Hoorcollege 1: Wiskundige berekeningen/Variabelen

App 1: De afstand die een auto reist op een snelweg kan berekend worden als volgt: Afstand = Snelheid x Tijd. Ontwikkel een python programma waarin de gebruiker zijn snelheid kan ingeven (in km/uur) en een tijdsduur. Het pytho programma berekend dan de afstand die afgelegd worden per uur. Dit wordt op de console weergegeven.

App 2: De volgende formule converteert de temperatuur van Celsius naar Fahrenheit (F): F = 5/9*C+32. De volgende formule converteert de temperatuur van Fahrenheit naar Celsius: C=5/9*(F-32). Ontwikkel twee python programmaatjes waarin de gebruiker de temperatuur ingeeft. Het ene programma zet de temperatuur om van F naar C. Het andere python programma zet de temperatuur om van C naar F.

App 3: Definieer drie variabelen var1, var2 en var3. Bereken het gemiddelde en stop het in een variabele gemiddelde. Toon het gemiddelde.

App 4: Lees een getal in die een straal voorstelt. Schrijf code die de oppervlakte en de omtrek van een cirkel berekent, gebruik makend van variabelen straal en pi = 3.14159. Toon de uitkomst als volgt: “De oppervlakte van een cirkel met straal … is … en de omtrek is …”

App 5: De stelling van Pythagoras zegt dat bij een rechthoekige driehoek het kwadraat van de schuine zijde gelijk is aan de som van de kwadraten van de twee andere zijden. Schrijf een programma dat de gebruiker om de lengte van de twee rechte zijden vraagt, en bereken dan de lengte van de schuine zijde (met andere woorden, trek de wortel uit de som van de kwadraten van de twee rechte zijden). Toon dit resultaat op de console.

Hoorcollege 2: Condities en iteraties

App 1: ontwikkel een programma dat de BMI (body mass index) berekent van een persoon. De gebruiker geeft zijn lengte (in meter) en zijn gewicht (in kg). Het BMI wordt als volgt berekend: BMI = gewicht/(lengte x lengte). Voeg een melding toe van de categorie waartoe de persoon behoort:

BMI index	Categorie
< 18.5	Ondergewicht
18.5 – 25.0	Gezond gewicht
25.0 – 30.0	Overgewicht
> 30	Obesitas

App 2: Ontwikkel een app die een testscore leest, en dan de graad toont. Volgende graadschaal wordt gebruikt:

BMI index	Categorie
< 18.5	Ondergewicht
18.5 – 25.0	Gezond gewicht
25.0 – 30.0	Overgewicht
> 30	Obesitas

App 3: Je kunt kwadratische vergelijkingen oplossen met de wortelformule. Kwadratische vergelijkingen hebben de vorm Ax2 + Bx + C = 0. Dit soort vergelijkingen heeft nul, één of twee oplossingen. De eerste oplossing is (−B + sqrt(B 2 −4AC))/(2A). De tweede oplossing is (−B − sqrt(B 2 − 4AC))/(2A). Er zijn geen oplossingen als de waarde onder de wortel negatief is. Er is één oplossing als de waarde onder de wortel nul is. Schrijf een programma dat de gebruiker vraagt om waardes voor A, B, en C, en dan rapporteert hoeveel oplossingen er zijn, en welke oplossingen dat zijn. Let erop dat je ook afhandelt wat er gebeurt als A nul is (er is dan één oplossing, namelijk −C/B), of als A en B allebei nul zijn.

App 4: Schrijf een programma dat de gebruiker een getal laat ingeven. Het programma geeft de tafel van vermenigvuldiging van het getal voor 1 tot en met 10. Bijvoorbeeld, als de gebruiker 12 ingeeft, dan is de eerste regel die afgedrukt wordt "1 * 12 = 12" en de laatste regel "10 * 12 = 120"

App 5: "*99 bottles of beer"* is a traditioneel liedje gezongen in Amerika en Canada. Het wordt vaak gezongen op lange reizen omdat het gemakkelijk te onthouden en mee te zingen is, en lang duurt. In vertaling is de tekst: *"99 flesjes met bier op de muur, 99 flesjes met bier. Open er een, drink hem meteen, 98 flesjes met bier op de muur."* Deze tekst wordt herhaald, steeds met één flesje minder. Het lied is voorbij als de zangers nul bereiken. Schrijf een programma dat het hele lied afdrukt (ik raad je aan te beginnen met niet meer dan 10 flesjes). Kijk uit dat je je loop niet eindeloos maakt. Zorg er ook voor dat je het juiste meervoud voor het woord "flesje" gebruikt.

App 6: Schrijf een programma dat een toevalsgetal neemt tussen 1 en 1000 (je kunt randint() daarvoor gebruiken). Het programma vraagt de gebruiker het getal te raden. Na iedere poging van de gebruiker zegt het programma "Lager" als het te raden getal lager is, "Hoger" als het te raden getal hoger is, of "Je hebt het geraden!" als het getal correct is. Het programma eindigt met afdrukken hoeveel pogingen de gebruiker nodig had om het getal te raden. Voor test-doeleinden kan het slim zijn om het te raden getal op het scherm te laten zien, totdat je zeker weet dat het programma goed werkt.

