0 Werken met Jupyter Notebook en PyCharm

0.1 Jupyter Notebook

In de Jupyter Notebook vind je een korte herhaling van de belangrijkste concepten uit het hoorcollege waarop in deze oefenzitting op geoefend zal worden. De Jupyter Notebooks hebben als extensie `.ipnb`.

De Notebook horende bij elke oefenzitting kan gedownload worden van Toledo. Voor het openen, start je eerst het programma Juypter Notebook, en navigeer je vervolgens naar de bestand.

0.2 PyCharm

Naast de introductie oefeningen in de Notebook, in het `.pdf` bestand volgen wat meer uitgebreide oefeningen. Deze worden opgelost in een PyCharm omgeving. Voor elke oefenzitting kan je een nieuw project aanmaken, waarbij elke oefening in een apart bestand wordt gemaakt.

Bij het aanmaken van een nieuw project, genereerd PyCharm drie folders: `.idea`, `__pycache__` en `venv`. Deze laat je best ongemoeid, ze bevatten de informatie met welke configuratie het project moet uitgevoerd worden. Zo kan je afhankelijk van je project definieren welke versie van python je wil gebruiken. In de cursus *Beginsellen van Programmeren* zullen we gebruik maken van Python 3.9. Dit is de versie die bevat zit in de laatste Anaconda distributie.

Indien je werkt op een computer van de KULAK, sla je de projecten op op de netwerkschijf horende bij jou studentennummer. Zo kan je de projecten op elke computer van de KULAK bereiken. In tegenstelling tot als je de projecten opslaat op de `C:\` schijf, zijn de projecten enkel bereikbaar op die bepaalde computer.

Enkele dagen na de oefenzittingen, worden voorbeeld oplossingen beschikbaar gemaakt op Toledo.

1 Interpreteren van code

Deze oefeningen reeks toetst jullie vermogen om code te lezen en begrijpen. Gebruik gerust de witruimte naast de code als klad om de waardes van variabelen, tijdens de verschillende stappen van het programma, bij te houden.

Oefening 1-1: Wat is de waarde van de variabele `mysterie` na het uitvoeren de volgende blok code? Vorm hierover eerst een idee en controleer daarna pas je antwoord door de code uit te voeren!

```
1    mysterie = 1
2    mysterie = mysterie + 1
3    mysterie = 1 + 2 * mysterie
4    print(mysterie)
```

Oefening 1-2: Wat is de waarde elke variabele na het uitvoeren van onderstaande code?

```
1    n = 1
2    k = 2
3    r = k
4    if r < k:
5       n = r
6    else:
7       k = n
8    print(n,k,r)
```

Oefening 1-3: Een oefening op integer deling en modulo, kan je de uitkomst van volgende print statements geven?

```
1    n = 124
2    k = 30
3    l = 12
4    print(n // k)
5    print(k % l)
6    print(l // n)
7    print(n // l)
```

2 Probleem oplossend denken

Oefening 2-1: Voor het inplannen van afspraken in onze kalender, willen we een programma schrijven die verifieerd of er geen overlap bestaat tussen twee afspraken. Hiervoor vraag het programma een strart tijdstip en de duurtijd voor beide afspraken. Geef een antwoord aan de gebruiker indien de afspraken al dan niet overlappen.

Tip: maak een eerste implementatie waarbij de tijden als rationale getallen worden ingegeven (bijvoorbeeld 12.5 stelt 12u 30min voor). Een tweede meer geavanceerde implementatie zou de tijdstippen kunnen bijhouden als twee variabelen (een voor de uren en een voor de minuten). Bij deze tweede implementatie kan je gebruik maken van de integer deling // en de modulo % operator, om met uren en minuten te kunnen rekenen.

Oefening 2-2: Ons huis heeft zonnepanelen gecombineerd met een thuisbatterij. Doordat de terugdraaiende teller is afgeschafd, is het belangrijk om het verbruik af te stellen op de productie van de zonnepanelen. Daardoor willen we een programma maken dat controleert of we onze vaatwas mogen starten of hiermee best nog wachten. Onze vaatwas verbruikt 0.75kWh per wasbeurt.

