Deel 1: Introductie

Stap 1: Teaser

Inleiding.

Demo simulatie van auto.
Enige gevoel krijgen voor achterliggende theorie.
Begrijpen PLC sweep, link met Arduino loop.
Hoe giet ik de theorie in de vorm van een programma.
Waar komt de besturing en begrijpen code-generatie.
Visualisatie en de rol van objecten, klassen en overerving hierin.
2e demo: wat moet realistisch zijn en wat doet er minder toe.

Stap 2: Wat kun je na deze les

Beoordelen of object georienteerd programmeren in een gegeven situatie voordelen biedt en beargumenteren waarom.
Instance variabelen creëren in de constructor, in afwijking van sommige andere programmeertalen (!)
Een eenvoudige simulatie maken van een optrekkende auto, met gebruikmaking van OOP, met name inkapseling.

Stap 3: Ophalen benodigde voorkennis

ERD's en UML class diagrams.
Structs.
Functies en parameters.
Interface en implementation.

Deel 2: Kern

Stap 4: Uitleg van de nieuwe lesstof

Korte samenvatting van de bijbehorende videoles, gelegenheid tot vragen.
Demo van maken klassen *Animal*, *Dog* en *Bird* met op de juiste plekken de methods *__init__*, *move* en *wagTail* en een polymorfe lijst van dieren. *Dog* en *Bird* hebben veschillende *move* methods. De methods drukken als tekst af wat het betreffende dier precies doet. In een reële situatie zullen zulke functies per soort iets heel verschillends doen.

Stap 5: Voorbeelden en vragen door docent

Bekijken broncode PLC simulator, met name modules *scene*, *engine* en *vectors*.
Module *vectors* werkt met tuples en globale functies, wat zijn argumenten voor en tegen een class *Vector* met methods in plaats van globale functies.

Stap 6: Oefenen met de nieuwe lesstof

Maak een klasse *Car* waarvan de objecten instance variables *position* en *speed* hebben.
Geef *Car* een method *drive (self, acceleration, deltaTime)* die:

- De nieuwe snelheid berekent volgens *self.speed += acceleration * deltaTime*
- De nieuwe plaats berekent volgens s*elf.position += self.speed * deltaTime*

Geef *Car* ook een method *report* die *self.position* en *self.speed* afdrukt.
Instantieer in je hoofdprograma object *car* van class *Car* en roep van dit object in een *while*-lus herhaald methods *drive* en *report* aan.

Stap 7: Feedback op gemaakte oefening

Een of meerdere leden van elke groep delen hun programma, docent en medestudenten geven feedback, evt. na vragen om nadere uitleg.
Vraag: Waarom is het vaak (maar niet altijd) verstandig het gebruik van niet-lokale variabelen te vermijden?

Deel 3: Afronding

Stap 8: Evalueren of deze les goed “geland” is

Eén of meer leden van elke groep stellen vragen en/of geven tips en/of tops.

Stap 9: Huiswerk om je de lesstof verder eigen te maken

Zie opdrachten-tab in MS-Teams.

Inleveren van de uitgewerkte opdrachten die bij een les horen is, samen met een positief verlopen eind-assessment, een noodzakelijke voorwaarde voor een voldoende en dient uiterlijk 2 volle dagen voor de volgende les plaats te vinden, uitsluitend op de geëigende wijze in MS-Teams.

De resultaten worden deels in de volgende les, deels individueel besproken. Maak de opdrachten op het door jou gekozen niveau zo goed mogelijk, maar wees ook niet bang om fouten te maken. Het gaat erom dat je een serieuze poging waagt en de docenten je indien nodig kunnen helpen om verder te komen.

--//--