Opdrachten Python Fundamentals

Opdracht 1.1 – Python prompt

Experimenteer met de Python prompt

1. Open een Python prompt
    a. via IDLE
    b. in een command window met de opdracht: python
    c. in een IDE zoals PyCharm
2. Voer verschillende numerieke berekeningen uit
3. Gebruik de print functie om 'Hello World' op de console te uit te voeren

Opdracht 1.2 – Hello

Creëer een Python module en voer deze uit.

1. Open IDLE of een andere IDE.
2. Creëer een nieuwe Python bestand met de naam first.py.
3. Sla het bestand op in een aparte map.
4. Gebruik de **print()** functie om 'Hello World' te uit te voeren.
5. Sla het bestand op.
6. Voer het bestand uit.
7. Verander de code zodat er eerst naar een naam wordt gevraagd met de **input()** functie. Sla het resultaat op in een variabele **naam.**
8. Gebruik de **print()** functie om 'Hello Jan' te uit te voeren (als je Jan hebt ingevoerd).
9. Sla het bestand op.
10. Voer het bestand uit.

Opdracht 1.3 – Schikkeljaar

Schrijf een programma die bepaald of een bepaalde jaar een schikkeljaar is of niet.

1. Creëer een nieuw Python bestand.
2. Vraag met de **input()** functie aan de gebruiker om een jaartal in te voeren.
3. Verander het resultaat in een getal met de **int()** functie.
4. Bepaal of het jaar een schikkeljaar is. Het is een schikkeljaar als:
    a. het jaartal kan worden gedeeld door 4
    b. behalve als (and) het jaartal kan worden gedeeld door 100
    c. tenzij (or) het jaartal kan worden gedeeld door 400
5. Print het resultaat.
6. Test jouw programma voor verschillende jaren.

Tip: Om te bepalen of een getal kan worden gedeeld door een ander getal kun je de modulo operator (%) gebruiken om de rest na een deling te vergelijken met 0. Bv.: 2021 % 4 == 0.

Opdracht 1.4 – Berekeningen aan een cirkel

Schrijf een programma die de oppervlakte en omtrek van een cirkel berekent.

Stappen:

1. Creëer een nieuw Python bestand.
2. Importeer de math bibliotheek met **import math**.
3. Vraag de gebruiker met de **input()** functie om de straal van een cirkel.
4. Verander het resultaat in een getal met de **float()** functie en sla op in een variabele **r**.
5. Bereken de oppervlakte met de formule πr^2
6. Bereken de omtrek met de formule $2\pi r$
7. Print de resultaten

Tip: De math blibliotheek heeft de waarde van **π** in de variabele math.pi.

Opdracht 1.5 – Dobbelstenen

Schrijf een programma die het gooien van 5 dobbelstenen simuleert.

Stappen:

1. Creëer een nieuw Python bestand.
2. Importeer de random bibliotheek met **import random**.
3. Geneer een willekeurige getal tussen 1 en 6 met de functie **random.randint(1, 6)** en sla het resultaat op in een variabele **dice1**.
4. Herhaal dit nog 4 keer en creëer de variabelen dice2 tot en met dice5.
5. Print de waarden van de dobbelstenen.
6. Print ook het totaal som van de waarden.

Opdracht 1.6 – Strings

Experimenteer met strings.

Stappen:

1. Creëer een nieuw Python bestand.
2. Vraag de gebruiker om een stukje tekst in te voeren met de functie **input()** en sla het resultaat op in een variabele **t**.
3. Print de teskt in allemaal hoofdletters met de **upper()** methode en in allemaal kleine letters met **lower()**.
4. Doe hetzelfde met de methoden **capitalize()** en **title()**.
5. Print de eerste drie karakters met behulp van slicing.
6. Controleer of de tekst eindigt met een vraagteken met de methode **endswith().**

7. Print de tekst in kleine letters waarin alle spaties zijn vervangen door een underscore (_) met de methode **replace()**. Dit wordt ook wel **snake_case** genoemd.

