

PRACTICUM REPORT
ALGORITHM AND DATA STRUCTURES
MODUL 1 : PYTHON REVIEW



Disusun Oleh :

JOHN DOE

L2002XXXXX

X

INFORMATICS ENGINEERING
FACULTY OF COMMUNICATION AND INFORMATICS
UNIVERSITAS MUHAMMADIYAH SURAKARTA

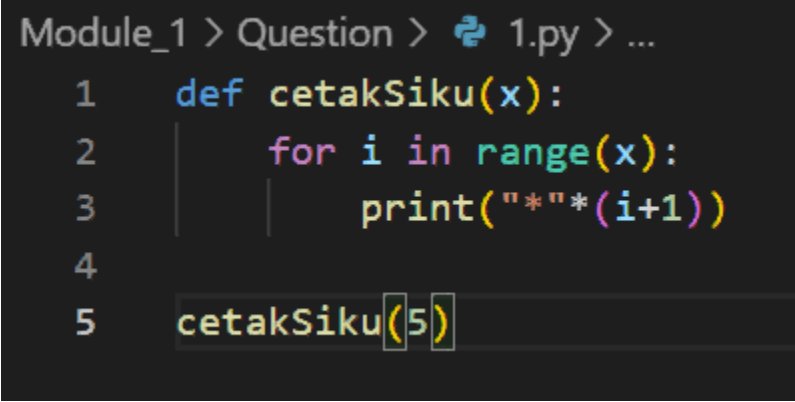
1.11 Questions

1. Create a function cetakSiku(x) that will print the following:

```
*  
**  
***  
****  
*****
```

The x value shows the height of the triangle (the image above means it can be obtained from running cetakSiku(5)). Use a double loop!

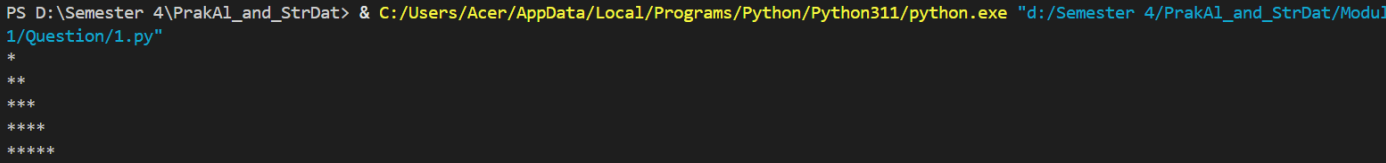
- Program Code



```
Module_1 > Question > 1.py > ...  
1  def cetakSiku(x):  
2      for i in range(x):  
3          print("*"*(i+1))  
4  
5  cetakSiku(5)
```

Picture 1.1 *the code.*

- Practicum Result Screenshot



```
PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Modul  
1/Question/1.py"  
*  
**  
***  
****  
*****
```

Gambar 1.2 the output

2. Create a function that accepts two positive integers, which will draw a rectangular shape. Example of calling :

```
>>> gambarlahPersegiEmpat(4,5) # Button <enter>
```

