Opdracht 1: een studiejaar plannen

1 Introductie

In deze opdracht ontwerp, implementeer en test je de kernfunctionaliteit van een applicatie waarmee een studieperiode ingepland kan worden. Met kernfunctionaliteit bedoelen we dat er geen user interface wordt ontwikkeld. De resultaten van het inplannen worden in de console afgedrukt.

In deze opgave plan je één jaar van de propedeuse in. De propedeuse duurt nominaal 2 jaar. Je mag er van uitgaan dat je per kwartiel steeds één cursus volgt. De opgave is dus om voor elk kwartiel een geschikte cursus te vinden.

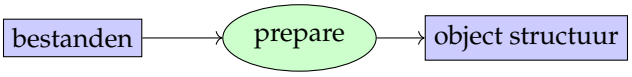
Je kunt niet elke willekeurige cursus kiezen. Aan cursussen zijn bijvoorbeeld ingangseisen verbonden. De meeste cursussen worden alleen in een bepaalde periode gegeven. De applicatie die je opstelt houdt bij de keuze van die vier cursussen rekening met dat soort voorwaarden.

Voor het ontwerp en implementatie maak je gebruik van OO- en functionele aspecten van Python. De klassen en bijbehorende losse functies worden verdeeld over vier modulen. Deze vier modulen krijg je van ons aangeleverd met delen van de code. In stappen begeleiden we je langs de ontwikkeling van de applicatie.

2 Het hoofdproces

Het hoofdproces bestaat uit twee stappen:

a Voorbereiden: het inlezen van alle benodigde gegevens en het vastleggen ervan in een structuur van objecten.

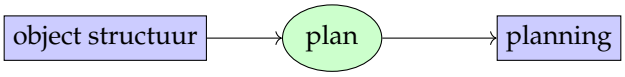


Hierbij worden twee bestanden ingelezen:

(a) een bestand met de cursuscodes van de cursussen die al gevolgd zijn (cursussengedaan.json)

(b) een bestand met de gegevens van elke propedeuse cursus (cursusaanbod.json).

b Plan: het plannen van het komende jaar.



De uitkomst is een planning voor het komend jaar: 4 kwartielen (te beginnen in kwartiel 1), met 1 vak per kwartiel.

3 Eisen

De eisen aan de te kiezen cursussen zijn als volgt:

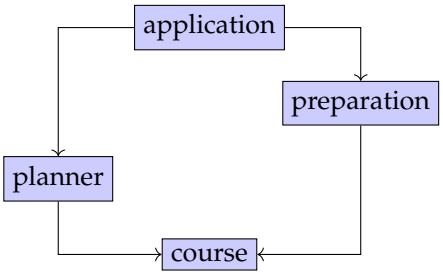
– Er moet voldaan zijn aan de *vereiste* voorkennis van de gekozen cursus.

– De keuze wordt gemaakt op basis van de volgende prioriteiten: de hoogste prioriteit staat bovenaan.
  a Een vaste cursus heeft voorrang op een variabele cursus.
  b Een cursus waarvoor je de *gewenste* voorkennis hebt, heeft voorrang boven een cursus met gewenste voorkennis die je nog niet hebt.
  c Een vaste cursus die onderdeel is van de *vereiste* voorkennis van *toekomstige* cursussen heeft voorrang boven andere vaste cursussen.
  d Een vaste cursus die onderdeel is van de *gewenste* voorkennis van *toekomstige* cursussen heeft daarna voorrang.
  e Een variabele cursus met een tentamen aan het eind van dit kwartiel heeft voorrang op een variabele cursus zonder zo'n geschikt tentamenmoment.

