


Opdracht 1: pannenkoeken van de bakker

Snap! runtime verificatie opdracht - 

Deze opgave gaat over een pannenkoekenbakker die pannenkoeken verkoopt. Zijn pannenkoeken zijn heel lekker, maar helaas kan de bakker niet zo goed rekenen. In Opdracht 1 kijken we wat de bakker fout doet. Dan introduceren we een blok voor het schrijven van robuuste scripts: het **assert** blok. In Opdracht 2 laten we je zien hoe je het **assert** blok moet gebruiken in de code. In Opdracht 3 leren we de bakker hoe hij goed moet rekenen.

We gaan ervan uit dat je al weet hoe je Snap! en Scratch moet programmeren. Is dit nieuw voor je? Vraag dan ons om hulp, of kijk eerst naar de Scratch opgaven. Die introduceren de basisconcepten van Snap! en Scratch.

Voordat we beginnen: ontmoet de bakker

Voordat we beginnen leggen we eerst uit hoe de bakker werkt! Dit is de pannenkoekenbakker. Hij verkoopt pannenkoeken voor 3 euro per pannenkoek. Er is een detail waar je op moet letten: onder zijn pannenkoekenkraampje staat een vierkantje met een nummer erin. In de foto hieronder staat daar “amount of batter **10**”. Dit nummer geeft aan hoeveel beslag de pannenkoekenbakker nog heeft, en hoeveel pannenkoeken hij dus kan bakken.



Taak 1: Pannenkoeken bestellen

Nu je snapt hoe de bakker werkt, voer het script een paar keer uit zodat je weet wat het doet. Als je daar klaar mee bent, probeer het volgende:




- Probeer 5 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?
- Probeer 15 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?
- Probeer -5 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?

De bakker doet het prima als je hem om 5 pannenkoeken vraagt. Maar de bakker kent zijn grenzen niet! Hij vindt 15 pannenkoeken bakken prima, ook al heeft hij maar genoeg beslag voor 10. De bakker vindt het ook prima om -5 pannenkoeken te bakken, ook al klopt dat van geen kant.

Antwoord

Nieuw blok: assert

Voor de volgende opgave moeten we je eerst leren over een nieuw blok. Dit blok heet het “**assert**” blok. Je kan het vinden in het “Besturen” paneel van Snap!.

Je kan dit blok zien als een blok om problemen mee te vinden. Wanneer Snap! dit blok uitvoert, kijkt Snap! eerst naar het blok binnenin het assert blok. Als het resultaat van het binnenste blok  is gaat Snap! verder naar het volgende blok. Als het resultaat  is geeft Snap! een foutmelding en gaat Snap! niet verder met het volgende blok. Wanneer het binnenste blok  is en er een foutmelding verschijnt, dan zeggen we dat “de assert afgegaan is”. Je kan je voorstellen dat, als je een probleem kan opschrijven als assert blok, dat het blok je dan helpt dat probleem te vinden zodra het gebeurt. Anders gezegd, als een assert afgaat, dan weet je dat er iets mis is!

Assert blok voorbeeld

Laten we een klein voorbeeld proberen om het beter te begrijpen. We maken een klein Snap! script dat je om een getal vraagt, en vervolgens een **assert** blok gebruikt om te zorgen dat de waarde gelijk is aan 3. In je Snap! project, maak de volgende blokken:



Klik dan op het “**wanneer ik aangeklikt word**” blok.

- Wat gebeurt er als je 5 invult? Is dat logisch?
- Wat gebeurt er als je 3 invult? Is dat logisch?

Antwoord

Als je 5 invult, zegt Snap! dat er wat mis is gegaan. Dit is logisch, want 5 is niet gelijk aan 3! Als je 3 invult, gebeurt er niks. Dat is logisch, want 3 is gelijk aan 3.