Schrijf een programma dat vraagt aan de gebruiker wat de huidige productie van de zonnepanelen is, het huidig verbruik van de rest van het huis en het oplaatniveau van zijn thuisbatterij.

- Indien de batterij meer dan 90% is opgeladen, mag het toestel gestart worden.
- Indien de productie meer dan 80% van de noodzakelijke stroom kan leveren, mag het toestel gestart worden.
- Indien de batterij meer dan 65% is opgeladen en de productie kan voor 65% van de noodzakelijke stroom instaan, mag het toestel gestart worden.
- In alle andere gevallen moet een waarschuwing geprint worden dat de productie en/of het batterij niveau te laag is, met de vraag aan de gebruiker of het toestel effectief mag gestart worden.

We nemen aan dat het huidige verbruik en de huidige productie constant zijn over de duurtijd van de wasbeurt.

Oefening 2-3: Schrijf een programma dat cartesische coördinaten (x, y) omzet naar pool-coördinaten (r, θ). De formules hiervoor zijn:

$$r = \sqrt{x^2 + y^2}$$

$$\theta = \arctan \frac{y}{x}$$

TIP: Je zal functies uit de voorgedefinieerde `math` *module nodig hebben. Voor* arctan *dien je* `atan` *te gebruiken.*

Oefening 2-4: We willen onze conditie terug opbouwen. Om oververmoeidheid tegen te gaan, maken we een programma zodat we kunnen vragen, gegeven onze laatste oefensessie, of we vandaag terug mogen gaan sporten.

Volgens onze trainer moet er telkens twee dagen tussen onze inspanningnen zitten. Hierop zijn er twee uitzonderingen. Ten eerste indien de vorige oefensessie korter was dan 30 min, mogen we de dag erna opnieuw gaan sporten. Ten tweede als de gemiddelde hartslag hoger was dan 180 slagen/min van een oefensessie die langer dan 30 min duurde, voeren we best nog een extra dag rust toe aan ons oefenschema.

3 UOVT

Oefening 3-1: Schrijf een programma dat een tic-tac-toe grid afdrukt. Vermijd de herhaling van regels die rechtstreeks strings afdrukken (zoals print("+–+–+–+")) maar gebruik variabelen om de strings in bij te houden.

```
+-+-+-+
| | | |
+-+-+-+
| | | |
+-+-+-+
| | | |
+-+-+-+
```

Oefening 3-2: Schrijf een programma dat de afmetingen van het standaard Amerikaans briefformaat (8.5 op 11 inch) in millimeter afdrukt. 1 inch is 25.4 millimeter. Gebruik constanten en voorzie commentaar in je code.

Oefening 3-3: Schrijf een programma dat de gebruiker om een string vraagt en deze als volgt opnieuw afdrukt: Eerst de eerste 3 karakters, dan 3 puntjes, dan de laatste 3 karakters.

TIP: Gebruik de slice operator: string[start:eind] geeft de substring terug vanaf index start tot2 index eind. Elk van de argumenten kan ook weggelaten worden, dan worden respectievelijk de waarden 0 of len(string) ingevuld.

```
Geef een lange string in: Mississippi
Mis...ppi
```

Oefening 3-4: Schrijf een programma dat een float inleest en afdrukt dat dit getal nul is, indien dat effectief zo is. Anders schrijft het programma positief of negatief uit. Voeg verder het woord klein toe indien de absolute waarde van het getal kleiner is dan 1, of groot indien de absolute waarde groter is dan 1 000 000.

```
Geef een float in: 0.33
Dit getal is een klein positief getal.
Geef een float in: 12.75
Dit getal is een positief getal.
Geef een float in: -1234567
Dit getal is een groot negatief getal.
Geef een float in: 0
Dit getal is nul.
```