Syntra - Python Developer - eerste jaar

Examen 1 - Leren programmeren

Dit examen bestaat uit 2 vragen. De eerste vraag is een logische schakeling. Het resultaat schrijf je weg in een tekst bestand. (zie opgave)

De tweede vraag is een programmeer-opdracht. Naast enkele kleine(re) opdrachten is er ook een opdracht die verschillende functies vraagt. Je verdient voor elk onderdeel punten. Ook al geraak je niet aan de finale oplossing, door telkens een deel van de opdracht te maken, verdien je de nodige punten om te slagen.

Het is heel belangrijk dat je je code test. Naast de functies schrijf je dus best ook de nodige testen om je eigen code te verifiëren. Daarop staan geen punten.

Alle antwoorden moeten gecommit worden in Github. Daarvoor zijn speciale repositories gemaakt. Elke student(e) kan zijn of haar repo vinden op: https://github.com/syntra-vindevoy/python1-2024-25-VOORNAAM.git

Deze opgave zit ook in deze repo onder de folder examen_1. Gelieve ook al je antwoorden in deze folder te zetten.

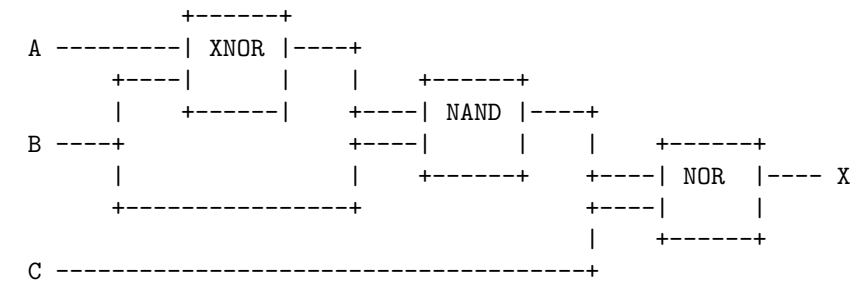
Commit na elke opdracht je code. Dat laat toe een evaluatie te maken en punten te geven. Als naam van je bestand neem je “opdracht_2_x.py” Geef als commentaar bij je commit “opdracht 2.x”. Vergeet niet te pushen.

Voor je het examen beëindigt, geef je door welke je laatste commit is. Dan kan met een pull gekeken worden dat alles effectief is opgeladen in GitHub.

Op het examen mag je handboeken en Internet gebruiken. Enkel hulp van en voor elkaar is niet toegestaan. Ook het gebruik van een telefoon is niet toegestaan. Microsoft Teams mag ook niet gebruikt worden tijdens het examen.

Vraag 1: logische schakeling (3 x 5 punten)

Gegeven: volgende logische schakeling:



Wat is de waarde van X, wanneer 0 = false, 1 = true, voor de volgende 3 gevallen:

- A = 0, B = 1, C = 1
- A = 1, B = 1, C = 0
- A = 1, B = 0, C = 1

De oplossingen van deze 3 vragen, mag je invullen in vraag1.txt.

Vraag 2: programmeer oefening

OPGELET: in deze oefening mag geen enkele Python library gebruikt worden tenzij het vermeld is. In één opdracht mag je de strings module importeren.

Opgave 2.1 (3 x 5 punten) In de wiskunde is de faculteit van een getal gedefineerd als volgt: n faculteit wordt geschreven als n!

n! = n.(n-1).(n-2)....1

en dus: 6! = 6.5.4.3.2.1

Schrijf 3 functies, telkens voor het berekenen van de faculteit.

- Eerste functie: bereken de faculteit aan de hand van een “for ... in range”
- Tweede functie: bereken de faculteit aan de hand van een “while” lus
- Derde functie: bereken de faculteit aan de hand van recursiviteit

Je functie moet rekening houden met 0!, welke gelijk is aan 1. Voorts moet er evenwel niet getest worden op de inkomende getallen. Het zullen altijd positieve integer waarden zijn.

Opgelet, de functie mag niets printen, maar moet het resultaat terug sturen. Indien je iets wil printen, moet het in een main() functie zijn. Hier mogen geen imports gebruikt worden.