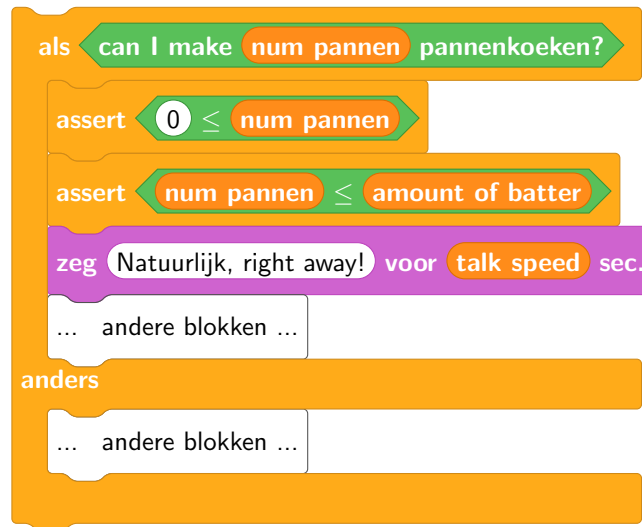
Taak 2: Problemen detecteren

Nu je het assert blok kent, gaan we het gebruiken om te detecteren wanneer de klant een aantal pannenkoeken vraagt wat niet kan.

In het script van de bakker kun je de volgende blokken vinden:



We gaan nu asserts toevoegen om te detecteren wanneer de baker een negatief of te groot aantal pannenkoeken wil bakken. Pas het blok als volgt aan:



Probeer nu nog een keer pannenkoeken te bestellen!

- Probeer 5 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?
- Probeer 15 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?
- Probeer -5 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?

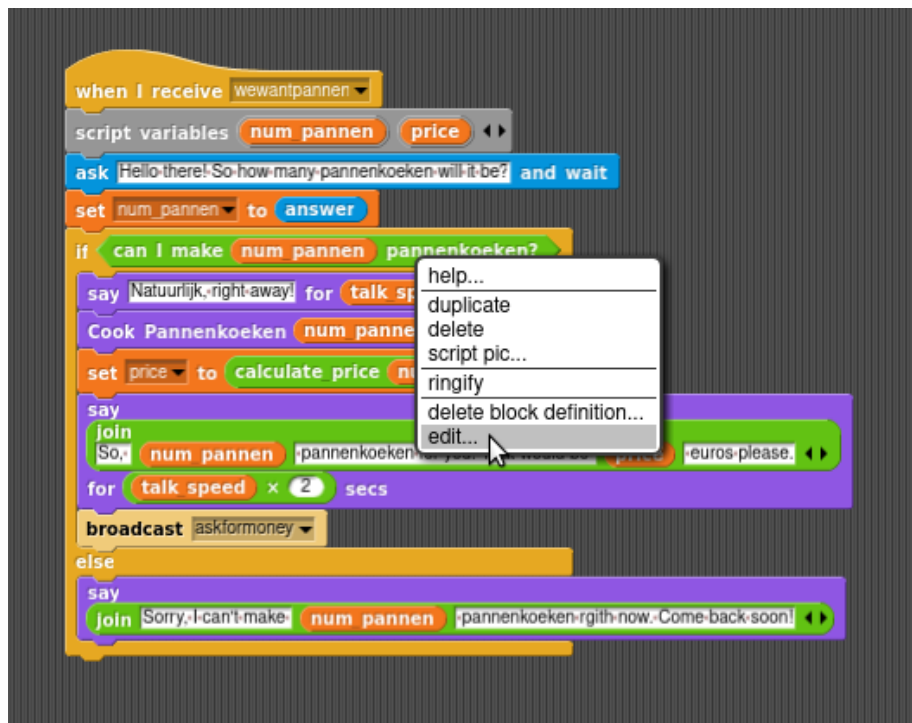
Antwoord

Als je de juiste asserts hebt toegevoegd zou 5 pannenkoeken bestellen geen problemen moeten opleveren. Maar bij -5 of 15 pannenkoeken geeft Snap! je een foutmelding dat een **assert** afgegaan is.

Dit gedrag van het script vind je misschien slechter dan wat de bakker eerst deed: misschien vond je het helemaal niet erg dat de bakker oneindig beslag heeft! Voor deze opdracht hebben we echter besloten dat het belangrijk is dat dit niet gebeurt. Daarom willen we weten wanneer er iets mis gaat!

