**Opzet Python-cursus 25-11-2019**

Welkom

Controle apparatuur

* Welke OS-en;
* Welke Python versies;
* Virtual environments per OS nagaan;
* IDE’s?

Herhaling

* Cheatsheet
* REPL
* Taal elementen:
  + Inspringen
  + Variablelen
    - Naamgeving (geen camel-case)
    - letters, nummers (niet aan het begin), underscore
  + If / Elif
  + While
    - * n = 10
      * while n > 0:
      * ... print(f"n={n}")
      * ... n -= 1
    - Of
      * naam = ""
      * while naam != "Eric":
      * ... naam = input("Wat is je naam: ")
  + For
    - * range(5)
      * for i in range(5), print(i)
      * for i in range(5):

print('Jimmy Five Times (' + **str(i)** + ')')

* + - * n = “Dit is een tekst”
      * len(n)
      * for i in range(len(n)), etc
    - Beter:
      * for i in n:
        + print(n)
        + String is een iterable (Iterable is an object, which one can iterate over.)
  + Open
    - f = open("sof-analyze.txt","r")
    - f.read()
    - f.seek(0)
    - f.readline()
    - f.seek(0)
    - f.readlines()
  + Input
  + With (context manager)
  + Functies (def)
    - def zonder parameter
    - def met parameter
    - def met return
  + Imports (packages)
    - voorbeelden in REPL + dir(), help():
      * import platform
      * import datetime
      * import random
        + import random
        + for **\_** in range(5):

print(random.randint(1, 10))

def print\_een\_naam():

print("Ik heet Eric.")

if \_\_name\_\_ == "\_\_main\_\_":

print\_een\_naam()

* + Oefeningen
    - Schrijf een kort programma welke op grond van de willekeurig gekozen getallen 1, 2 en 3 een specifieke tekst print.
    - Collatz
      * Start met een getal > 1
      * Getal even:
    - import random
    - for \_ in range(3):
    - r = random.randint(1, 3)
    - if r == 1:
    - print(f"Er zat {1} aapje op een stokje...")
    - elif r == 2:
    - print(f"{2} Emmertjes water halen...")
    - elif r == 3:
    - print(f"{r} Kleine kleutertjes die zaten op een hek…")
  + Data-typen
    - Boolean
      * t = True
      * while t:
      * ... naam = input("Wat is je naam? ")
      * ... t = False if naam == "Eric" else True
    - String
      * slicing
        + string[start: end: step] Voorbeelden:

s[2:5] karakters 2 t/m 4

s[-1] Laatste karakter

s[::2] Ieder 2e karakter

s[::-1] String van rechts naar links

* + - Set
      * Een set is unordered en unindexed. Een element komt maar 1 keer voor. Set-methodes.
      * fruitschaal = {"appel", "banaan", "kers"}
      * Geen index dus niet fruitschaal[1] maar alleen:
      * for fruit in fruitschaal:
        + print(fruit)
      * Wel veel gebruikt: in of not in:
      * if “appel” in fruitschaal:
    - Tuple
      * Een tuple is ordered en unchangeable/immutable.
    - List
    - Dictionary
  + Comprehensions
    - List comprehension
      * **letter\_list** = []
      * **for letter in “Hello World!”**:
        + letter\_list.append(**letter**)
      * Wordt:
        + **letter\_list** = [**letter** **for letter in “Hello World!”**]
      * **cijfer\_list** = []
      * **for cijfer in range(20)**:
        + **if cijfer % 2 == 0**:

cijfer\_list.append(**cijfer**)

* + - * Wordt
        + **cijfer\_list** = [**cijfer** **for cijfer in range(20)** **if cijfer % 2 == 0**]
      * Complexer…
      * non\_flat = [ [1,2,3], [4,5,6], [7,8] ]
      * for x in non\_flat:
        + for y in x:

y

* + - * Wordt:
      * flat = [y for x in non\_flat for y in x]
      * Inlezen bestand:
      * f = open("sof-analyze.txt","r")
      * l = [x.lstrip("\t").rstrip("\n") for x in f.readlines()]
    - Dict comprehension
  + Oefeningen
    - cijfers = [1, 2, 3, 4, 5]
    - kwadraten = [?]
    - list\_a = [1, 2, 3 ,4]
    - list\_b = [2, 3, 4, 5]
    - gelijke\_nrs = [?]
* Virtual environments
  + pip installs
  + pip list
  + pip freeze (> requirements.txt)

Pause?

Hoofd gedeelte

Analyzeren van de 2019 Stack Overflow Developer Survey

Vraag: “Welke computertalen zijn het meest populair per Developer type”. Of: welke computertalen worden veel gebruikt wanneer embedded computers (micro controllers) worden gebruikt?

Website:

<https://insights.stackoverflow.com/survey/2019?utm_source=so-owned&utm_medium=blog&utm_campaign=dev-survey-2019&utm_content=launch-blog>

Data:

<https://insights.stackoverflow.com/survey>

Opzet

* Aanmaken virtual environment
  + mkdir cursus-2019-11
  + cd cursus-2019-11
  + python -m venv venv
  + Bekijken venv/ directory
  + source venv/bin/activate
  + which python3
  + which pip
* Start de IDE
  + code .
  + Korte uitleg van Visual Studio Code
    - Denk ook aan python/lint extensions

* Downloaden CSV handmatig of via Python? Toch even leuk om een data download te doen via Requests.
  + Zie source 01-csv-halen.py
* ZIP uitpakken en plaatsen in Data directory
  + Zie source 02-csv-zip-uitpakken.py
* Data bekijken, bv Readme.txt, data en schema.
* Maak een sof-analyze.py
* Daarna:
  + import CSV
  + Open het bestand:
    - with open('data/survey\_results\_public.csv') as f:

csv\_reader = csv.DictReader(f)