4 De modulen en de klassen

4.1 DE MODULEN

Je krijgt vier modulen bij de uitgangscode. Je mag meer modulen aanmaken of modulen opsplitsen, maar nodig is dat niet.
– De module *application* stuurt de preparation en de planning aan.
– De module *preparation* bevat een klasse Preparation die de invoerbestanden kan inlezen en een gegevensstructuur kan opstellen.
– De module *planner* bevat een klasse Planner die de cursussen kan uitkiezen.
– De module *course* bevat de klasse Course, voor de objectenstructur, een klasse Start_and_enddate die door de klasse Course wordt gebruikt, en functies die voor het filteren van lijsten van cursussen.

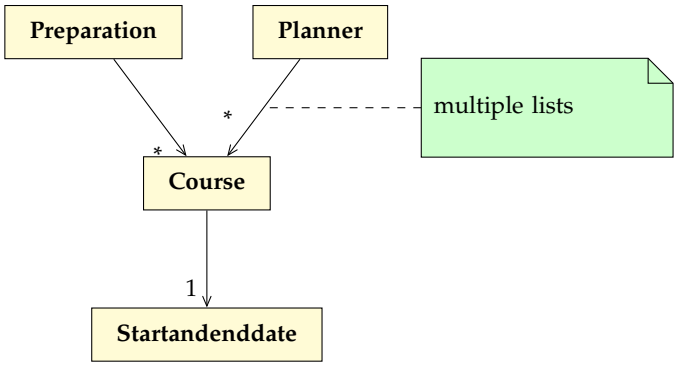


FIGUUR 1.1 Afhankelijkheden van de modulen

Figuur 1.1 laat zien hoe de modulen afhankelijk zijn van elkaar. De module *application* is afhankelijk van zowel *preparation* als *planner*. De modulen *preparation* en *planner* zijn beide afhankelijk van de module *course*.

Klassendiagram
Figuur 1.2 laat zien welke klassen in de applicatie worden gebruikt.

Een object van de klasse Preparation (in de module *preparation*) beschikt over een lijst met objecten van de klasse Course.



FIGUUR 1.2 De klassen

Een object van de klasse Planner (in de module *preparation*) beschikt over een aantal lijsten met objecten van de klasse Course.

Een object van de klasse Course beschikt over één object van klasse Start_and_enddate. Beide klassen bevinden zich in de module *course*.

4.2 DE MODULE APPLICATION

De module application bevat een functie en een opdrachtregel die alles in gang zet:

```
import planner
import preparation


def generate_planning():
  prep = preparation.Preparation()
  prep.load_courses_done()
  prep.load_courses_offer()
  my_planner = planner.Planner(prep)
  print(my_planner.generate())


generate_planning()
```

Er wordt een object van de klasse Preparation gecreëerd. Dat object krijgt de opdracht de twee bestanden in te lezen. Vervolgens wordt er een object van de klasse Planner gecreëerd dat het Preparation object als parameter meekrijgt. Tenslotte krijgt het object van de klasse Planner de opdracht om een planning te genereren.

4.3 DE MODULE PREPARATION

In de module *preparation* zit één klasse: de klasse Preparation.

```
import pathlib
import json
import course


class Preparation:
  COURSES_DONE_FILE = 'curssengedaan.json'
  COURSES_OFFER_FILE = 'cursusaanbod.json'
  available_courses = []
```

```
def __init__(self, path=None):
  if path:
    self.path = path
  else:
    self.path = pathlib.Path.cwd()
  self.done_codes = set()

def load_courses_done(self):
  path = self.path / pathlib.Path(self.COURSES_DONE_FILE)
  file_path = pathlib.Path(path)
  with file_path.open('r') as f:
  self.done_codes = set(json.load(f))

def load_courses_offer(self):
  path = self.path / pathlib.Path(self.COURSES_OFFER_FILE)
  file_path = pathlib.Path(path)
  with file_path.open('r') as f:
    for info in json.load(f):
      self.available_courses.append(course.create_course(info))
```

De klasse bevat methoden om de invoerbestanden in te lezen, en bouwt met behulp daarvan een lijst op met objecten van de klasse Course: available_courses.

