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

YEARS 2024/2025

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 > ...
def cetakSiku(x):
    for i in range(x):
        print("*"*(i+1))

cetakSiku(5)
```

Picture 1.1 *the code.*

   - **Practicum Result Screenshot**

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

Gambar 1.2 the output

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

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

   Presed

   @@@@@

   @ @

   @ @

   @@@@@

- **Program Code**

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

gambarlahPersegiEmpat(4,5)
```

Picture 2.1 the code.

- **Practikum result 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/2.py"
@@@@@
@   @
@   @
@@@@@
```

Picture 2.2 the output.

3. The following are two interrelated questions

   a. 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
def jumlahHurufVokal(s):
    vowels = 'aiueoAIUEO'
    total_letters = 0
    vowel_count = 0
    for char in s:
        if char.isalpha():
            total_letters += 1
            if char in vowels:
                vowel_count += 1
    return (total_letters, vowel_count)

def jumlahHurufKonsonan(s):
    vowels = 'aiueoAIUEO'
    total_letters = 0
    consonant_count = 0
    for char in s:
        if char.isalpha():
            total_letters += 1
            if char not in vowels:
                consonant_count += 1
    return (total_letters, consonant_count)

k = jumlahHurufVokal('Surakarta')
print(k)

v = jumlahHurufKonsonan('Surakarta')
print(v)
```

Picture 3.2 the code

- **Practikum 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 containingnumbers. The average has a formula

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

(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 > ...

```python
def rerata(b):
    total = sum(b)
    n = len(b)
    hasil = total / n
    return hasil

def variance(x):
    mean = rerata(x)
    total_squared_diff = sum((xi - mean) ** 2 for xi in x)
    var = total_squared_diff / len(x)
    return var

def stdev(x):
    var = variance(x)
    std_dev = var ** 0.5
    return std_dev


print(rerata([1, 2, 3, 4, 5]))
g = [3, 4, 5, 4, 3, 4, 5, 2, 2, 10, 11, 23]
print(rerata(g))
print(variance(g))
print(stdev(g))
```

Picture 4.1 the code

- **Practicum 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/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] #caugh 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**

```
from math import sqrt as sq

def apakahPrima(n):
    n = int(n)
    assert n >= 0
    primaKecil = [2, 3, 5, 7, 11]
    bukanPrKecil = [0, 1, 4, 6, 8, 9, 10]
    if n in primaKecil:
        return True
    elif n in bukanPrKecil:
        return False
    else:
        for i in range(2, int(sq(n)) + 1):
            if n % i == 0:
                return False
        return True

# Example usage:
print(apakahPrima(17))
print(apakahPrima(97))
print(apakahPrima(123))
```

Picture 5.1 the code

- **Practicum result 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/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**

```python
def apakahPrima(n):
    if n <= 1:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def cetakPrima():
    for i in range(2, 1001):
        if apakahPrima(i):
            print(i)

# Contoh penggunaan
cetakPrima()
```

Picture 6.1 the code

- **Practicum results screenshot**

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

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**

Module_1 > Question > 7.py > ...

```python
def faktorPrima(n):
    i = 2
    faktor = []
    while i * i <= n:
        if n % i:
            i += 1
        else:
            n //= i
            faktor.append(i)
    if n > 1:
        faktor.append(n)
    return tuple(faktor)

# Contoh penggunaan
print(faktorPrima(10))
print(faktorPrima(120))
print(faktorPrima(19))
```

Picture 7.1 the code

- **Practicum 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/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 > ...
def apakahTerkandung(a, b):
    return a in b

# Example usage:
h = 'do'
k = 'Indonesia tanah air beta'
print(apakahTerkandung(h, k))
print(apakahTerkandung('pusaka', k))
```

Picture 8.1 the code.

- **Practicum 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/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**

```python
for i in range(1, 101):
    if i % 3 == 0 and i % 5 == 0:
        print("Python UMS")
    elif i % 3 == 0:
        print("Python")
    elif i % 5 == 0:
        print("UMS")
    else:
        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 > ...

```python
import math

def selesaikanABC(a, b, c):
    determinant = b**2 - 4*a*c
    if determinant < 0:
        print("Determinannya negatif. Persamaan tidak mempunyai akar real.")
    else:
        root1 = (-b + math.sqrt(determinant)) / (2*a)
        root2 = (-b - math.sqrt(determinant)) / (2*a)
        return root1, root2

# Example usage:
print(selesaikanABC(1, 2, 3))
print(selesaikanABC(1, -3, 2))
```

Picture 10.1 the code.