* + Print daarna de eerste regel:
    - for line in csv\_reader:
    - print(line)
    - break
  + Laat zien dat wat je ziet een dictionary is (van tuples)
  + Laat zien dat je een alleen veld kunt printen door een key aan te geven:
    - print(line[“Hobbyist”])
  + Verwijder eventueel het break-statement.
  + Percentage hobby-programmeurs? 3 Manieren:
    - Standaard manier
      * Variabelen voor ‘yes’ en ‘no’ plus increment ervan
      * Ná het with-statement:
        + plaats yes\_counter en no\_counter
        + Increment counters als line[“Hobbyist”] == “Yes” / “No”
        + Maak een total counter en yes\_pct (yes\_counter / total \* 100)
        + Voer een round door
        + print resultaten
        + Laat line.get(“Hobbyist) zien + default waarden.
    - Gebruik van een dictionary (histogram?)
      * Verwijder yes- en no\_counters
      * Maak een counter dict:
        + counter = {“Yes”: 0, “No: 0}
      * Het If/Elif commando kan nu vervangen worden door:
        + counter[line[“Hobbyist”]] += 1
        + Wijzig nu ook alle andere variabelen en de print statements
      * Leg nu het gebruik van defaultdict uit (in REPL).
        + From collections import defaultdict
        + Wijzig de counter in: counter = defaultdict(int)
    - Via Counter-object in Collections
      * From collections import Counter
      * Leg Counter uit, bijvoorbeeld in de REPL
      * Wijzig de counter in: counter = Count()
  + Nu over naar: Wat zijn de meest populaire computertalen?
    - Open de file schema.csv en ga na in welke kolom de computertalen staan: LanguageWorkedWith (AQ)
      * Wijzig de counter in ‘language\_counter’
      * print de eerste regel met ‘language\_counter’
      * Laat het type zien van ‘language\_counter’ (str)
      * Laat eventueel de eerste 10 regels zien mbv enumerate en If i < 11
    - Een regel met languages bevat meerdere computertalen: gescheiden met ‘;’
      * Lees een LanguageWorkedWith regel in en split deze op ‘;’ (languages\_list)
      * Print het resultaat
      * Doe het eventueel voor in de REPL
    - Parse de language\_list met een for-loop
      * for language in language\_list:
    - Tel de talen: language\_counter[language] += 1
    - print de language\_counter
    - De for-loop kan vervangen worden door: language\_counter.update(languages\_list)
    - Doe het voor in de REPL
    - vervang de loop en print de language\_counter
    - Laat deze eventueel over de eerste 10 regels zien.
    - Verwijder nu de regel restrictie en verwerk alle regels.
    - Wijzig het print statement in language\_counter.most\_common(5) om de 5 meest gebruikte computertalen te laten zien.
    - Laat zien dat de output van most\_common een list met tupels is.
    - Maak een total counter en tel het aantal regels.
    - Ga nu met een for-loop alle talen en hoe vaak deze worden gebruikt na:
      * for language, value in language\_counter.most\_common(5):
        + language\_pct = round((value / total) \* 100, 2)
        + print(f"{language.rjust(10)}: {language\_pct}%")
  + Nu naar de vraag: “Talen per Developer Type”
    - We gaan een dictionairy maken zoals:

{“<devtype>” : {

“total”: 0,

“language\_counter”:

{“<language>: “counter”: 0”}

}

}

* + - Laat zien dat in de data het veld DevType alle developertypes bevat. Gescheiden door ‘;’
    - Verwijder total en language\_counter. Maak een dev\_type\_info als een lege dictionary: dev\_type\_info = {}.

Verwijder verder alles behalve de for-loop over de lines.

* + - Dan, voor iedere line:
      * dev\_types = line[“DevType”].split(“;”)
    - Ga nu bij alle dev\_types langs en maak een lege dict voor iedere dev\_type key:
      * for dev\_type in dev\_types:
        + dev\_type\_info[dev\_type] = {}
        + dit levert: {'Designer': {}, 'Developer, back-end': {}, 'Developer, front-end': {}, 'Developer, full-stack': {}}
        + Laat dit in de REPL zien.
    - Print de verschillende dev\_type keys:
      * for key in dev\_types\_info:
        + print(key)
    - Nu de vraag: hoe zet ik de default waarden van de dev\_type\_info[<dev\_key>]?
      * Setdefault(). Setdefault maakt de opgegeven waarden in de dict als deze nog niet bestaan. Anders worden ze ongemoeid gelaten. Laat in REPL zien.
      * verwijder de regel:
        + dev\_type\_info\_dict[dev\_type] = {}
      * Voeg toe:
        + dev\_type\_info\_dict.setdefault(dev\_type, {
        + "Total": 0,
        + "language\_counter": Counter()
        + })
      * Daarna, verhoog aantal dev\_types:
        + dev\_type\_info\_dict[dev\_type]["Total"] += 1
      * Dan, bepaal languages (per dev\_type) en tel languages
        + dev\_languages\_list = line.get("LanguageWorkedWith").split(";")
        + dev\_type\_info\_dict[dev\_type["language\_counter"].update(dev\_languages\_list)
      * Print tenslotte de resultaten:
        + for dev\_type, info in dev\_type\_info\_dict.items():

total = info["Total"]

print(f"{dev\_type} ({total})")

for language, counter in info["language\_counter"].most\_common(5):

print(f"\t{language} – {(counter/total)\*100:.1f}")

print()

Klaar!