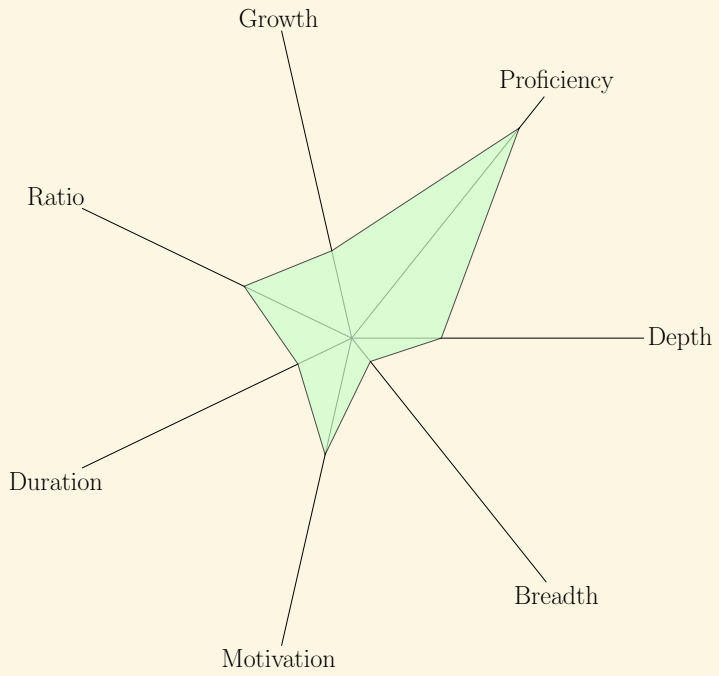

Learn

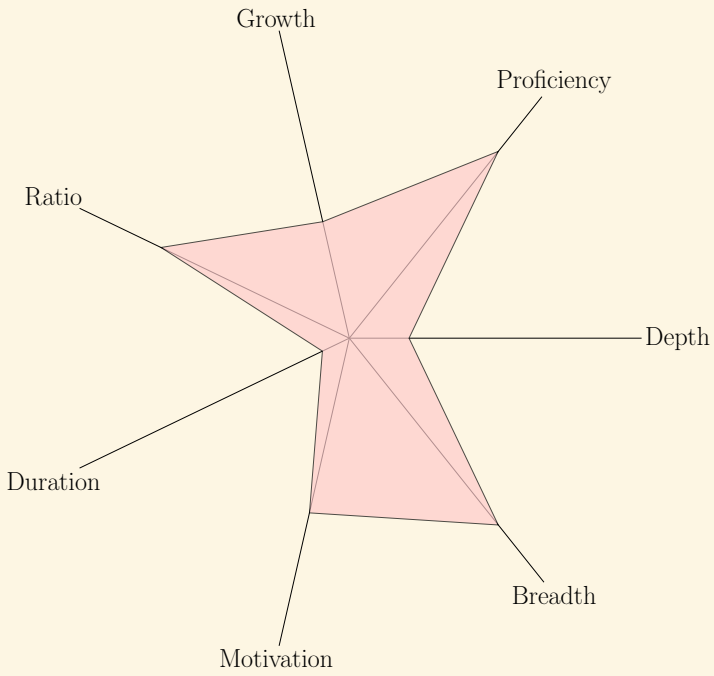

Classes
Network
Base_Network
Node
Transmitter
Arrived, s, Name, Nmax
Service, t, 1, 1, 3
Self.nodes = nodes
Self.links = links
Self.classes = classes
Self.queue = queue
Self.server = server
Self.queue_capacity
Self.schedule = schedule
check_error(self)
N.classes[3].service[2] = 12
N.nodes[2] = 20
N.nodes[3]
BENQ

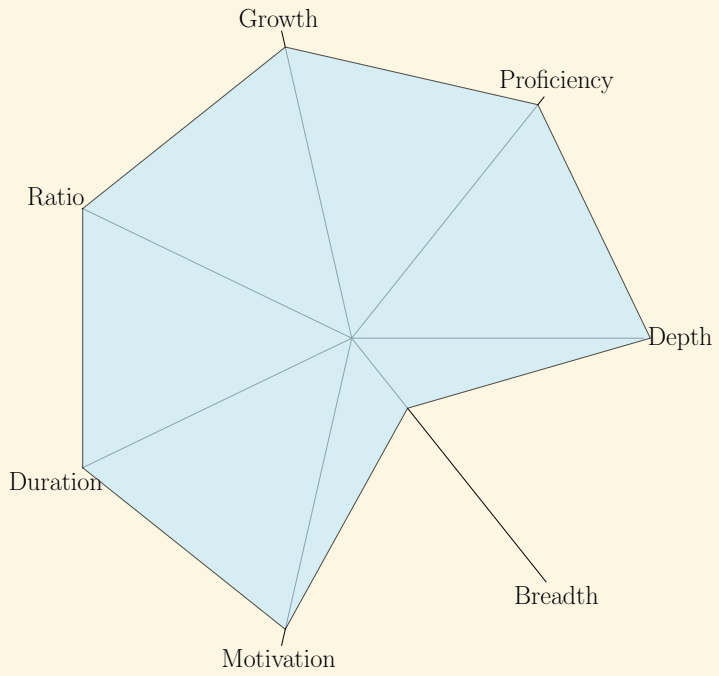


Introduction
K67R
FD=0
S=0
AK=1
AD=0
FD=0
C=0
modking
Dead Blonde Cog
Dragon woman
Stark family
married
woman
Flash back woman
Dead
Baromir
Jon


Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation


Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation


Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation


Growth
Proficiency
Ratio
Depth
Duration
Breadth
Motivation

Enthusiasm

- Anaconda
- Jupyter notebooks
- `pip install --user <package>`
- gh-pages
- @drvinceknight
- `knightva@cf.ac.uk`