Presed

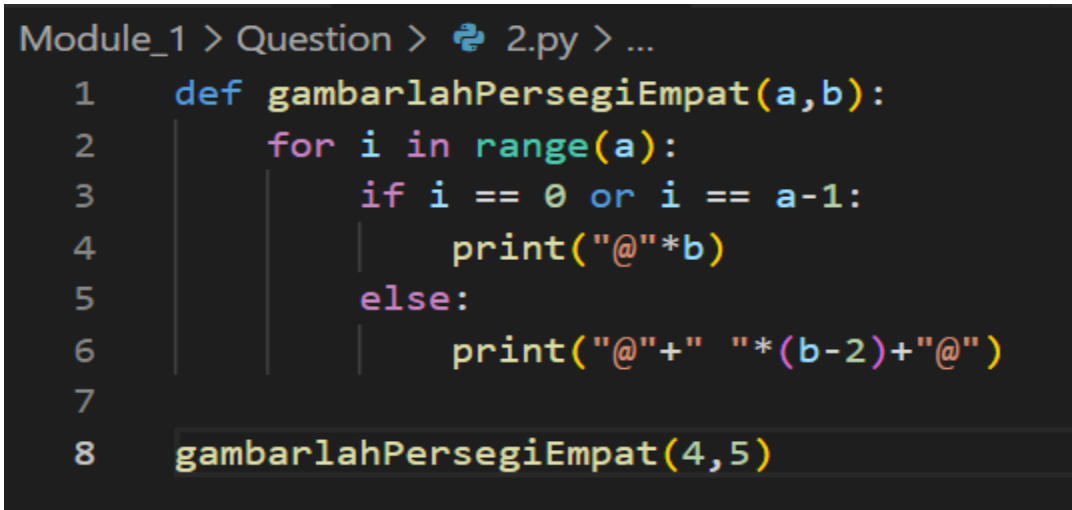
```
@ @ @ @ @
```

```
@       @
```

```
@       @
```

```
@ @ @ @ @
```

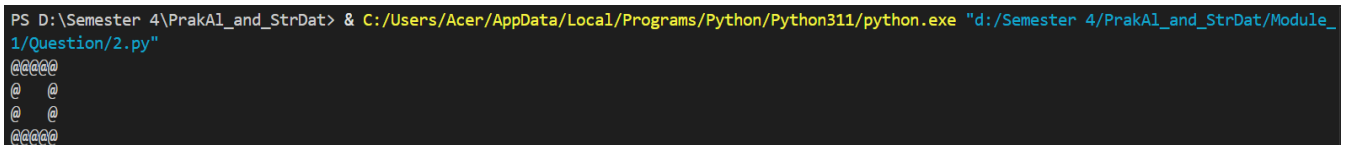
- **Program Code**



```
Module_1 > Question > 2.py > ...
1  def gambarlahPersegiEmpat(a,b):
2      for i in range(a):
3          if i == 0 or i == a-1:
4              print("@"*b)
5          else:
6              print("@"+" "*(b-2)+"@")
7
8  gambarlahPersegiEmpat(4,5)
```

Picture 2.1 the code.

- **Praktikum result screenshot**



```
PS D:\Semester 4\PrakA1_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakA1_and_StrDat/Module_1/Question/2.py"
@@@@@
@  @
@  @
@@@@@
```

Picture 2.2 the output.

3. The following are two interrelated questions
- Create a function that accepts a string and returns a list of two integers. These two returned integers are: the number of letters in that string and the number of vowels (vowels are vowels) in that string. Example of calling:

```
>>> k = jumlahHurufVokal('Surakarta')
>>> k
```

(9, 4)# Nine letters, and four of them are

Vowels

- b. Same as question (a) above, but now the consonants are counted. There's only one different line in the code! Example of calling:

```
>>> k = jumlahHurufKonsonan('Surakarta')
```

```
>>> k
```

(9, 5)# Nine

- Program code

```
Module_1 > Question > 3.py > jumlahHurufKonsonan
1  def jumlahHurufVokal(s):
2      vowels = 'aiueoAIUEO'
3      total_letters = 0
4      vowel_count = 0
5      for char in s:
6          if char.isalpha():
7              total_letters += 1
8              if char in vowels:
9                  vowel_count += 1
10     return (total_letters, vowel_count)
11
12     def jumlahHurufKonsonan(s):
13         vowels = 'aiueoAIUEO'
14         total_letters = 0
15         consonant_count = 0
16         for char in s:
17             if char.isalpha():
18                 total_letters += 1
19                 if char not in vowels:
20                     consonant_count += 1
21         return (total_letters, consonant_count)
22
23     k = jumlahHurufVokal('Surakarta')
24     print(k)
25
26     v = jumlahHurufKonsonan('Surakarta')
27     print(v)
```

Picture 3.2 the code

- Praktikum results screenshot

```
PS D:\Semester 4\PrakAl_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAl_and_StrDat/Module_1/Question/3.py"
(9, 4)
(9, 5)
```

4.

