Python GGH Lesplan 2 16 April

Introductie
-Zorg dat IDLE bij iedereen werkt.
-Laat zelf een mooi programma zien van eigen maak (bijv. Roosterapp / Encryptie)
-Nabespreking huiswerk:
- Hoe ging het huiswerk?
- Laat werk van leerlingen, indien mogelijk zien.
- Welke hulpbronnen hebben leerlingen gebruikt?
- Oplossingsmethodes

-Nabespreking Les 1, bevraag leerlingen kort:
Gebruik print(), input(), comments, (herkennen van) types, assignment operator =, soorten operatoren, comparison operators, if statement. Herinner leerlingen aan gebruik haakjes / rekenvolgorde.

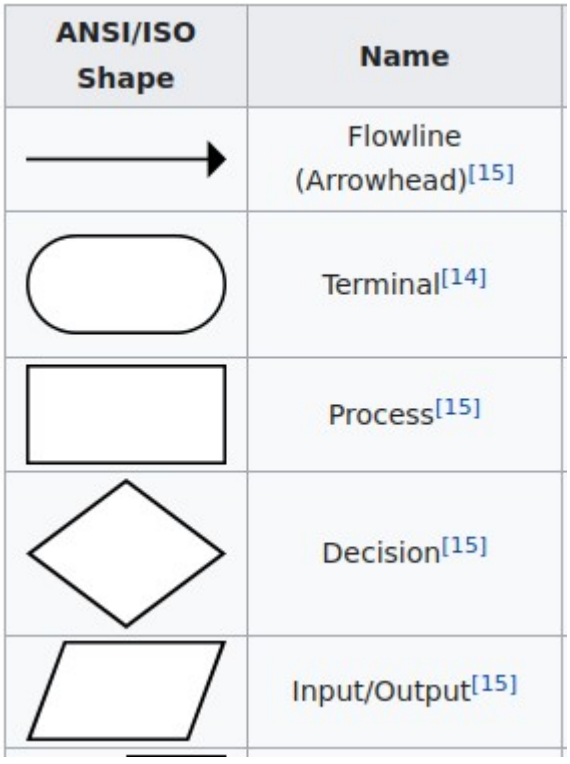
Middenstuk
Typecasting:
-Leg uit wat 'typecasting' betekent, demonstreet int(), float(), string(), bool(), in het bijzonder wat dezen met strings doen en vice-versa wat string() doet.

Extra assignment operators:
-Leg uit dat a = a + … etc. combinaties voorkomen en dat je dus de bekende operatoren +, -, *, /, //, ** kunt combineren met = als afkorting a += 1.

Expressions, (conditional) statements
Leg uit dat je in Python elke (niet lege) regel een statement is. Je hebt twee soorten:
-(reguliere) statements. Deze doen iets, zoals een declaratie =, of een functie print().
-Conditional statements: die beïnvloeden de flow van het programma, zoals if, else en while.
Daarnaast kan een statement een 'expression' bevatten. Een expression is een uitdrukking, een combinatie van operatoren / functies, met een waarde. Een expression herken je als de rechterkant van een toewijzing =. Expressions kun je ook afdrukken met print().

While:
-Introduceer de 'while' conditional statement. Leg uit aan de hand van een voorbeeld waarin input wordt genomen en gesomd totdat 0 als input wordt gegeven.

Flowcharts:
THEORETISCH INTERMEZZO OP BORD:
-Leg uit dat onze programmas nu "Turing-compleet zijn", wie Alan Turing was (benoem Enigma machine, Imitation game), en wat Turing compleet is.
-Leg uit dat de kern van programmeren algoritmes ontwerpen is, en wat een algoritme is.
-Introduceer flowcharts als middel om programmas te ontwerpen. Raad aan deze altijd te tekenen voor het schrijven van een programma.
-Leg de flowchart regels uit.
(https://en.wikipedia.org/wiki/Flowchart)



-Maak samen een flowchart voor het vorige programma.
-Maak samen een flowchart voor een rekenmachine.

Continue en Break:
-Licht de continue en break statements toe.
-Illustreer continue en break aan de hand van een tekstverwerker die steeds nieuwe regels inleest. Als een lege regel wordt geschreven, slaat hij over, als een 000 wordt geschreven, stopt hij.
-Een programma dat y/n vraagt. Als y/n wordt ingevoerd, doet hij iets, en breakt hij de while loop. Als iets anders wordt ingevoerd start hij opnieuw.
-Een programma dat van 1 tot 100 telt. Als een getal even is, slaat hij over met continue.

Logical Operators:
-Leg de logical operators (and, or, not) toe.
-Voorbeeld: systeemborden.

Simpele Lists
-Leg uit hoe je een list maakt met a =[].
-Leg uit hoe je de lengte van een list neemt met len()
-Leg uit hoe je een list item kunt selecteren met [index].
-Leg uit hoe je een list kunt extenden met +=.
-Voorbeeld: neem lijst van namen, en output deze weer.
-Controleer of een getal in een lijst even is.

Strings
-Leg uit dat een string lijkt op een lijst karakters. Niet alle karakters kun je zomaar in een string coderen: tab, newline, aanhalingstekens, single/double quotes. Deze moet men escapen met een backslash '\'. De backslash zelf moet ook worden escaped. *Nota bene: er zijn een aantal bijzondere ASCII karakters, en dan vooral de '\a', bell!*
-Leg uit dat een print statement altijd '\n' aan het einde zet. Als je dit niet wil hebben, gebruik dan ', end='.

Workshop
Bespreek tijdens workshop Bugs & Errors:
Er zijn drie soorten errors in een python programma:
-Syntax errors: de code is incorrect. De grammatica is fout. Bijvoorbeeld een vergeten haakje, een misplaatse komma, een foute naam. Deze kun je opsporen met een Syntax checker in IDLE.
-Runtime errors: er is iets misgegaan tijdens het runnen van het Python programma, bij voorbeeld divide by zero, of wanneer je een niet-bestaande variabele probeer te gebruiken. Deze kun je vinden door je programma te runnen.
-Bugs: dit zijn logische fouten in je programma. Deze zijn het ernstigst, want je programma kan lijken te werken, maar toch iets verkeerds doen. Om deze op te sporen, moet je redeneren over je programma, en hem testen op verschillende gevallen. Voorbeelden zijn wanneer de programmeur de verkeerde rekenvolgorde van operatoren aanhoudt, of het type van een variabele verkeerd aanneemt, bijv. Int vs float. Bugs kunnen zeer ernstig zijn wanneer er sprake is van een security vulnerability. Zo had kon je in Python 2 door de input() functie functies aanroepen en met bijvoorbeeld time.sleep(10) het programma 10 seconden laten slapen. Daarom moest men altijd raw_input() gebruiken.

Einde les:
-Bespreek wat we geleerd hebben.
-Als de leerlingen willen bijlezen, kunnen ze het best op https://cscircles.cemc.uwaterloo.ca/nl/ lessen 6D t/m 9 + 13 maken.
-Vraag leerlingen volgende les met voorstellen te komen voor themas / projecten les 5-8. Dat wordt in les 3 besproken.