App 7: Schrijf een app die het aantal cijfers van een geheel getal telt (tip: het getal herhaaldelijk delen door 10).

App 8: lees een geheel getal in en bepaal of het een volkomen kwadraat is.

$1^2 = 1$	$21^2 = 441$	$41^2 = 1681$	$61^2 = 3721$	$81^2 = 6561$
$2^2 = 4$	$22^2 = 484$	$42^2 = 1764$	$62^2 = 3844$	$82^2 = 6724$
$3^2 = 9$	$23^2 = 529$	$43^2 = 1849$	$63^2 = 3969$	$83^2 = 6889$
$4^2 = 16$	$24^2 = 576$	$44^2 = 1936$	$64^2 = 4096$	$84^2 = 7056$
$5^2 = 25$	$25^2 = 625$	$45^2 = 2025$	$65^2 = 4225$	$85^2 = 7225$
$6^2 = 36$	$26^2 = 676$	$46^2 = 2116$	$66^2 = 4356$	$86^2 = 7396$
$7^2 = 49$	$27^2 = 729$	$47^2 = 2209$	$67^2 = 4489$	$87^2 = 7569$
$8^2 = 64$	$28^2 = 784$	$48^2 = 2304$	$68^2 = 4624$	$88^2 = 7744$
$9^2 = 81$	$29^2 = 841$	$49^2 = 2401$	$69^2 = 4761$	$89^2 = 7921$
$10^2 = 100$	$30^2 = 900$	$50^2 = 2500$	$70^2 = 4900$	$90^2 = 8100$
$11^2 = 121$	$31^2 = 961$	$51^2 = 2601$	$71^2 = 5041$	$91^2 = 8281$
$12^2 = 144$	$32^2 = 1024$	$52^2 = 2704$	$72^2 = 5184$	$92^2 = 8464$
$13^2 = 169$	$33^2 = 1089$	$53^2 = 2809$	$73^2 = 5329$	$93^2 = 8649$
$14^2 = 196$	$34^2 = 1156$	$54^2 = 2916$	$74^2 = 5476$	$94^2 = 8836$
$15^2 = 225$	$35^2 = 1225$	$55^2 = 3025$	$75^2 = 5625$	$95^2 = 9025$
$16^2 = 256$	$36^2 = 1296$	$56^2 = 3136$	$76^2 = 5776$	$96^2 = 9216$
$17^2 = 289$	$37^2 = 1369$	$57^2 = 3249$	$77^2 = 5929$	$97^2 = 9409$
$18^2 = 324$	$38^2 = 1444$	$58^2 = 3364$	$78^2 = 6084$	$98^2 = 9604$
$19^2 = 361$	$39^2 = 1521$	$59^2 = 3481$	$79^2 = 6241$	$99^2 = 9801$
$20^2 = 400$	$40^2 = 1600$	$60^2 = 3600$	$80^2 = 6400$	$100^2 = 10000$

Hoorcollege 3: Tuples

App 1: Een complex getal is een getal van de vorm a + bi, waarbij a en b constanten zijn, en i een speciale waarde, die gedefinieerd is als de wortel uit -1. Natuurlijk kun je niet echt de wortel uit -1 berekenen, dat zou een runtime error geven; in complexe berekeningen laat je altijd de i staan. Bijvoorbeeld, het complexe getal 3 + 2i kan niet verder gesimplificeerd worden. Het optellen van complexe getallen a + bi en c + di is gedefinieerd als (a + c) + (b + d)i. Representeer een complex getal als een tuple met twee numerieke waardes, en creëer een tuple die de optelling van twee complexe getallen implementeert en eentje die de vermenigvuldiging ervan implementeert. De vermenigvuldiging van twee complexe getallen a + bi en c + di is gedefinieerd als (a*c - b*d) + (a*d + b*c)i. Druk vervolgens beide complexe getallen af.

App 2: De zeef van Eratosthenes is een methode om alle priemgetallen te vinden tussen 1 en een gegeven getal, gebruik makend van een list. Dit werkt als volgt. Je begint met een list te maken die bestaat uit de getallen 1 tot en met een zeker "hoogste getal." Zet de waarde van 1 op nul, omdat 1 geen priemgetal is. Verwerk nu de list in een loop. Zoek naar het eerste nummer dat niet op 0 staat, wat nummer 2 is. Dat betekent dat 2 een priemgetal is, maar alle veelvouden van 2 zijn dat niet. Dus zet alle veelvouden van 2 op 0. Zoek dan naar het volgende nummer dat geen nul is, en dat is 3. Zet alle veelvouden van 3 op nul. Zoek dan naar het volgende nummer dat geen nul is, en dat is 5. Zet alle veelvouden van 5 op nul. Verwerk zo de hele list. Als je klaar bent, zijn alleen nog de getallen over die priemgetallen zijn. Gebruik deze methode om alle priemgetallen tussen 1 en 100 te bepalen.