Create a function that calculates the average of an array containing numbers. The average has a formula

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n} \quad (1.3)$$

But remember that Python starts index from 0. The function must have a form `rerata(x)`, where `x` is a list containing the numbers whose average you want to calculate. So, your work will have a form:

□ Create a file with contents like this

□ Run the program by pressing “F5”, then call the program like this

```
rerata([1,2,3,4,5]) #hasilnya 3
g = [3,4,5,4,3,4,5,2,2,10,11,23]
rerata(g)
```

Extra credit: Also create a function to calculate the variance and standard deviation with the prototype, respectively, `variance(x)` and `stdev(x)`.

- Program code

```
Module_1 > Question > 4.py > ...
1  def rerata(b):
2      total = sum(b)
3      n = len(b)
4      hasil = total / n
5      return hasil
6
7  def variance(x):
8      mean = rerata(x)
9      total_squared_diff = sum((xi - mean) ** 2 for xi in x)
10     var = total_squared_diff / len(x)
11     return var
12
13 def stdev(x):
14     var = variance(x)
15     std_dev = var ** 0.5
16     return std_dev
17
18
19 print(rerata([1, 2, 3, 4, 5]))
20 g = [3, 4, 5, 4, 3, 4, 5, 2, 2, 10, 11, 23]
21 print(rerata(g))
22 print(variance(g))
23 print(stdev(g))
```

Picture 4.1 the code

- Practicum results screenshot

```
PS D:\Semester 4\PrakA1_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakA1_and_StrDat/Module_1/Question/4.py"
3.0
6.333333333333333
32.72222222222222
5.720334100576838
```

Picture 4.2 the output

5. Create a function to determine whether an integer is a prime number or not. To make it easier, complete the program below Once done, run the above program and then test it in Python Shell:

```
from math import sqrt as sq
def apakahPrima(n):
    n = int(n) # If the number is a fraction,
               # discard the fraction.
    assert n >= 0 # Only accepts non-negative
                 # numbers.
    primaKecil = [2, 3, 5, 7, 11] # If the number is
    # small, then
    bukanPrKecil = [0, 1, 4, 6, 8, 9, 10] # caught here.
    if n in primaKecil:
        return True
    elif n in bukanPrKecil:
        return False
    else:
        for I in range(2, int(sq(n)) + 1): # Just
            # get to the roots.
            #.....#Your task
            #.....#is fill
            #.....# fill in this dot.
```

```
apakahPrima(17)
apakahPrima(97)
apakahPrima(123)
```

- Program code

```
1  from math import sqrt as sq
2
3  def apakahPrima(n):
4      n = int(n)
5      assert n >= 0
6      primaKecil = [2, 3, 5, 7, 11]
7      bukanPrKecil = [0, 1, 4, 6, 8, 9, 10]
8      if n in primaKecil:
9          return True
10     elif n in bukanPrKecil:
11         return False
12     else:
13         for i in range(2, int(sq(n)) + 1):
14             if n % i == 0:
15                 return False
16         return True
17
18     # Example usage:
19     print(apakahPrima(17))
20     print(apakahPrima(97))
21     print(apakahPrima(123))
```

Picture 5.1 the code

- **Practicum result screenshot**

```
PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Module_1/Question/5.py"
True
True
False
```

Picture 5.2 the output.

6. Write a program that prints all the prime numbers from 2 to 1000. Make use of functions `apakahPrima()` on the number above.

- **Program code**

```
1  def apakahPrima(n):
2      if n <= 1:
3          return False
4      for i in range(2, int(n**0.5) + 1):
5          if n % i == 0:
6              return False
7      return True
8
9  def cetakPrima():
10     for i in range(2, 1001):
11         if apakahPrima(i):
12             print(i)
13
14     # Contoh penggunaan
15     cetakPrima()
```