- **Practicum results**

```
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/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 > ...

```python
def apakahKabisat(tahun):
    if (tahun % 4 == 0 and tahun % 100 != 0) or (tahun % 400 == 0):
        return True
    else:
        return False

# Contoh penggunaan
print(apakahKabisat(1896))
print(apakahKabisat(1897))
print(apakahKabisat(1900))
print(apakahKabisat(2000))
print(apakahKabisat(2004))
print(apakahKabisat(2100))
print(apakahKabisat(2400))
```

Picture 11.1 the code.

- **Practicum results**

```
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/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 > ...

```python
import random

def tebak_angka():
    angka_rahasia = random.randint(1, 100)
    tebakan = None
    jumlah_tebakan = 0

    print("Permainan tebak angka.")
    print("Saya menyimpan sebuah angka bulat antara 1 sampai 100. Coba tebak.")

    while tebakan != angka_rahasia:
        jumlah_tebakan += 1
        tebakan = int(input(f"Masukkan tebakan ke-{jumlah_tebakan}:> "))

        if tebakan < angka_rahasia:
            print("Itu terlalu kecil. Coba lagi.")
        elif tebakan > angka_rahasia:
            print("Itu terlalu besar. Coba lagi.")
        else:
            print("Ya. Anda benar")

if __name__ == "__main__":
    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. Creat 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 serratus dua puluh lima ribu tujuh ratus lima puluh'
    Limit the input to less than one billion. Extra credit: use recursion.

- **Program code**

```python
Module_1 > Question > 13.py > ...
def katakan(angka):
    satuan = ["", "satu", "dua", "tiga", "empat", "lima", "enam", "tujuh", "delapan", "sembilan", "sepuluh", "sebelas"]

    def terbilang(n):
        if n < 12:
            return satuan[n]
        elif n < 20:
            return terbilang(n - 10) + " belas"
        elif n < 100:
            return terbilang(n // 10) + " puluh" + ("" if n % 10 == 0 else " " + terbilang(n % 10))
        elif n < 200:
            return "seratus" + ("" if n % 100 == 0 else " " + terbilang(n % 100))
        elif n < 1000:
            return terbilang(n // 100) + " ratus" + ("" if n % 100 == 0 else " " + terbilang(n % 100))
        elif n < 2000:
            return "seribu" + ("" if n % 1000 == 0 else " " + terbilang(n % 1000))
        elif n < 1000000:
            return terbilang(n // 1000) + " ribu" + ("" if n % 1000 == 0 else " " + terbilang(n % 1000))
        elif n < 1000000000:
            return terbilang(n // 1000000) + " juta" + ("" if n % 1000000 == 0 else " " + terbilang(n % 1000000))
        else:
            return terbilang(n // 1000000000) + " milyar" + ("" if n % 1000000000 == 0 else " " + terbilang(n % 1000000000))

    return terbilang(angka)

# Contoh penggunaan
print(katakan(3125750))
```

Picture 13.1 the code.

- **Practicum results**

```
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/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**

```python
def formatRupiah(angka):
    return "Rp " + "{:,}".format(angka).replace(",", ".")

# Contoh penggunaan
print(formatRupiah(1500))      # Rp 1.500
print(formatRupiah(2560000))   # Rp 2.560.000
```

Picture 14.1 the code

- **Practicum results**

```
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/14.py"
Rp 1.500
Rp 2.560.000
PS D:\Semester 4\PrakAl_and_StrDat>
```

Picture 14.2 the resu;lts