4.4 DE MODULE PLANNER

In de module planner zit een klasse Planner.

4.4.1 *Klasse Planner*

De klasse Planner implementeer je zelf. Wat je meekrijgt zijn een aantal methodenamen. Denk er aan dat je meer methoden nodig hebt om duidelijke, onderhoudbare, leesbare code te krijgen.

```
import course


class Planner:
  # TODO: implement and add attributes

  def __init__(self, application):
    # TODO: implement

  def compute_current_state(self):
    # TODO: implement

  def choose_course(self, quartile):
    """
    :param quartile: int that shows the quartile
    """
    # TODO: implement

  def generate_for_quartile(self, quartile):
      """
    :param quartile: int that shows the quartile
    :return: string for this quartile
    """
    # TODO: implement

  def generate(self):
    """
```

```
        :return: string showing the planning
        """
        self.generate_for_quartile(1)
```

De enige methode die van buiten de module gebruikt zal worden (die je dus als public kunt opvatten) is *generate*. De overige methoden, inclusief de methoden die je zelf zult toevoegen, zijn hulpmethoden (die kun je dus als private opvatten).

We bespreken de bovenstaande methoden hier.

compute_current_state
Deze methode berekent wat de huidige stand van zaken is. Dat betreft:
- De cursussen uit het aanbod die nog niet zijn gedaan.
- De cursuscodes die behoren tot de *vereiste* voorkennis van de cursussen die nog niet zijn gedaan.
- De cursuscodes die behoren tot de *gewenste* voorkennis van de cursussen die nog niet zijn gedaan.
- De cursussen uit het aanbod die nog niet zijn gedaan en die op basis van de huidige voorkennis, gedaan zouden kunnen worden.
- De *vaste* cursussen uit het aanbod die nog niet zijn gedaan en die op basis van de huidige voorkennis, gedaan kunnen worden.
- De *variabele* cursussen uit het aanbod die nog niet zijn gedaan en die, op basis van de huidige voorkennis, gedaan kunnen worden.

N.B. Een vaste cursus is een cursus met een startdatum; een variabele cursus is een cursus zonder startdatum.

Denk er aan dat je aan deze methode kunt zien dat je een aantal attributen nodig hebt in de klasse Planner. Bedenk duidelijke namen voor die attributen.

choose_course
Deze methode heeft één parameter: het kwartiel waar een cursus voor wordt gekozen, in de vorm van een int. De methode doet het volgende:
- De methode berekent welke van de vaste cursussen die gedaan kunnen worden in dit kwartiel worden gegeven. Het is niet nodig om de lijst met die cursussen als attribuut op te nemen; de lijst kan een tijdelijke variabele binnen deze methoden worden.
- Als de lijst met mogelijke vaste cursussen in dit kwartiel niet leeg is, kies dan een geschikte vaste cursus uit. Maak een attribuut voor de gekozen cursus.
- Als de lijst met mogelijke vaste cursussen in dit kwartiel leeg is, kies dan een geschikte variabele cursus uit. Gebruik het attribuut voor de gekozen cursus.

Een eventuele geschikte cursus wordt gebruikt als waarde van een attribuut (bedenk een goede naam). Wanneer er geen cursus gevonden is, geef dat attribuut dan als waarde None.

generate_for_quartile
Deze methode:
- maakt gebruik van de methode compute_current_state om de huidige toestand te berekenen,

- voegt de gegevens van het kwartiel en de voorkennis op dat moment toe aan de terug te geven string,
- gebruikt de methode choose_course om een cursus uit te kiezen,
- als er een cursus gekozen kon worden,
  - voegt de gegevens toe aan de terug te geven string,
  - voegt de cursuscode van de gekozen cursus toe aan de lijst van cursuscode van cursussen die je hebt gevolgd,

- anders:
  - voegt aan de terug te geven string een mededeling toe dat er geen geschikte cursus in dat kwartiel is,