Picture 6.1 the code

- **Practicum results screenshot**

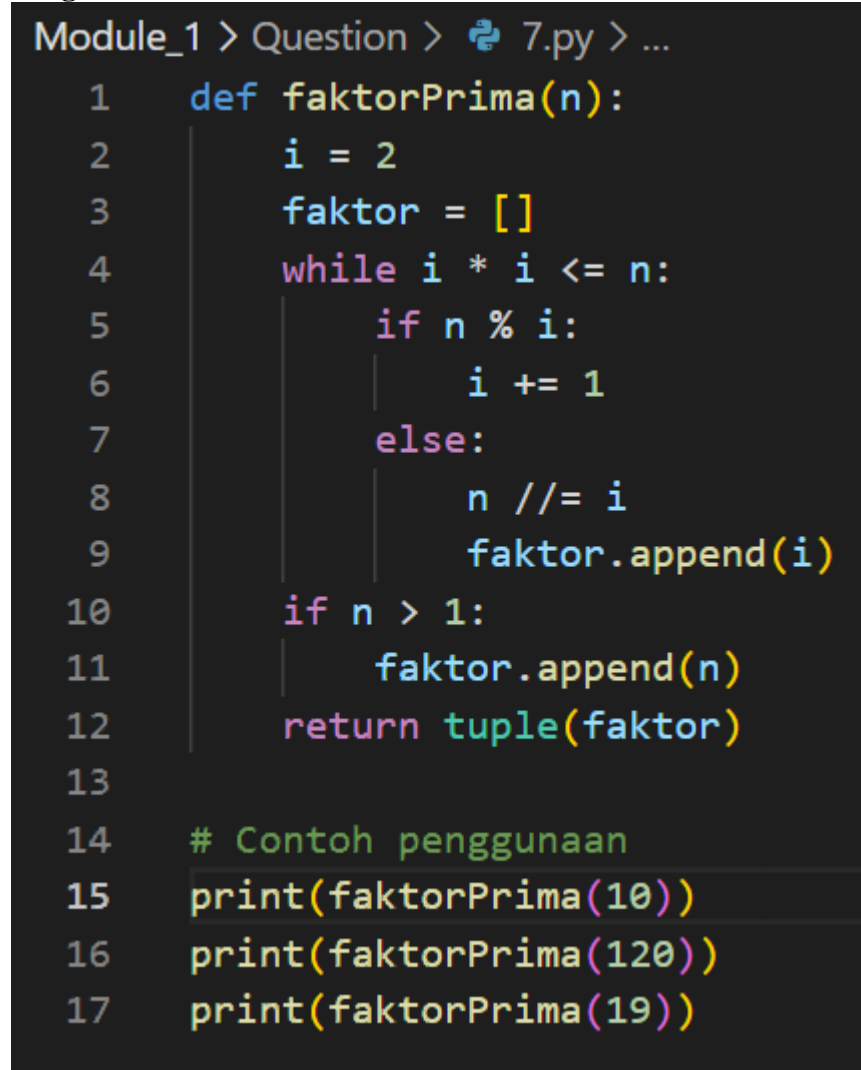
PS D:\Semester 4		479	
1/Question/6.py"	199	487	797
2	211	491	809
3	223	499	811
5	227	503	821
7	229	509	823
11	233	521	827
13	239	523	829
17	241	541	839
19	251	547	853
23	257	557	857
29	263	563	859
31	269	569	863
37	271	571	877
41	277	577	881
43	281	587	883
47	283	593	887
53	293	599	907
59	307	601	911
61	311	607	919
67	313	613	929
71	317	617	937
73	331	619	941
79	337	631	947
83	347	641	953
89	349	643	967
97	353	647	971
101	359	653	977
103	367	659	983
107	373	661	991
109	379	673	997
113	383	677	
127	389	683	
131	397	691	
137	401	701	
139	409	709	
149	419	719	
151	421	727	
157	431	733	
163	433	739	
167	439	743	
173	443	751	
179	449	757	
181	457	761	
191	461	769	
193	463	773	
197	467	787	
	479		

Picture 6.2 the output

7. Write a program that accepts a positive integer and provides its prime factorization. Prime factorization is factoring an integer into its constituent prime numbers. Example:

```
>>> faktorPrima(10)
(2, 5)
>>> faktorPrima(120)
(2, 2, 2, 3, 5)
>>> faktorPrima(19)
(19,)
```