Opgave 2.2 (15 punten) Schrijf een functie die een paswoord verifieert op de geldigheid. Een geldig paswoord in dit geval bevat:

- Minimum 20 characters (2 punten)
- Minstens 1 hoofletter (3 punten)
- Minstens 1 kleine letter (3 punten)
- Minstens 1 cijfer van 0 tot 9 (3 punten)
- Minstens 1 ander teken (4 punten)

In deze functie mag je de volgende imports gebruiken:

import string string.ascii_lowercase 'abcdefghijklmnopqrstuvwxyz'

string.ascii_uppercase 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

Bekijk ook volgende link:

https://www.w3schools.com/python/ref_string_isdigit.asp

Opgave 2.3 (10 punten) Schrijf voor bovenstaande functie in opgave 2.2 een volwaardige docstring ter documentatie. Mocht je er niet in geslaagd zijn om de functie te implementeren, geen nood. Deze vraag gaat niet over die functie, maar over de docstring. Ga er dus vanuit dat je functie werkt en dat je enkel nog de documentatie moet schrijven.

Vergeet niet de belangrijke delen van een docstring !

Opgave 2.4 (20 punten) Bekijk volgende link: https://www.wikihow.com/Multiply-Using-the-Russian-Peasant-Method#:~:text=Russian%20peasant%20multiplication%20is%20an,multiplication%

Deze link bevat de uitleg over de methode die gekend is als "The Russian Peasant Method" voor de vermenigvuldiging.

Implementeer deze functie. Er is een gemakkelijke manier via een "while" lus en een gemakkelijke manier via recursie. Je bent vrij te kiezen welke methode je gebruikt. (18 punten)

Schrijf voor deze functie te testen ook enkele "asserts". De assert zal moeten testen of jouw uitkomst via de functie identiek is aan de uitkomst die we gewoon via de vermenigvuldiging in Python bekomen. Belangrijk zal zijn om te testen op de randwaarden zoals 1 x 1, 2 x 2, 3 x 3, maar ook op extreem grote waarden. (2 punten). In deze oefening mogen geen imports gebruikt worden.

Opgave 2.5 (25 punten) Schrijf de volgende functies:

def opp_driehoek(h, b): (2 punten)

waarbij h de hoogte is en b de breedte (of basis) om de oppervlakte van een driehoek te berekenen. De oppervlakte van een driehoek is de hoogte maal de breedte, dit alles gedeeld door 2

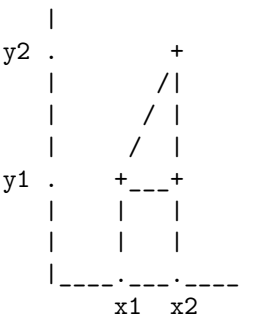
def opp_vierkant(h, b): (2 punten)

waarbij h de hoogte is en b de breedte (of basis) om de oppervlakte van een rechthoek te berekenen. De oppervlakte van een vierhoek is de hoogte maal de breedte.

def opp(x1, x2, y1, y2): (3 punten)

waarbij je 2 figuren combineert:

- een rechthoek met basis (x2 - x1) (x1 zal kleiner zijn dan x2) en als hoogte y1 (y1 zal kleiner zijn dan y2)
- een driekhoek met basis (x2 - x1) en hoogte (y2 - y1)



Bekijk de definitie van een parabool: https://nl.wikipedia.org/wiki/Parabool_%28wiskunde%29

We nemen de gemakkelijkste parabool: $y = x^2$

Schrijf een functie die de waarde van y retourneert als je x geeft:

def parabool(x): (3 punten)

Combineer alle bovenstaande functies om het oppervlak te berekenen van deze parabool tussen x1 = 0 en x2 = 4.

- Bij een eerste berekening zal je de oppervlakte berekenen alsof er een rechte lijn zou zijn tussen y1 en y2 (respectievelijk 0 en 16). Deze berekening is uiteraard niet correct.
- Verfijn de berekening door een tussenwaarde te nemen voor x. Deze tussenwaarde zal op 2 liggen, midden tussen 0 en 4, onze oorspronkelijke x1 en x2. Bereken de oppervlakte opnieuw. En kijk hoeveel nauwkeuriger dit reeds is.

- Dit zal nog steeds niet correct genoeg zijn. Dus herhaal het principe en verdubbel de x waarden.
- Doe dit tot je resultaat voor de oppervlakte nauwkeurig is tot 4 cijfers na de komma.

def opp_parabool(x1, x2): (15 punten)

In deze oefening zijn geen imports toegelaten.