

LAPORAN PRAKTIKUM
ALGORITMA DAN STRUKTUR DATA
MODUL 1 : TINJAUAN ULANG PYTHON



Disusun Oleh :
MHD. FARHAN LUBIS
L200220277
F

TEKNIK INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH SURAKARTA
TAHUN 2024

Daftar Isi

Daftar Isi.....	2
1.11 Soal – Soal Mahasiswa.....	3
2. Buatlah sebuah fungsi yang menerima dua integer positif, yang akan menggambar bentuk persegi empat.	3
• Kode Program.....	3
• Screenshot hasil praktikum.....	3
6. Buatlah suatu program yang mencetak semua bilangan prima dari 2 sampai 1000.	4
• Kode Program.....	4
• Screenshot hasil praktikum.....	4
7. Buatlah suatu program yang menerima bilangan bulat positif dan memberikan faktorisasi prima-nya.	5
• Kode Program.....	5
• Screenshot hasil praktikum.....	6
8. Buat suatu fungsi apakahTerkandung(a,b) yang menerima dua string a dan b, lalu menentukan apakah string a terkandung dalam string b.....	6
• Kode Program.....	6
• Screenshot hasil praktikum.....	7
9. Buat program untuk mencetak angka dari 1 sampai 100.....	7
• Kode Program.....	8
• Screenshot hasil praktikum.....	8
10. Buat modifikasi pada Contoh 1.4, agar bisa menangkap kasus di mana determinannya kurang dari nol.	9
• Kode Program.....	9
• Screenshot hasil praktikum.....	10
11. Buat suatu fungsi apakahKabisat() yang menerima suatu angka (tahun). Jika tahun itu kabisat, kembalikan True. Jika bukan kabisat, kembalikan False.....	10
• Kode Program.....	10
• Screenshot hasil praktikum.....	11
12. Program permainan tebak angka.	11
• Kode Program.....	12
• Screenshot hasil praktikum.....	13
13. Buat suatu fungsi katakan() yang menerima bilangan bulat positif dan mengembalikan suatu string yang merupakan pengucapan angka itu dalam Bahasa Indonesia.....	13
• Kode Program.....	13
• Screenshot hasil praktikum.....	14
14. Buat suatu fungsi formatRupiah() yang menerima suatu bilangan bulat positif dan mengembalikan suatu string yang merupakan bilangan itu tapi dengan ‘format rupiah’.	14
• Kode Program.....	15
• Screenshot hasil praktikum.....	15

1.11 Soal – Soal Mahasiswa

2. Buatlah sebuah fungsi yang menerima dua integer positif, yang akan menggambar bentuk persegi empat.

Contoh pemanggilan:

```
>>> gambarlahPersegiEmpat(4,5) #tombol <enter> dipencet
```

```
@@@@@
@  @
@  @
@@@@@
```

Untuk memudahkan pengerjaan, kamu cukup melengkapi snippet kode berikut (kamu bisa membuat yang sama sekali lain):

