NAAM STUDENT :

Student nummer:

d.d:

Mail je uitwerkingen aan: <docent@.....>

PRACTICUMTOETS 1 - THEMA 2.3 PYTHON

- tijdsduur : 80 minuten !!!!! let op dit is een proeftoets duur is 60 minuten!!!!!
- aantal opdrachten : 4 !!!!! let op dit is een proeftoets met 3 opgaven!!!!!
- toegang tot internet is **NIET** toegestaan bij het maken van opgaven
- plagiaat wordt altijd gemeld bij de examencommissie (conform Studentenstatuut)
- gebruik van boeken en gebruik van bestanden op je laptop is wel toegestaan
- aan het begin downloaden van Blackboard, onder "Python", item "practicum toets"
- aan het einde je uitwerkingen mailen naar je docent in de vorm van 4 Python source files
- na afloop dit papier weer inleveren bij je docent
- let erop dat de tijd beperkt is, maak eerst de opgaven die je makkelijk vindt
- succes !

OPGAVE 1 (2.5 PT)

In de template "opgave_1.py" zie je een functie cm_to_inch(), calc_cost_price() en een functie main() die nog niet af zijn.

a) (0,75 pt) Maak de functie cm_to_inch() af, zodat een integer waarde wordt geconverteerd naar inches(“). Deze functie wordt gebruikt om centimeters om te rekenen naar inches waarbij geld dat 1”=2,54cm
Zet onderaan in de code een asset die test of de functie een correcte berekening uitvoert.

b) (0,75 pt) Maak de functie calc_cost_pipe() af. Aan de hand van de diameter wordt uit de pricing_list de prijs gehaald. Deze prijs is per inch bij bijhorende diameter. De lijst is zo ingedeeld dat de eerste waarde in de lijst de kosten van een pijp met een diameter van één inch per inch weergeeft, de tweede waarde geeft de prijs weer voor een diameter van 2”, etc.
Zet onderaan in de code een asset die test of de functie een correcte berekening uitvoert.

c) (0.5 pt) Voeg in main() vier statements toe : twee statement die de gebruiker om decimaal getallen vraagt, en twee print statement. De eerste print geeft aan wat de input was en wat de geconverteerde waarde in inches zijn. Het tweede print statement geeft aan wat de pijp gaat kosten.

punten:

```
a) 0,6 pt; assert statements 0.15 pt
b) 0,6 pt; assert statements 0.15 pt
c) 0,25 per print statement
```

```
# Convert a length and diameter given in centimeter to inches
import math

pricing_list = (0.1 , 0.25 , 0.5 , 0.8 , 1.05 , 1.5 , 2.0 , 2.5 , 3.1 , 3.7)

def cm_to_inch(input_cm):
    input_to_inch= math.ceil((input_cm/2.54))
    return input_to_inch

def calc_cost_pipe(inch_length , inch_diameter):
    return inch_length * pricing_list[inch_diameter-1]

def main():
    length = float(input("Enter pipe length in cm: "))
    diameter = float(input("Enter pipe diameter in cm (max. 25 cm): "))
    print('The pipe with l=%dcm,d=%dcm will be converted to:  l=%d\",d=%d\"'
          % (length, diameter, cm_to_inch(length),cm_to_inch(diameter)))
    # cost could also be calculated in print statement
    costs = calc_cost_pipe(cm_to_inch(length),cm_to_inch(diameter))
    print('The costs of the pipe are: €',costs)

main()

assert cm_to_inch(254)==100 , 'Error : Faulty calculation'
assert calc_cost_pipe(100,cm_to_inch(25)), 'Error: D<25cm!!'
```

OPGAVE 2 (2.5 PT)

In de template "opgave_2.py" is een begin gemaakt met een programma dat je moet afmaken.

a) (0.5 pt) Lees de file " packet_example.txt" in als een string met de variabele naam packet_str. In deze string staan een aantal pakketjes die over het netwerk zijn verstuurd.

b) (2 pt) De inhoud van het pakket begint *na* het 15e karakter. De eerste 12 karakters vormen een hexadecimale identificatie van een netwerkkaart die het bericht heeft verstuurd. Daarna volgen er drie hexadecimale karakters die de resterende lengte van het pakket weergeven. Daarna volgt de inhoud van het pakket met de lengte die hiervoor is aangegeven. Daarop volgt het volgende pakket.
Sla elk pakketje op als een tuple in een lijst genaamd packet_list.
P.s. schrijf code dusdanig dat deze ook werkt als je de uitkomst niet kent.

Punten
a) 0,5 bij uitlezen
b) 0,5 pt netwerknummer uitlezen
0,75 pt lengte van het pakket berekenen
0,75 pt vinden volgende pakketten

```
# read file that contains network packets


packet_length=''
mac_address =''
packet_content=''
packetstart = 0
packetend = -1

packets_str = '' # string copy of the file

with open('packet_example.txt', 'r') as f:
    packets_str = f.read()



packet_list=[] #list with packets

for counter in range(0,len(packets_str)):
    if counter < (packetstart+12):
        mac_address+=packets_str[counter]

    if counter>packetstart+11 and counter<packetstart+15:
        packet length += (packets str[counter])

    if counter>=packetstart+15 and counter< \
                                    (int(packet_length,16)+packetstart+15):
        packet_content+=packets_str[counter]

    if counter>packetstart+15 and packetend<counter:
```

```
        packetend=(int(packet_length,16)+packetstart+15)

    if counter == (packetend):
        #print(mac_address, packet_length, packet_content)
        packet_list.append((mac_address, packet_length, packet_content))
        packetstart=counter
        mac_address= packets_str[counter]
        packet_content=''
        packet_length=''

#loop did not place last packet in list -> put last packet in list
packet_list.append((mac_address, packet_length, packet_content))


#packet_list.sort(key=lambda packet: packet[1])
for packet in packet_list:
    print(packet)
```

output (5 paketten):

```
('012a6f812341', '606', '21CD136EFF214EB420CCE191B2F20453EBB0DF62D972986FFCBC6F805
('b12a6dea2212', '499', '3B7101518828C6FB77C33BE15856E01D05C5B1B7B78928579B74B9CCC
('cd2a6f87833a', '12C', '0B23854E736B83CBFB88FE4B381C87EDDDE0FBF896D7D33FF52712806
('210a6f8d99dd', '052', '894758DBB9C3CC0D2D4B9829B979C6C9639BC9035950156E1C3C3E14F
('a6f8123414fa', '0DC', '56EDD329FFD5D156F206F47BE8294479583CC262EDDE50D4C18F2CC44

Process finished with exit code 0
```