- **Program code**

A screenshot of a Python IDE window titled 'Module_1 > Question > 7.py > ...'. The code defines a function 'faktorPrima(n)' that takes a positive integer 'n' and returns its prime factorization as a tuple. The function uses a while loop to find factors starting from 'i = 2'. If 'n' is divisible by 'i', 'i' is added to the 'faktor' list and 'n' is divided by 'i'. If 'n' is greater than 1 after the loop, it is added to the list. The function returns the 'faktor' list as a tuple. Below the function definition, there are three example calls: 'print(faktorPrima(10))', 'print(faktorPrima(120))', and 'print(faktorPrima(19))'.

```
Module_1 > Question > 7.py > ...
1  def faktorPrima(n):
2      i = 2
3      faktor = []
4      while i * i <= n:
5          if n % i:
6              i += 1
7          else:
8              n //= i
9              faktor.append(i)
10     if n > 1:
11         faktor.append(n)
12     return tuple(faktor)
13
14     # Contoh penggunaan
15     print(faktorPrima(10))
16     print(faktorPrima(120))
17     print(faktorPrima(19))
```

Picture 7.1 the code

- **Practicum results screenshot**

```
PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Module_1/Question/7.py"
(2, 5)
(2, 2, 2, 3, 5)
(19,)
```

Picture 7.2 the output.

8. Create a function **apakahTerkandung(a,b)** which accepts two strings **a** and **b**, then determines whether string **a** is contained in string **b**. The execution is like this:

```
>>> h = 'do'
>>> k = 'Indonesia tanah air beta'
>>> apakahTerkandung(h,k)
True
>>> apakahTerkandung('pusaka',k)
False
```

- **Program code**

```
Module_1 > Question > 8.py > ...
1  def apakahTerkandung(a, b):
2      return a in b
3
4  # Example usage:
5  h = 'do'
6  k = 'Indonesia tanah air beta'
7  print(apakahTerkandung(h, k))
8  print(apakahTerkandung('pusaka', k))
```

Picture 8.1 the code.

- **Practicum results screenshot**

```
PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Module_1/Question/8.py"
True
False
```

Picture 8.1 the output.

9. Create a program to print numbers from 1 to 100. If the number is a multiple of 3, print 'Python'. If it is a multiple of 5, print 'UMS'. If it fits multiples of 3 and multiples of 5, print 'Python UMS'. So the result:

```
1
2
Python
4
UMS
```

```
Python
7
8
Python
UMS
11
Python
13
14
Python UMS
16
17
...
```

- **Program code**

```
1  for i in range(1, 101):
2      if i % 3 == 0 and i % 5 == 0:
3          print("Python UMS")
4      elif i % 3 == 0:
5          print("Python")
6      elif i % 5 == 0:
7          print("UMS")
8      else:
9          print(i)
```

Picture 9.1 the code

- **Practicum results screensho**

1/Question/9.py"

1
2
Python
4
UMS
Python
7
8
Python
UMS
11
Python
13
14
Python UMS
16
17
Python
19
UMS
Python
22
23
Python
UMS
26
Python
28
29
Python UMS
31
32
Python
34
UMS
Python
37
38
Python
UMS

41
Python
43
44
Python UMS
46
47
Python
49
UMS
Python
52
53
Python
UMS
56
Python
58
59
Python UMS
61
62
Python
64
UMS
Python
67
68
Python
UMS
71
Python
73
74
Python UMS
76
77
Python
79
UMS
Python
82
83

83
Python
UMS
86
Python
88
89
Python UMS
91
92
Python
94
UMS
Python
97
98
Python
UMS

Picture 9.2 the output.