- **Kode Program**

```
def gambarlah_persegi_empat(tinggi, lebar):
    for i in range(tinggi):
        for j in range(lebar):
            if i == 0 or i == tinggi - 1 or j == 0 or j ==
lebar - 1:
                print("@", end="")
            else:
                print(" ", end="")
        print()

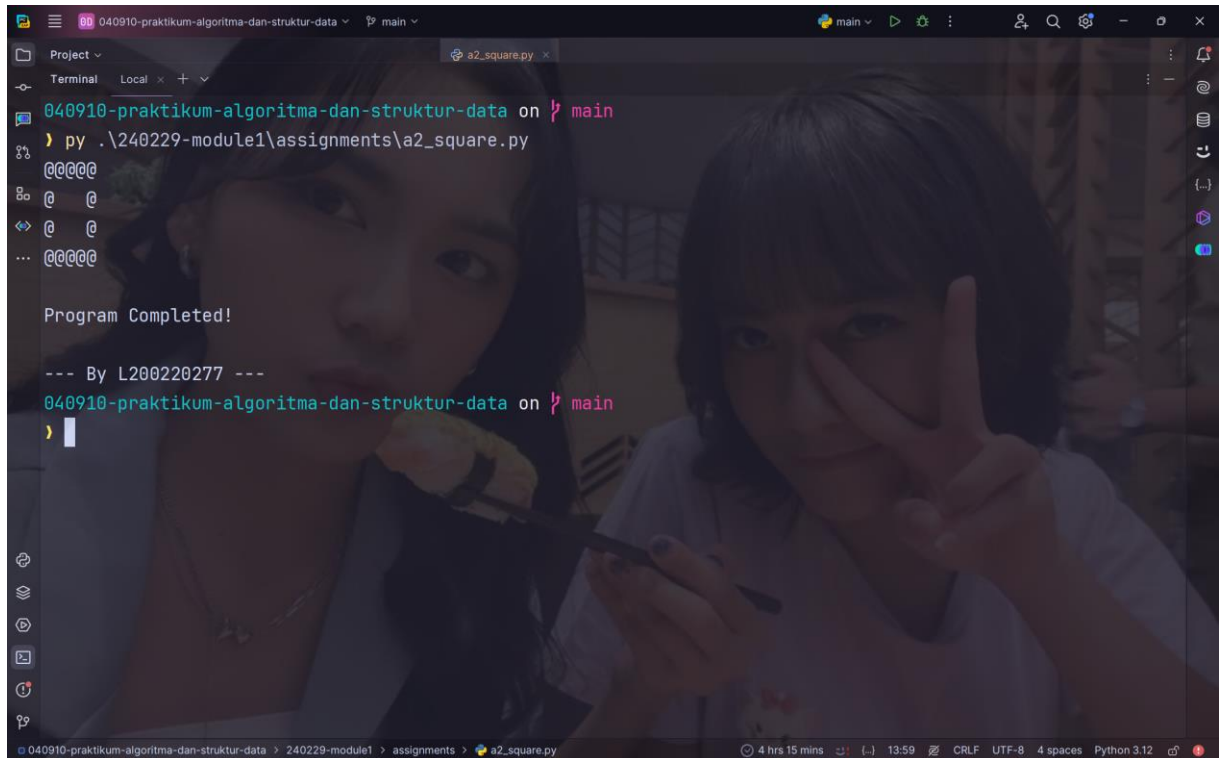
gambarlah_persegi_empat(4, 5)
print("\nProgram Completed!\n\n--- By L200220277 ---")
```

Kode 1.2 membuat persegi menggunakan @ sebagai batas

PENJELASAN:

function gambarlah_persegi_empat bertujuan menggambar sebuah persegi empat menggunakan simbol "@" dengan tinggi dan lebar yang ditentukan. function ini menggunakan perulangan bersarang untuk mengulang setiap baris dan kolom persegi. Untuk setiap posisinya akan apakah posisi saat itu berada pada batas persegi baik baris pertama atau terakhir dan kolom pertama atau terakhir. Apabila kondisi ini benar, akan mencetak "@" untuk membuat batas. Tetapi jika tidak, akan mencetak spasi untuk mengisi bagian dalam persegi. Setelah itu akan berpindah ke baris berikutnya untuk mulai membuat baris berikutnya dari persegi.

- **Screenshot hasil praktikum**



Gambar 1.2 output a2_square.py

6. Buatlah suatu program yang mencetak semua bilangan prima dari 2 sampai 1000.

Manfaatkan fungsi apakahPrima() pada nomer di atas.