- roept, als dit nog niet kwartiel vier was, generate_for_quartiel aan voor het volgende kwartiel, en voegt de string die wordt teruggegeven toe aan de terug te geven string,
- geeft de string terug.

generate
De implementatie van deze methode is simpel; je krijgt hem mee:

```
def generate(self):
    """
    :return: string met planning
    """
    return self.generate_for(1)
```

4.4.2 *Aanwijzingen*

Let bij het implementeren op het volgende:
- In de beschrijvingen kun je zien welke attributen je nodig hebt in deze klasse. Bedenk duidelijke namen voor deze attributen.
- Je ziet dat de beschreven methoden in de meeste gevallen veel verantwoordelijkheden hebben. Dat houdt in dat je hulpmethoden moet schrijven die elk van die verantwoordelijkheden uitvoeren.
- Wanneer je gegevens nodig hebt van een cursus (in welk kwartiel een cursus begint, of een cursus vast of variabel is bijvoorbeeld), zorg er dan voor dat je daar een methode voor schrijft in de klasse Course.
- Je hebt op meerdere plaatsen een *nieuwe* lijst nodig, op basis van een lijst waar je bepaalde cursussen uit filtert. Gebruik daar list comprehension voor. Het mooist is het om daar functies voor te schrijven die je in de module *course* onderbrengt, met duidelijke namen.

4.5 DE MODULE COURSE

De module *course* bevat twee klassen, Start_and_enddate en Course. Daarnaast zul je in deze modulen ook een aantal lossen functies onderbrengen, die je zelf schrijft.

4.5.1 *De klasse Course*

Deze klasse implementeer je vrijwel geheel zelf. je krijgt het volgende mee:

```
class Course:
  # TODO: implement and extend with attributes and methods

  def __init__(self, code, title, startdate=None, enddate=None):
    # TODO: implement

  def __str__(self):
    """ string with:
    - code,
    - title,
    - period,
    - new line
    - codes of required foreknowledge or 'geen verplichte voorkennis'
    - new line
    - codes of desired foreknowledge or 'geen gewenste voorkennis'
    - new line
    """
    # TODO: implement
```

4.5.2 *Start_and_enddate*

De klasse Start_and_enddate krijg je mee. Het is een klasse die het gemakkelijk maakt om met de start- en einddatum van een cursus te werken.

Bij de init-methode van Course creëer je een Start_and_enddate-object met de startdatum en einddatum die bij de cursus horen. Je kunt dat doen op de volgende manier:

```
self.dates = Start_and_enddate(startdate, enddate)
```

De data staan in de bestanden met de gegevens die je meekrijgt in het juiste format.

De code van de klasse *Start_and_enddate* is als volgt:

```
class Start_and_enddate:
  """ Class for objects with a startdate and an enddate """
  def __init__(self, startdate=None, enddate=None):
    self.startdate = None
    self.enddate = None
    if startdate:
      self.startdate = datetime.date.fromisoformat(startdate)
    if enddate:
      self.enddate = datetime.date.fromisoformat(enddate)

  def __str__(self):
    """ String representation
    no startdate and enddate: returns variable
    otherwise: returns startdate, enddate
    """
    if self.nodates():
    return 'variable'
    return str(self.startdate) + ', ' + str(self.enddate)

  def nodates(self):
```

```
    """ true when there is no startdate and no enddate """
    return (not self.startdate) and (not self.enddate)

  def quartile(self):
    """ Compute quartile
    startdate in month 9: return 1
    startdate in month 11: return 2
    startdate in month 2: return 3
    startdate in month 4: return 4
    otherwise: return -1
    """
    if self.startdate.month == 9:
      quartile = 1
    elif self.startdate.month == 11:
      quartile = 2
    elif self.startdate.month == 2:
      quartile = 3
    elif self.startdate.month == 4:
      quartile = 4
    else:
      quartile = -1
    return quartile
```

4.5.3 *Course*

Deze klasse ga je geheel zelf implementeren. De klasse beschrijft een cursus zoals die wordt aangeboden.