10. Make a modification of Example 1.4, to capture the case where the determinant is less than zero. If this happens, display an on-screen warning like this:

```
>>> selesaikanABC(1,2,3)
```

Determinannya negatif. Persamaan tidak mempunyai akar real.

```
>>>
```

- **Program code**

```
Module_1 > Question > 10.py > ...
1  import math
2
3  def selesaikanABC(a, b, c):
4      determinant = b**2 - 4*a*c
5      if determinant < 0:
6          print("Determinannya negatif. Persamaan tidak mempunyai akar real.")
7      else:
8          root1 = (-b + math.sqrt(determinant)) / (2*a)
9          root2 = (-b - math.sqrt(determinant)) / (2*a)
10         return root1, root2
11
12 # Example usage:
13 print(selesaikanABC(1, 2, 3))
14 print(selesaikanABC(1, -3, 2))
```

Picture 10.1 the code.

- **Practicum results**

```
PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Module_1/Question/10.py"
Determinannya negatif. Persamaan tidak mempunyai akar real.
None
(2.0, 1.0)
```

Picture 10.2 the output.

11. Create a function **apakahKabisat()** yang menerima suatu angka (tahun). which receives a number (year). If the year is a leap year, return it **True**. If it's not leap, return it **False**.

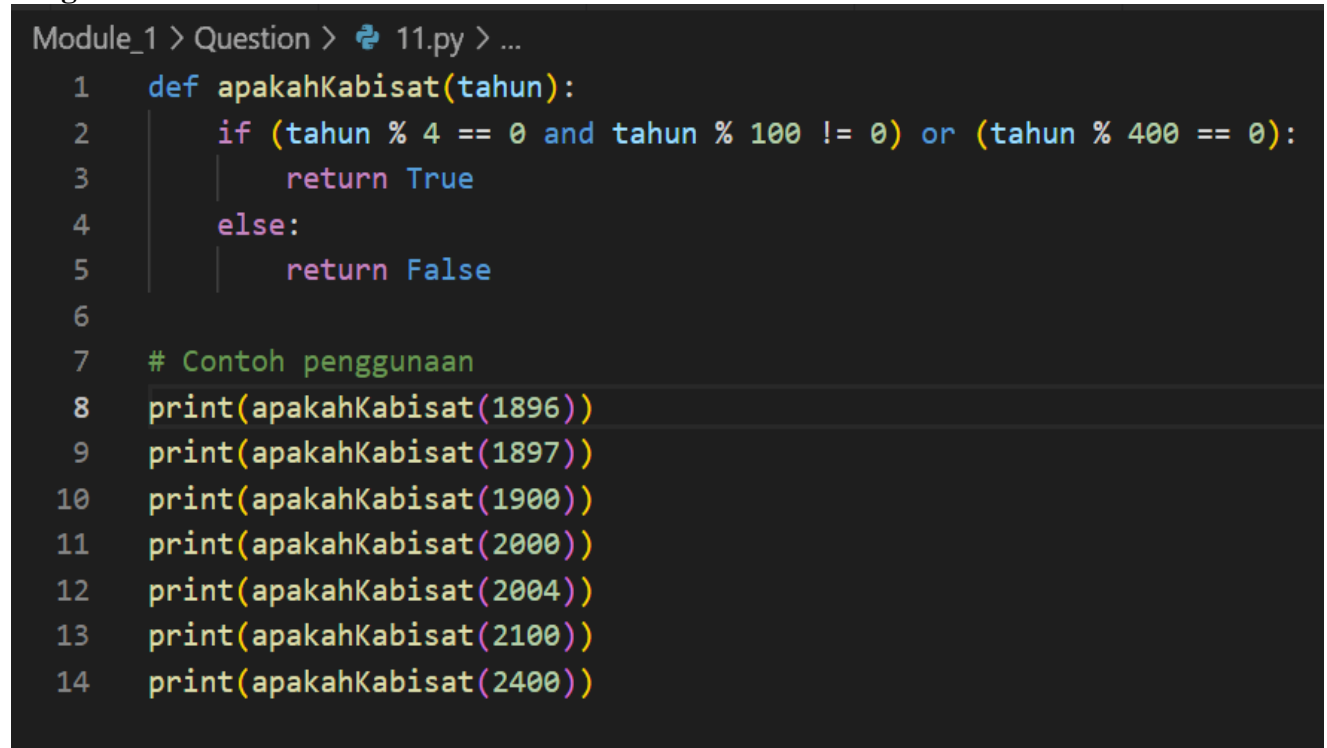
A leap year – a year with a date of February 29 – is a year that is divisible by 4, unless it is divisible by 100 (in which case it is not a leap year). But if it is divisible by 400, it is a leap year (even if it is divisible by 100).

