**Start les** (5 min)

* Doel van de les: Recursief problemen oplossen zonder pc

**Opdracht 1** (30 min)

* Studenten Bedenken een strategie uit in de leerteams. groepjes van 4-6(15 min)
* Bespreek gevonden strategieën (10 min)
  + Laat een groepje hun strategie uitbeelden
* Bespreek juiste strategie in Python Tutor. (5 min)

**Opdracht 2** (10 min)

* Studenten Bedenken een strategie uit in de leerteams. groepjes van 4-6(10 min)

**pauze**

**Opdracht 2** (10 min)

* Bespreek gevonden strategieën (5 min)
* Bespreek juiste strategie in laat Python code zien. (5 min)

**Opdracht 3** (20 min)

* Studenten Bedenken een strategie uit in de leerteams. groepjes van 4-6(10 min)
* Bespreek gevonden strategieën (5 min)
* Bespreek juiste strategie in Python. (5 min)

**Opdracht 4** (15 min)

* Studenten Bedenken een strategie uit in de leerteams. groepjes van 4-6(5 min)
* Bespreek gevonden strategieën (5 min)
* Bespreek juiste strategie in Python. (5 min)

**Afronden**

Verwachting is dat de studenten overeenkomt in strategie beginnen te zien en de opdrachten steeds sneller kunnen oplossen.

**Antwoorden**

**Opdracht 1**

Elke dief pakt 1 papiertje. Als een dief 42 pakt uit de beker stopt hij met het doorgeven van de beker en fluistert hij True naar zijn buurman en. Die geeft het weer door aan zijn buurman en dat gaat door totdat de eerste dief de informatie heeft. Als een dief een lege beker krijgt, dan fluistert hij false naar zijn buurman. Die geeft het weer door aan zijn buurman en dat gaat door totdat de eerste dief de informatie heeft.

def find\_forty\_two(beker):

    if len(beker) == 0: # check of de beker leeg is.

        return False    # vertel je buurman dat de beker leeg is

    papiertje = beker[0]    # pak een papiertje

    if papiertje == 42:     # getal 42 is gevonden

        return True         # vertel je buurman dat 42 is gevonden

    return find\_forty\_two(beker[1:])    # geef de beker aan je buurmam en wacht op zijn antwoord. geef vervolgens zijn antwoord door aan je buurman

**Opdracht 2**

Elke dief pakt 1 papiertje. Als een dief het laatste papiertje pakt, dan fluistert hij dit getal naar zijn buurman. Deze dief check het gegeven getal met zijn eigen getal en fluistert het hoogste getal naar zijn buurman. Die vergelijkt het gegeven getal met zijn eigen getal en fluistert het hoogste getal naar zijn buurman, etc, etc, etc.

def find\_max(beker):

    if len(beker) == 1: # check of er nog maar één paprties in de beker zit.

        return beker[0]    # fluister het laatste getal naar je rechter buurman

    papiertje = beker[0]    # pak een papiertje

    buurman = find\_max(beker[1:]) # geef de beker aan de linkter buurman en wacht op antwoord

    if papiertje > buurman: # vergelijk jou getal met het getal dat je van de linker buurman hebt gekregen

        return papiertje    # je eigen getal is groter, dus dat geef je door aan de rechter buurman

    else:

        return buurman      # je linker buurman was groter if gelijk aan jou eigen getal dus dat geef je door aan de rechter buurman

**Opdracht 3**

Elke dief pakt 1 papiertje. Als een dief het laatste papiertje pakt, dan fluistert hij 1 naar zijn buurman. Deze telt er 1 bij op en fluistert 2 naar zijn buurman. Deze telt er ook 1 bij op en fluistert 3 naar zijn buurman, etc, etc etc.

def count(beker):

    if len(beker) == 1: # check of er nog maar één paprties in de beker zit.

        return 1    # fluister 1 naar je rechter buurman

    papiertje = beker[0]    # pak een papiertje

    buurman = count(beker[1:]) # geef de beker aan de linker buurman en wacht op antwoord

    count = buurman + 1        # tel 1 op bij het resultaat van je linker buurman.

    return count                # geef het resultaat door aan je rechter buurman.

**Opdracht 4**