Probeer er aan te denken dat je hulpmethoden schrijft zodra je merkt dat de code van een methode lang wordt, of lastig leesbaar.

4.6 DE FUNCTIE CREATE_COURSE

Deze functie krijg je geïmplementeerd en al mee. Om de functie werkend te krijgen, moet je er voor zorgen dat de __init__ van de klasse Course aansluit op wat wordt verwacht in deze functie. Dat betekent ook dat je in deze functie aanwijzingen vindt voor attributen waar de klasse Course over moet beschikken.

```
def create_course(courseinfo):
  newcourse = Course(courseinfo['code'], courseinfo['naam'],
        courseinfo['sbu'], courseinfo['startdatum'],
        courseinfo['einddatum'])
  newcourse.add_required_courses(courseinfo['voorkennisverplicht'])
  newcourse.add_desired_courses(courseinfo['voorkennisgewenst'])
  newcourse.add_exams(courseinfo['tentamens'])
  return newcourse
```

4.6.1 *Functies voor lijsten met cursussen*

Je hebt bij de methoden van Planner al gezien dat je vaak een nieuwe lijst met cursussen moet produceren op basis van een lijst met cursussen, waarbij je bijvoorbeeld bepaalde cursussen er uit filtert. Dat gaat het handigst met list comprehension.

De code wordt leesbaarder wanneer je in de module planner niet dat soort list comprehensions opschrijft, maar er functies van maakt met duidelijke namen. Die neem je op in de module course.

5 De opdracht

5.1 DE VOLLEDIGE APPLICATIE

Implementeer de applicatie. Zorg er voor dat de console met de door ons meegegeven invoerbestanden het volgende laat zien:

```
Kwartiel 1
voorkennis: {'IB1102', 'IB0102'}
Te volgen cursus:
IB0402, Logica, verzamelingen en relaties, variable
geen verplichte voorkennis
gewenste voorkennis: {'IB0102'}

Kwartiel 2
voorkennis: {'IB0402', 'IB1102', 'IB0102'}
Te volgen cursus:
IB1002, Objectgeorienteerd analyseren en ontwerpen, 2021-11-15,
        2022-01-28
geen verplichte voorkennis
gewenste voorkennis: {'IB1102', 'IB0102'}

Kwartiel 3
voorkennis: {'IB0402', 'IB1102', 'IB0102', 'IB1002'}
Te volgen cursus:
IB0302, Relationele databases, 2022-02-07, 2022-04-15
geen verplichte voorkennis
gewenste voorkennis: {'IB1002'}

Kwartiel 4
voorkennis: {'IB0302', 'IB1102', 'IB0402', 'IB0102', 'IB1002'}
Te volgen cursus:
IB0202, Inleiding informatiekunde, 2022-04-25, 2022-07-01
geen verplichte voorkennis
geen gewenste voorkennis
```

N.B. In kwartiel twee hebben we de regel met de cursusgegevens afgebroken om hem goed te laten zien; in de console kan dat gewoon op één regel staan.

5.2 TESTEN

Probeer waar mogelijk je code te testen. Doe in ieder geval het volgende:

- Creëer een functie print_courses in de module course. Daarmee kun je een lijst met cursussen printen. Die functie kun je op allerlei plekken in de code gebruiken om aan de console te bekijken of er gebeurt wat je wilt dat er gebeurt.