The following are some examples:

- 1896 leap year (divisible by 4)
- 1897 was not a leap year (obviously)

- 1900 is not a leap year (even though it is divisible by 4, it is divisible by 100, and not divisible by 400)
- 2000 leap years (divisible by 400)
- 2004, 2008, 2012, 2016, ..., 2096 leap year
- 2100, 2200, 2300 are not leap years
- 2400 leap years

- **Program code**



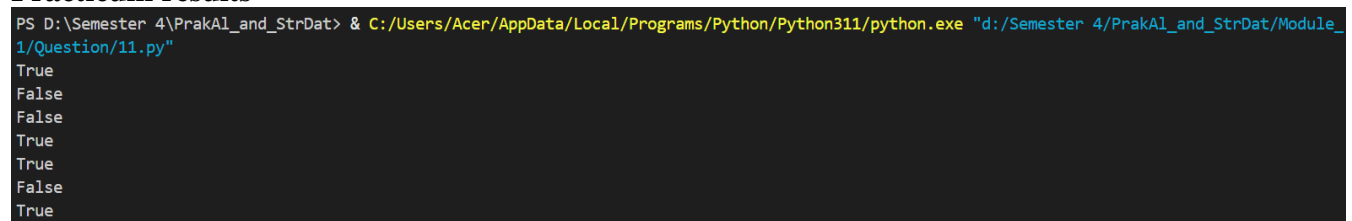
```

Module_1 > Question > 11.py > ...
1  def apakahKabisat(tahun):
2      if (tahun % 4 == 0 and tahun % 100 != 0) or (tahun % 400 == 0):
3          return True
4      else:
5          return False
6
7  # Contoh penggunaan
8  print(apakahKabisat(1896))
9  print(apakahKabisat(1897))
10 print(apakahKabisat(1900))
11 print(apakahKabisat(2000))
12 print(apakahKabisat(2004))
13 print(apakahKabisat(2100))
14 print(apakahKabisat(2400))

```

Picture 11.1 the code.

- **Practicum results**



```

PS D:\Semester 4\PrakA1_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakA1_and_StrDat/Module_1/Question/11.py"
True
False
False
True
True
False
True

```

Picture 11.2 the results

12. Number guessing game program. Create a program whose global flow is like this:

- The computer generates a random integer between 1 and 100. The value is stored in a variable and is not displayed to the user.
- The user is asked to guess the number, entered via the keyboard.
- If the input number is too small or too large, the user gets feedback from the computer ("That number is too small. Try

again”)

- The process is repeated until the number is guessed or until a certain number of guesses are wrong 8 .

When the program is run, the process is more or less like below
Number guessing game.

I store a round number between 1 and 100. Guess
what.

Enter 1st guess: > 50

It's too small. Try again.

Enter the 2nd guess: > 75

It's too big. Try again.

Enter the 3rd guess: > 58

1920

Yes. You are right

- Program code

```
Module_1 > Question > 12.py > ...
1  import random
2
3  def tebak_angka():
4      angka_rahasia = random.randint(1, 100)
5      tebakan = None
6      jumlah_tebakan = 0
7
8      print("Permainan tebak angka.")
9      print("Saya menyimpan sebuah angka bulat antara 1 sampai 100. Coba tebak.")
10
11     while tebakan != angka_rahasia:
12         jumlah_tebakan += 1
13         tebakan = int(input(f"Masukkan tebakan ke-{jumlah_tebakan}:> "))
14
15         if tebakan < angka_rahasia:
16             print("Itu terlalu kecil. Coba lagi.")
17         elif tebakan > angka_rahasia:
18             print("Itu terlalu besar. Coba lagi.")
19         else:
20             print("Ya. Anda benar")
21
22 if __name__ == "__main__":
23     tebak_angka()
```

Picture 12.1 the code.