OPGAVE 3 (2.5 PT)

In de template "opgave_3.py" is een begin gemaakt met een programma dat je moet afmaken.

De klasse Employee is bijna hetzelfde als de klasse Management.

a) (0.5 pt) De klasse Employee heeft de volgende 5 attributen: name, role, departement, salary, isTemporary.
Voeg aan Employee een constructor toe die deze 5 velden definieert. De default waarde van is_Temporary is False.

b) (0.5pt) De klasse Management heeft de volgende 6 attributen : name, role, departement, salary, bonus, isTemporary. Hierbij is het attribuut bonus een toeslagpercentage over het jaarsalaris
Voeg aan Management een constructor toe die deze 5 velden definieert. De default waarde van is_Temporary is False.

c) (0.5 pt) Voeg een def __repr__() toe aan Employee zodat een print() van een Employee-instantie wordt afgedrukt : de name, role, yearly_income en isTemporary. Het jaarlijks inkomen moet in veelvouden van 1000 worden afgedrukt. B.v. €22500 wordt als €22.5k afgedrukt.

d) (0.5 pt) De klasse Management moet print() van een instantie van deze klasse, vrijwel hetzelfde weergeven als bij een Employee instantie echter moet bij het jaarlijks inkomen ook de bonus (is percentage van het jaar inkomen) verwerkt zijn.

e) (0,5) De bonus wordt aangepast zodat deze uit twee delen komt te bestaan. Het eerste deel is een percentage dat uitgekeerd wordt op basis van het jaarlijks salaris (zoals in d). Het tweede deel is een verhoging van de bonus percentage van een halve procent per medewerker die binnen de afdeling valt, die de manager beheert. Zijn er meerdere managers op dezelfde afdeling dan wordt het personeel evenredig verdeelt over de managers.
Pas de classes Employee en Manager dusdanig aan dat er per afdeling wordt bijgehouden hoeveel instanties er gecreëerd zijn zodat de bonus per manager berekend kan worden met behulp van deze registratie. Het registreren van managers/employees/managers employees vindt plaats d.m.v. een dictionary die per afdeling bijhoudt hoeveel employees/managers er zijn.

```
class Employee:

    employees = {}


    def __init__(self, name, role, departement, salary, isTemporary=True):
        self.name = name
        self.isTemporary = isTemporary
        self.role = role
        self.salary = salary
        self.departement = departement
        if (self.employees.get(departement)) == None:
            counter = 0
        else:
            counter = self.employees.get(departement)
        self.employees.update({departement:counter+1})
```

```
    def money_in_k(self, money):
        return money/1000

    def yearly_income(self):
        return (self.money_in_k(12*self.salary))

    def __repr__(self):
        return ('Employee: %s, %s, %s, €%sk, %s' % (self.name, self.role,
self.departement, self.yearly_income(),
                self.isTemporary))


class Management(Employee):

    managers = {}

    def __init__(self, name, role, departement, salary, bonus, isTemporary):
        Employee.__init__(self, name, role, departement, salary, isTemporary)
        self.bonus = bonus
        if (self.managers.get(departement))==None:
            counter = 0
        else:
            counter = self.managers.get(departement)
        self.managers.update({departement:counter+1})

    def __repr__(self):
        return ('Management: %s, %s, %s, €%sk, %s' % (self.name, self.role,
self.departement, self.yearly_income(),self.isTemporary))

    def yearly_income(self):
        yearly_salery = 12 * self.salary
        employee_bonus = ((Employee.employees.get(self.departement) -
self.managers.get(self.departement)) * 0.5) / \
                         self.managers.get(self.departement)
        yearly_bonus =  yearly_salery * (1.0+(self.bonus + employee_bonus)/100)
        return self.money_in_k(yearly_bonus)



company = []
company.append(Employee('Johnson', 'sr. clerc', 'finance', 2300, False))
company.append(Management('Bush jr.', 'manager', 'finance', 5400, 10, True))
company.append(Employee('Jasons', 'clerc', 'finance', 1900, False))
company.append(Management('Bushes', 'manager', 'R&D', 5400, 10, True))
company.append(Employee('Hunter', 'researcher', 'R&D', 2300, False))
company.append(Employee('Vries', 'researcher', 'R&D', 2300, False))
company.append(Employee('Locker', 'researcher', 'R&D', 2300, False))
company.append(Management('Schneier', 'manager', 'R&D', 6400, 10, True))



for employee in company:
    print(employee)
```

output :
Employee: Johnson, sr. clerc, finance, €27.6k, False
Management: Bush jr., manager, finance, €71.928k, True
Employee: Jasons, clerc, finance, €22.8k, False
Management: Bushes, manager, R&D, €71.766k, True
Employee: Hunter, researcher, R&D, €27.6k, False
Employee: Vries, researcher, R&D, €27.6k, False
Employee: Locker, researcher, R&D, €27.6k, False
Management: Schneier, manager, R&D, €85.056k, True