– Test je programma ook met de variaties voor cursusengedaan.json. Hieronder zie je een overzicht van de variaties en de gewenste output We hebben lange regels hier weer afgebroken; in de console hoeft dat niet.

test-1-cursussengedaan.json

```
Kwartiel 1
voorkennis: {'IB1002', 'IB0102', 'IB1102'}
Te volgen cursus:
IB0402, Logica, verzamelingen en relaties, variable
geen verplichte voorkennis
gewenste voorkennis: {'IB0102'}

Kwartiel 2
voorkennis: {'IB1002', 'IB0102', 'IB0402', 'IB1102'}
Te volgen cursus:
IB0202, Inleiding informatiekunde, 2021-11-15, 2022-01-28
geen verplichte voorkennis
geen gewenste voorkennis

Kwartiel 3
voorkennis: {'IB1102', 'IB1002', 'IB0102', 'IB0202', 'IB0402'}
Te volgen cursus:
IB0302, Relationele databases, 2022-02-07, 2022-04-15
geen verplichte voorkennis
gewenste voorkennis: {'IB1002'}

Kwartiel 4
voorkennis: {'IB0302', 'IB1102', 'IB1002', 'IB0102', 'IB0202',
       'IB0402'}
Te volgen cursus:
IB0502, Model-driven development, 2022-04-25, 2022-07-01
verplichte voorkennis: {'IB0302'}
gewenste voorkennis: {'IB0102'}
```

test-2-cursussengedaan.json

```
Kwartiel 1
voorkennis: {'IB1002', 'IB0102', 'IB0402', 'IB0502', 'IB1102',
        'IB0202', 'IB0302'}
Te volgen cursus:
IB0602, Lineaire algebra, variable
geen verplichte voorkennis
gewenste voorkennis: {'IB0102'}

Kwartiel 2
voorkennis: {'IB1002', 'IB0102', 'IB0602', 'IB0402', 'IB0502',
        'IB1102', 'IB0202', 'IB0302'}
Te volgen cursus:
IB0702, Computernetwerken, variable
geen verplichte voorkennis
gewenste voorkennis: {'IB0102'}
```

```
Kwartiel 3
voorkennis: {'IB1002', 'IB0102', 'IB0602', 'IB0702', 'IB0402',
        'IB0502', 'IB1102', 'IB0202', 'IB0302'}
Te volgen cursus:
IB0902, Geavanceerd objectgeorienteerd programmeren, 2022-02-07,
         2022-04-15
verplichte voorkennis: {'IB1102'}
geen gewenste voorkennis

Kwartiel 4
voorkennis: {'IB1002', 'IB0102', 'IB0902', 'IB0602', 'IB0702',
        'IB0402', 'IB0502', 'IB1102', 'IB0202', 'IB0302'}
Te volgen cursus:
IB1202, Practicum ontwerpen en implementeren, 2022-04-25,
        2022-07-01
verplichte voorkennis: {'IB1102', 'IB0902', 'IB1002', 'IB0302'}
geen gewenste voorkennis
```

test-3-cursussengedaan.json

```
Kwartiel 1
voorkennis: {'IB1102', 'IB0202', 'IB0602', 'IB1202', 'IB0702',
        'IB0102', 'IB1002', 'IB0402', 'IB0302', 'IB0902',
        'IB0502'}
Te volgen cursus:
IB0802, Formele talen en automaten, variable
geen verplichte voorkennis
gewenste voorkennis: {'IB1102', 'IB0102'}

Kwartiel 2
voorkennis: {'IB1102', 'IB0202', 'IB0802', 'IB0602', 'IB1202',
        'IB0702', 'IB0102', 'IB1002', 'IB0402', 'IB0302',
        'IB0902', 'IB0502'}
Geen geschikte cursus dit kwartiel
Kwartiel 3
voorkennis: {'IB1102', 'IB0202', 'IB0802', 'IB0602', 'IB1202',
        'IB0702', 'IB0102', 'IB1002', 'IB0402', 'IB0302',
        'IB0902', 'IB0502'}
Geen geschikte cursus dit kwartiel
Kwartiel 4
voorkennis: {'IB1102', 'IB0202', 'IB0802', 'IB0602', 'IB1202',
        'IB0702', 'IB0102', 'IB1002', 'IB0402', 'IB0302',
        'IB0902', 'IB0502'}
Geen geschikte cursus dit kwartiel
```