Opdracht 1.7 – Levensfase

Print de levensfase op basis van een ingevoerde leeftijd,

Leeftijd	Levensfase
0 – 2	Baby
2 – 4	Kleuter
4 – 13	Kind
13 – 20	Tiener
20 – 65	Volwassene
65 or older	Oudere

Stappen:

1. Creëer een nieuw Python bestand.
2. Vraag de gebruiker om een leeftijd met de functie **input()**.
3. Sla de numerieke waarde op in een variabele. Gebruik de functie **int()**.
4. Gebruik een reeks **if** en **elif** opdrachten om de bijbehorende levensfase te printen. De bovengrenzen zijn niet geïncludeerd.

Opdracht 5.6 (page 123) uit het boek "Python Crash Course"

Opdracht 1.8 – Aantal klinkers

Tel het aantal klinkers in een stukje tekst.

Stappen:

1. Creëer een nieuw Python bestand.
2. Vraag de gebruiker om een stukje tekst met de functie **input().**
3. Sla dit op in een variabele **t**.
4. Loop in een **for-loop** door de klinkers ['a', 'e', 'i', 'o', 'u', 'y']
5. Print voor iedere klinker hoe vaak die in de tekst voorkomt. Gebruik hiervoor de **count()** methode.
6. Print daarna de totale lengte van de tekst met de len() functie.
7. Print tenslotte het totaal aantal klinkers

Output:

Klinker 'a' komt 58 keer voor
Klinker 'e' komt 97 keer voor
Klinker 'i' komt 66 keer voor
Klinker 'o' komt 39 keer voor
Klinker 'u' komt 23 keer voor

Klinker 'y' komt 8 keer voor
De tekst bevat 929 karakters
De tekst bevat 291 klinkers

Opdracht 1.9 – Hoger lager

Maak een programma waarmee je het hoger – lager spelletje kan spelen.

Output:
Raak een getal tussen 1 en 100
Wat denk je dat het getal is? 50
lager ...
Wat denk je dat het getal is? 25
lager...
Wat denk je dat het getal is? 12
hoger ...
Wat denk je dat het getal is? 19
hoger...
Wat denk je dat het getal is? 22
lager...
Wat denk je dat het getal is? 21
Jaaaa! Je hebt het goed in 6 pogingen

Stappen:

1. Creëer een nieuw Python bestand
2. Importeer het random bibliotheek. Bv. **import random**
3. Bepaal een geheim getal. Bv. **secret_number = random.randint(1, 100)**
4. Vraag in een while loop om de volgende gok van de gebruiker
5. Print als de gok onjuist is 'hoger' of 'lager'
6. Break als de gok correct is uit de loop en geef een feestelijke melding
7. Als uitbreiding kan in de loop worden bijgehouden hoeveel keer is gegokt.

Opdracht 2.1 – Print ingevoerde namen

Voer een aantal namen in. Als er geen naam wordt ingevoerd (return) ga dan verder met het tweede deel van de programma waarin de ingevoerde namen weer worden geprint. Bij voorkeur gesorteerd.

Stappen:

1. Start met een lege lijst **namen = []**
2. Gebruik een **while-loop** om telkens weer een naam in te voeren met **naam = input(…)**
3. Voeg de ingevoerde naam toe aan de lijst van namen met **namen.append(naam)**
4. Als er geen naam is ingevoerd stop de loop met een **break**

5. Gebruik een **for-loop** om door de lijst van namen te lopen en deze een voor een te printen.
6. Sorteer desnoods eerst de lijst net namen met **namen.sort()** of met **sorted(namen).**

Opdracht 2.2 – Tel hoe vaak woorden voorkomen

Tel hoe vaak woorden voorkomen in een stuk tekst.

Tips:

1. Kopieer een stukje tekst van het internet.
2. Creëer een Python bestand die de tekst in een variabele plaatst met **s = input()**
3. Schoon de tekst op door alle in kleine letters om te zetten en vervolgens door alle leestekens te verwijderen. Hiervoor zijn verschillende manieren denkbaar:
    a. **s.lower().replace('.', '').replace(',', '')**
    b. **s.lower().translate(str.maketrans('', '', '.,!?()[]'))**
    c. met met een regular expression **re.sub('[^a-z\s]', '', s.lower())**
4. Split de tekst in afzonderlijke woorden met **text.split()**
5. Creëer een set met unieke worden **unieke_woorden = set(woorden)**
6. Creëer een lege dictionary met **d = dict()** waarin de resultaten worden opgeslagen
7. Tel voor ieder unieke woord het aantal keer dat het voorkomt in de lijst van alle woorden met **n = woorden.count(woord)**
8. Sla het resultaat op in de dictionary **d[woord] = n**
9. Print the resultaten met **for woord, n in d.items()**

Opdracht 2.3 – Wachtwoord generator

Genereer een wachtwoord met tenminste 6 karakters met tenminste 1 hoofdletter, 1 kleine letter, 1 getal en 1 speciale karakter.

