



Oefening 3



We maken nu een grotere oefening. Je kan die best eerst volledig maken in PyCharm. INGINIOUS test elk deeltje afzonderlijk als je al sneller feedback wil.

Schrijf een klasse `Date` die de representatie van een datum implementeert. Gebruik geen ingebouwde functies of klassen. We doen dat in stappen. Eerst schrijf je een aantal hulpfuncties (geen methodes!). Pas daarna start je met de klasse `Date` en ga je die hulpfuncties gebruiken (geen duplicate code!). Je wordt hier ook beoordeeld op code quality:

- zorg dat je functies niet langer zijn dan 50 regels. Als die toch langer is, splits die dan op in logische deelfuncties die je afzonderlijk test.
- gebruik geen te diepe nesting (if binnen for binnen if binnen while ...)
- schrijf documentatie in elke functie/methode via een docblock. In Pycharm ga je achter het signatuur van de functie staan (bv na `def test():`), je drukt op enter, dan geef je 3 aanhalingstekens in (""") en druk je terug op enter. Je krijgt dan een blok waarin je commentaar kan ingeven.

Your answer passed the tests! Your score is 100.0%. [Submission #672115fa3baf09455aba766c (2024-10-29 17:06:02)]

Question 1: Schrikkeljaar

✓ Perfect

Schrijf een functie `leap_year` die nakijkt of een gegeven jaar een schrikkeljaar is. Als het een schrikkeljaar is, geef je `True` terug anders `False`. Een schrikkeljaar is deelbaar door 4 maar niet door 100, tenzij het deelbaar is door 400. E.g. 2004 en 2000 zijn schrikkeljaren, maar 1800 en 1900 niet.

```
1 def leap_year(y):
2     # result = als y is deelbaar door 4 and (y niet deelbaar door 100 of y deelbaar door 400)
3     return y % 4 == 0 and (y % 100 != 0 or y % 400 == 0)
```

Submit

Question 2: Aantal dagen in een maand

✓ Perfect

Schrijf een functie `days_in_month(m,y)` die een maand en een jaar meekrijgt als parameters en het aantal dagen van die maand teruggeeft.

```
>>> days_in_month(10,2021)
31
>>> days_in_month(2,2021)
28
```

```
1 def days_in_month(m, y):
2     # this is a list with all the day of the 12 months
3     d = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
4     if m == 2: return 29 if leap_year(y) else 28
5     return d[m-1]
```

Submit

Question 3: Volgende maand

✓ Perfect

✕

Schrijf een functie `next_month(m,y)` die een tuple teruggeeft met daarin resp. de volgende maand en het jaar.

```
>>> next_month(10,2021)
(11,2021)
>>> next_month(12,2021)
(1,2022)
```

```
1 def next_month(m,y):
2     # return m + 1, y als de maand minder is dan 12 anders (m + 1) % 12, y + 1
3     return (m + 1, y) if m + 1 <= 12 else ((m + 1) % 12, y + 1)
```

Submit

Question 4: De klasse Date

✓ Perfect

✕

Schrijf een klasse `Date` die de representatie van een datum implementeert met een constructor die resp. dag, maand en jaar meekrijgt als parameters.

```
1 class Date:
2     # gewoon initialization
3     def __init__(self, dag, maand, jaar):
4         self.dag = dag
5         self.maand = maand
6         self.jaar = jaar
```

Submit

Question 5: Omzetting naar een string

✓ Perfect



Schrijf een methode `__str__()` (let op de dubbele underscores voor en na `str`; dat is een ingebouwde methode zoals `__init__()`) die de datum als string teruggeeft in het formaat `dd-mm-yyyy`.

```
1 def __str__(self):
2     # dd--mm-yyyy formaat
3     return f"{'0' if self.dag < 10 else ''}{self.dag}-{'0' if self.maand < 10 else ''}{self.maand}-{self.jaar}"
```

Submit

Question 6: Vergelijking van data

✓ Perfect



Schrijf een methode `comes_before(Date)` die een andere Date als parameter meekrijgt en `True` teruggeeft als de datum (`self`) voor de opgegeven datum komt en anders `False`.

```
1 def comes_before(self, d):
2     # if self.day is before d.day and self.year is before or equal to d.jaar then it's true or
  self.jaar is before than d.jaar
3     return (self.maand < d.maand and self.jaar <= d.jaar) or self.jaar < d.jaar or (self.dag < d.dag
  and self.maand <= d.maand and self.jaar <= d.jaar)
4
```

Submit

Question 7: Dagen bijtellen

✓ Perfect



Schrijf een methode `add_days(n)` die `n` dagen telt bij de datum.

```
1 def add_days(self, n):
2     # add the extra day to the current day
3     self.dag += n
4
5     i = 0
6     # while the current day is bigger than the days of the current month do the following
7     while self.dag > days_in_month(self.maand % 12 if self.maand > 12 else self.maand, self.jaar):
8         # if i equals the number of days in the current month
9         if i == days_in_month(self.maand % 12 if self.maand > 12 else self.maand, self.jaar):
10            # add one month to the current month
11            self.maand += 1
12            # decrease the current day with the number of days of the current month
13            self.dag -= i
14            # reset i to 0 to start counting
15            i = 0
16
17            # if the current month is bigger than 12 then
18            if self.maand > 12:
```

```

19         # add one year
20         self.jaar += self.maand // 12
21         # reset the current month, so here it will equal 1
22         self.maand %= 12
23
24
25         i += 1

```

Submit

Question 8: Verschil in dagen

✓ Perfect



Schrijf een methode `get_days_to(Date)` die een datum meekrijgt en het aantal dagen tot een opgegeven datum bepaalt (dit kan negatief zijn als de gegeven datum voor de huidige datum komt).

```

>>> d1 = Date(11,10,2021)
>>> d2 = Date(12,10,2021)
>>> d1.get_days_to(d2)
1
>>> d2.get_days_to(d1)
-1

```

```

1 def get_days_to(self, d):
2     # saving the entered variables so that I can assign them again
3     # in the end because they will be changed in the code
4     original_self = [self.dag, self.maand, self.jaar]
5     original_d = [d.dag, d.maand, d.jaar]
6
7     i = 0
8     # if self comes before d
9     if self.comes_before(d):
10        # add 1 day to self.day and to i until self and d are the same
11        while self.dag != d.dag or self.maand != d.maand or self.jaar != d.jaar:
12            self.add_days(1)
13            i += 1
14
15    else:
16        # add 1 day to self.day and -1 to i until self and d are the same
17        while self.dag != d.dag or self.maand != d.maand or self.jaar != d.jaar:
18            d.add_days(1)
19            i -= 1
20
21    self.dag = original_self[0]
22    self.maand = original_self[1]
23    self.jaar = original_self[2]
24
25    d.dag = original_d[0]
26    d.maand = original_d[1]
27    d.jaar = original_d[2]
28
29    return i
30

```

Submit

Question 9: Is je code gedocumenteerd?

✓ Perfect



Submit

Question 10: Zijn de functies niet te lang?

✓ Perfect



Submit

Question 11: Is de nesting diepte ok?

✓ Perfect



Dit is nog ok:

```
for i in range(3):
    if i % 2 == 0:
        if i % 4 == 0:
            print("hallo")
```

Dit niet meer:

```
for i in range(3):
    if i % 2 == 0:
        if i % 4 == 0:
            for j in range(3):
                print("hallo")
```

Submit