Hoorcollege 6: Functies/Strings/Dictionaries

App 1 (functies): Maak een functie die als parameter een getal krijgt, en die dan de tafel van vermenigvuldiging voor 1 tot en met 10 van dat getal afdrukt. Bijvoorbeeld, als de parameter 12 is, dan drukt het programma als eerste regel "1 * 12 = 12" af, en als laatste regel "10 * 12 = 120."

App 2 (functies): De Grerory-Leibnitz reeks benadert de waarde van π door de berekening van 4 ∗ (1/1 − 1/3 + 1/5 − 1/7 + 1/9...). Schrijf een functie die π benadert via deze reeks. De functie krijgt één parameter, namelijk een integer die aangeeft hoeveel van de termen tussen de haakjes in de reeks berekend moeten worden.

App 3 (functies): In een eerder hoofdstuk werd je gevraagd de wortelformule te implementeren om kwadratische vergelijkingen op te lossen. Een kwadratische vergelijking wordt beschreven door drie numerieke waardes, gewoonlijk A, B, en C genoemd. De vergelijking heeft nul, één, of twee oplossingen, afhankelijk van de discriminant (het deel van de vergelijking onder de wortel). Schrijf een functie die een kwadratische vergelijking kan oplossen. Als parameters krijgt het A, B, en C. Het retourneert drie waardes. De eerste is een integer die het aantal oplossingen aangeeft. De tweede is de eerste oplossing. De derde is de tweede oplossing. Als een oplossing niet bestaat, kun je een nul retourneren voor de corresponderende retourwaarde.

App 4 (strings): Tel hoeveel er van iedere klinker (a, e, i, o, u) staan in een tekst string, en druk die teller af voor iedere klinker met een enkele geformatteerde string. Bedenk dat klinkers zowel hoofd- als kleine letters kunnen zijn.

App 5 (strings): Hieronder staat een tekst met een aantal tekens tussen vierkante haken. Doorloop de tekst en druk alle tekens af die tussen vierkante haken staan.

```
tekst = """En ze stu[re]n [i]ngekleurde prentbriefkaarten van plekken waarvan ze zich niet reali[s]eren dat ze er nooit geweest zijn [a]an ' Iedereen op nummer 22, weer is prachti[g], onz[e] kamer is aa[n]gekruisd. Missen jullie. E[t]en[ ]i[s] vettig , maar we hebben een geweldig leuk restaurantje gevonden in de achterstraatjes waar ze Heine[ke]n hebben en kaas en uien chips en iemand die "Een beetje verliefd" speel[t] op een a[c]cordeon ' en je zit vier dagen vast op Schip[h]ol voor je vijfdaagse vliegvakantie met niks anders te eten dan uitgedroogde voorverpakte boterhammen..."""
```

App 6 (strings): Druk een regel af met alle hoofdletters "A" tot en met "Z". Druk eronder een regel af die op 13 letters afstand in het alfabet liggen ten opzichte van de letters erboven. Bijvoorbeeld, onder de "A" druk je de "N" af, onder de "B" druk je de "O" af, etcetera. Beschouw het alfabet als circulair, dat wil zeggen, na de "Z", gaat het weer terug naar de "A". Dit kan natuurlijk met twee print-commando's, maar probeer het te doen met loops en gebruik te maken van ord() en chr().

App 7 (strings): Schrijf een programma dat een string neemt en er een nieuwe string van maakt die precies dezelfde tekens bevat als de originele string, maar in volgorde van hun ASCII codes. Bijvoorbeeld, de string "Hello, world!" geeft als resultaat de string " !,Hdellloorw". Dit kan vrij gemakkelijk gedaan worden met "list" functies, maar die komen pas aan bod in hoofdstuk 12, dus probeer het nu te doen met string manipulatie.

App 8 (strings): Schrijf een programma dat een tekst neemt (bijvoorbeeld de tekst hieronoder), de tekst splitst in woorden (waarbij alles dat geen letter is beschouwd wordt als een woord-scheider), en een dictionary bouwt die voor ieder woord (case-insensitive) opslaat hoe vaak het woord voorkomt in de tekst. Print dan alle woorden met hun hoeveelheden in alfabetische volgorde.

```
tekst = "Kapper Knap , de knappe kapper , knipt en kapt heel knap , maar de knecht van kapper Knap , de knappe kapper , knipt en kapt nog knapper dan kapper Knap , de knappe kapper."
```

App 9 (dictionaries): De code hieronder bevat een list van films. Voor iedere film is er ook een list met scores. Verander deze code zo dat het alle data opslaat in één dictionary, en gebruik dan de dictionary om de gemiddelde score voor iedere film af te drukken, afgerond op één decimal

```
films = ["Monty Python and the Holy Grail",
         "Monty Python ' s Life of Brian",
         "Monty Python ' s Meaning of Life",
         "And Now For Something Completely Different"]
grail_scores = [ 9, 10, 9.5, 8.5, 3, 7.5 ,8 ]
brian_scores = [ 10, 10, 0, 9, 1, 8, 7.5, 8, 6, 9 ]
life_scores = [ 7, 6, 5 ]
different_scores = [ 6, 5, 6, 6 ]
```