Tips:

1. Start met 4 strings met karakter families.
    - Bv. hoofdletters = 'ABCDEF..', kleine_letters = 'abcdef…', getallen = '0123456789' en speciale_karakters = '!@#$%&?/<>+=_-'
2. Gebruik de random bibliotheek om hieruit een willekeurige selectie te nemen.
    - import random
      part1 = **random.choices(hoofdletters, k=3)**
3. Voeg de verschillende delen samen = part1 + part2 + part3 + part4
4. Shuffle de volgorde van de karakters met **random.shuffle(karakters)**.
5. Zet de lijst van karakters om in een string met join(): password = **''.join(karakters)**
6. Print het gegenereerde wachtwoord.

Opdracht 2.4 – Spelkaarten

Selecteer 5 willekeurige kaarten van een deck speelkaarten.

Tips:

1. Defineer een lijst met de kleuren van een speelkaart:
    - **suits = ['clubs', 'diamonds', 'hearts', 'spades']**
2. Defineer een lijst met de rangen van een speelkaart:
    - **ranks = '2 3 4 5 6 7 8 9 10 J Q K A'.split()**
3. Combineer deze twee lijst tot één lijst met alle combinaties. Gebruik hiervoor een dubbele list comprehension:
    - **cards = [r + s for r in ranks for s in suits]**
4. Schud de kaarten met **random.shuffle(cards)**
5. Selecteer 5 kaarten met **cards.pop()**
    - **hand = [cards.pop() for _ in range(5)]**

Opdracht 2.5 – Banner

Creëer een functie die een tekst kan printen met sterretjes er om heen. Een soort banner.

```
***********
*  Peter  *
***********
```

Tips:

- Definieer een functie **banner**
- Definieer een argument **tekst**
- Bepaal de lengte van de tekst met **n = len(tekst)**
- Print de eerste regel met het juiste aantal sterretjes
- Print de tweede regel een sterretje een aantal spaties en de tekst
- Print de derde regel met allen sterretjes

Verbeteringen:

- Definieer een tweede argument voor de karakter die je wilt gebruiken voor de kader en geef die als default waarde een sterretje zodat het compatibel is met de oorspronkelijke functie.
- Geef een string terug met de banner in plaats van de banner te printen

Opdracht 2.6 – Range of floats

De range functie kan alleen een reeks van gehele getallen genereren. Creëer een functie die een reeks van floats kan genereren.

Tips:

- Definieer een functie **frange** met argumenten **start, stop** en **step**.
- Voeg eventueel ook een argument **endpoint** toe die aangeeft of de stopwaarde ook in de reeks moet worden opgenomen.

- Geef de argumenten default waarden. 1 voor de step en False voor de endpoint.
  Bv. **def frange(start, stop, step=1, endpoint=False)**
- Begin met een lege lijst van getallen. Bv. **numbers = []**
- Creëer een while loop die start bij het start argument en telkens het volgende getal berekent met een increment step en deze toevoegt aan de lijst van getallen.
  Bv**. number = start**, **number += step** en **numbers.append(number)**
- Als het getal groter of gelijk is aan het stop argument stop de loop met een break.
- Als het argument endpoint True is wordt de loop gestopt als het getal groter is dan het stop argument
- Vergeet niet de lijst van getallen terug te geven met **return**

Verbeteringen:

- In plaats van het werken met floats kan ook gebruik worden gebruikt van Decimal om de nauwkeurigheid te verbeteren.
  Bv. **from decimal import Decimal**
- De functie kan ook worden gerealiseerd als een generator. Gebruik hiervoor in plaats van een lijst van getallen in de loop de **yield** keyword.

Opdracht 2.7 – Sorteer een list

Soorteer de woorden in een stuk tekst.