- Practicum results

```
1/Question/12.py"
Permainan tebak angka.
Saya menyimpan sebuah angka bulat antara 1 sampai 100. Coba tebak.
Masukkan tebakan ke-1:> 80
Itu terlalu kecil. Coba lagi.
Masukkan tebakan ke-2:> 99
Itu terlalu besar. Coba lagi.
Masukkan tebakan ke-3:> 90
Itu terlalu kecil. Coba lagi.
Masukkan tebakan ke-4:> 93
Itu terlalu kecil. Coba lagi.
Masukkan tebakan ke-5:> 95
Itu terlalu besar. Coba lagi.
Masukkan tebakan ke-6:> 94
Ya. Anda benar
```

Picture 12.2 the results

13. Create a function **katakan()** which accepts a positive integer and returns a string which is the Indonesian pronunciation of that number. Example:
- ```
>>> katakan(3125750)
'Tiga juta seratus dua puluh lima ribu tujuh ratus lima puluh'
```
- Limit the input to less than one billion. Extra credit: use recursion.

- **Program code**



```

Module_1 > Question > 13.py > ...
1 def katakan(angka):
2 satuan = ["", "satu", "dua", "tiga", "empat", "lima", "enam", "tujuh", "delapan", "sembilan", "sepuluh", "sebelas"]
3
4 def terbilang(n):
5 if n < 12:
6 return satuan[n]
7 elif n < 20:
8 return terbilang(n - 10) + " belas"
9 elif n < 100:
10 return terbilang(n // 10) + " puluh" + (" " if n % 10 == 0 else " " + terbilang(n % 10))
11 elif n < 200:
12 return "seratus" + (" " if n % 100 == 0 else " " + terbilang(n % 100))
13 elif n < 1000:
14 return terbilang(n // 100) + " ratus" + (" " if n % 100 == 0 else " " + terbilang(n % 100))
15 elif n < 2000:
16 return "seribu" + (" " if n % 1000 == 0 else " " + terbilang(n % 1000))
17 elif n < 1000000:
18 return terbilang(n // 1000) + " ribu" + (" " if n % 1000 == 0 else " " + terbilang(n % 1000))
19 elif n < 1000000000:
20 return terbilang(n // 1000000) + " juta" + (" " if n % 1000000 == 0 else " " + terbilang(n % 1000000))
21 else:
22 return terbilang(n // 1000000000) + " milyar" + (" " if n % 1000000000 == 0 else " " + terbilang(n % 1000000000))
23
24 return terbilang(angka)
25
26 # Contoh penggunaan
27 print(katakan(3125750))

```

Picture 13.1 the code.

#### - Practicum results

```

PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Module_1/Question/13.py"
tiga juta seratus dua puluh lima ribu tujuh ratus lima puluh

```

Picture 13.2 the output.

14. Creat a function **formatRupiah()** which accepts a positive integer and returns a string which is that number but in 'rupiah format'.

Example:

```
>>> formatRupiah(1500)
```

```
'Rp1.500'
```

```
>>> formatRupiah(2560000)
```

```
'Rp2.560.000'
```

#### - Program code

```

1 def formatRupiah(angka):
2 return "Rp " + "{:,}".format(angka).replace(",", ".")
3
4 # Contoh penggunaan
5 print(formatRupiah(1500)) # Rp 1.500
6 print(formatRupiah(2560000)) # Rp 2.560.000

```

Picture 14.1 the code

- **Practicum results**

```
PS D:\Semester 4\PrakAI_and_StrDat> & C:/Users/Acer/AppData/Local/Programs/Python/Python311/python.exe "d:/Semester 4/PrakAI_and_StrDat/Module
1/Question/14.py"
Rp 1.500
Rp 2.560.000
PS D:\Semester 4\PrakAI_and_StrDat>
```

Picture 14.2 the results