Taak 3: Leer de bakker rekenen

Als de bakker zou kunnen rekenen, zouden de **assert** blokken nooit afgaan. Laten we de bakker leren te rekenen! We gaan hiervoor het **can I make** **pannenkoeken?** blok aanpassen. Klik met de rechtermuisknop op het blok en klik op “edit...”:



Het volgende scherm verschijnt:

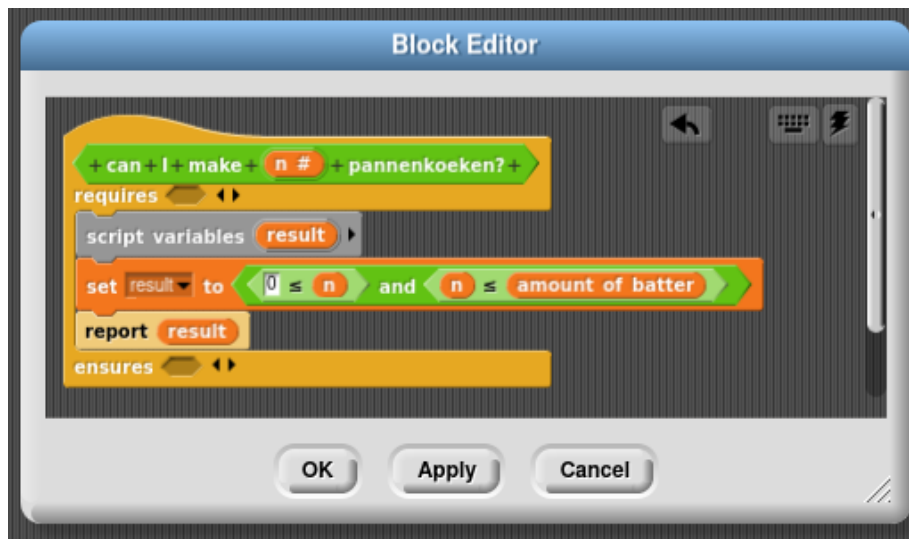


Oh nee! Het lijkt erop dat de bakker wel rekent of er genoeg beslag is, maar niet het juiste resultaat rapporteert. Kun je uitleggen waarom?

Antwoord

Binnenin **can I make** **pannenkoeken?** wordt **result** op **amount of batter** gezet wanneer **n** tussen de 0 en **amount of batter** is. Maar, **result** wordt daarna niet gebruikt! In plaats daarvan wordt **waar** gerapporteerd, wat betekent dat **result** wordt genegeerd. Daarom is het resultaat van het blok verkeerd.

Om dit te repareren moet je **result** rapporteren in plaats van **waar**. Om dit aan te passen, sleep **result** vanuit het blok **script variables** naar het **report** blok. Na de aanpassing moet je blok er zo uitzien:



Klik op “OK”, en probeer alle mogelijkheden weer!

- Probeer 5 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?
- Probeer 15 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?
- Probeer -5 pannenkoeken te bestellen. Wat gebeurt er? Is dit logisch?

Antwoord

Wanneer je 5 pannenkoeken bestelt, bakt de bakker 5 pannenkoeken. Wanneer je er 15 of -5 besteld zegt de bakker dat dit niet kan.

Je hebt de bakker rekenen geleerd, en de asserts gaan niet meer af. Goed gedaan!

Samenvatting

Door Taak 1, 2 en 3 te doen heb je geleerd problemen te vinden door naar een Snap! script te kijken, het uit te voeren en het probleem te formuleren als assert blokken die het probleem detecteren. Dit is een voorbeeld van “**een goeie manier van werken in software engineering**”. Dat wil zeggen, wanneer je een aanname doet over het gedrag van je programma, is het nuttig om deze aanname expliciet te maken met een assert. Dit kan niet alleen dienst doen als documentatie voor iemand anders die je code leest, maar ook om te zorgen dat je een foutmelding krijgt als de aanname om een of andere reden niet meer klopt.

In de situatie van de bakker betekent dit: als we ooit de code aanpassen, en misschien een fout veroorzaken, krijgen we nu een foutmelding wanneer het aantal bestelde pannenkoeken onder 0 of boven de hoeveelheid beslag gaat. Dit geeft ons meer vertrouwen in het gedrag van de bakker.