Tips:

- Geef een variabele een stuk tekst. Bv. text = **'this is some random text'**
- Split de tekst in woorden. Bv. **words = text.split()**
- Gebruik de **sorted()** functie of de **sort()** methode om de woorden te sorteren.
- Print het resultaat
- Definieer een functie die het aantal klinkers telt in een woord.
  Bv met. **n = sum([word.lower().count(v) for v in 'aeiouy'])**
- Gebruik deze functie om de woorden te sorteren op basis van het aantal klinkers
  Bv. **sorted(words, key=number_of_vowels)**

Opdracht 2.8 – Schrijven naar en lezen van een bestand

Tekst naar een bestand wegschrijven en vervolgens de tekst weer lezen uit het bestand.

Tips:

- Creëer een nieuw Python bestand. Bv. **writing_to_a_file.py**
- Bepaal een naam voor het bestand. Bv. **filename = 'demo.txt'**
- Open de file in write mode. Gebruik hiervoor de context manager keyword with.
  Bv. **with open(filename, 'w') as f:**
- Schrijf een aantal regels weg naar de file met de methode write. Vergeet niet zelf de newline karakters toe te voegen.
  Bv. **f.write('Line 1\n')**
- Creëer een tweede Python bestand. Bv. **reading_from_a_file.py**

- Bepaal dezelfde naam voor het bestand. Bv. **filename = 'demo.txt'**
- Open de file in lees mode. Gebruik hiervoor weer de context manager keyword with.
  Bv. **with open(filename) as f:**
- Lees alle regels in het bestand in een for loop.
  Bv. **for line in f:**
- Print iedere regel. Voorkom dat er steeds een extra lege regel wordt geprint.
  Bv. **print(line.rstrip())**

Opdracht 2.9 – Lees een CSV bestand

Lees een CSV bestand en filter regels eruit.

Tips:

- Plaats het CSV bestand ca-500.csv in dezelfde directory als het Python programma.
- Creëer een nieuw Python bestand.
- Open het bestand met een context manager met de keyword with.
  Bv. **with open(filename) as f:**
- Lees de eerste regel. Dit is de header.
- Loop vervolgens door de rest van de regels.
- Strip van iedere regel het laatste karakter met de methode **strip()**
- Split de regel in een lijst van waarden met de methode **split()**
- Selecteer alleen de regels met **city** 'Montreal'
- Print van deze regels de kolommen **firstname**, **lastname**, **city** en **email**

Opdracht 2.10 – Eigen foutmelding

Probeer een niet bestaand bestand te openen en geef in een aangepaste gebruikersvriendelijke foutmelding terug dat dat het bestand niet bestaat.

Tips:

- Definieer een variabele **filename** met een fictieve bestandsnaam.
  Bv. **filename = 'a_file_that_does_not_exist.txt'**
- Voeg een **try** statement toe.
- Open het bestand zoals gebruikelijk met een context manager.
  Bv. **with open(filename) as f:**
- Lees vervolgens het bestand en print de inhoud.
- Voeg een except statement toe voor een **FileNotFoundError** exception.
  Bv. **except FileNotFoundError:**
- Print binnen het except blok een eigen foutmelding.

Opdracht 2.11 – Foolproof numerieke invoer

Creëer een functie die de gebruik vraagt om een getal in te voeren tussen getallen dia als argumenten zijn meegegeven. De functie dient foutieve invoer netjes af te handelen. Zowel

als het ingevoerde getal buiten de aangegeven grenzen valt als dat er een ongeldige type is ingevoerd.

Tips:

- Definieer een functie **numeric_input()** met argumenten **lower** en **upper**.
- Vraag in een while loop om invoer van de gebruiker.
  Bv. **response = input(f'Geef een getal tussen {lower} en {upper}: ')**
- Converteer het ingevoerde tekst naar een getal.
  Bv. **number = int(response)**
- Als er hierbij een fout op treedt, vang deze af en geef een toepasselijke foutmelding.
- Controleer of het getal binnen de opgegeven grenzen valt.
- Geeft een toepasselijke melding als dat niet het geval is.
- Break uit de loop als een goed getal is ingevoerd en geef deze terug met **return**.
- Test de functie