- **Kode Program**

```
from a5_isprime import apakah_prima

def cek_prima():
    angka_prima = []
    for i in range(2, 1000):
        if apakah_prima(i):
            angka_prima.append(i)
    return angka_prima

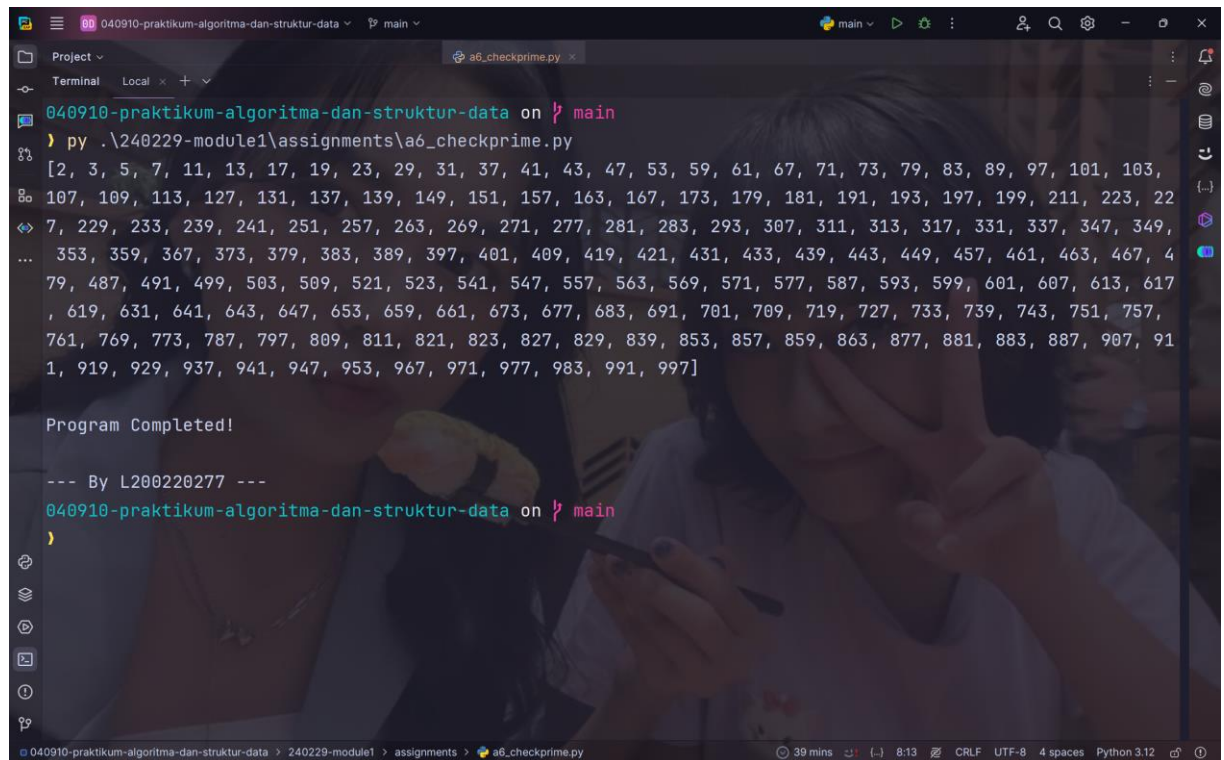
print(cek_prima())
print("\nProgram Completed!\n\n--- By L200220277 ---")
```

Kode 1.6 mengecek angka prima pada range tertentu

PENJELASAN:

Sesuai dengan instruksi soal, dalam function ini akan memakai function dari file lain yaitu a5_isprime yang memiliki function apakah_prima. Setelah mendefinisikan function bernama cek_prima, terdapat inisialisasi variable berupa list kosong bernama angka_prima. Kemudian akan dilakukan perulangan angka dari 2 sampai 999, dan untuk setiap angkanya dicek apakah angka itu prima menggunakan function apakah_prima. Jika bernilai True, maka angka itu akan ditambahkan ke dalam list angka_prima. Terakhir, akan dikembalikan list angka prima yang ditemukan.

- **Screenshot hasil praktikum**



```
040910-praktikum-algoritma-dan-struktur-data on main
py .\240229-module1\assignments\a6_checkprime.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, 22
7, 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, 4
79, 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, 91
1, 919, 929, 937, 941, 947, 953, 967, 971, 977, 983, 991, 997]

Program Completed!

--- By L200220277 ---
040910-praktikum-algoritma-dan-struktur-data on main
)
```

Gambar 1.6 output a6_checkprime.py

7. Buatlah suatu program yang menerima bilangan bulat positif dan memberikan faktorisasi prima-nya.

Faktorisasi prima adalah pemfaktoran suatu bilangan bulat ke dalam bilangan-bilangan prima yang menjadi konstituennya. Contoh:

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

- **Kode Program**

```
def faktor_prima(n):
    faktor = []
    pembagi = 2
    while n > 1:
        while n % pembagi == 0:
            faktor.append(pembagi)
            n //= pembagi
        pembagi += 1
    return tuple(faktor)

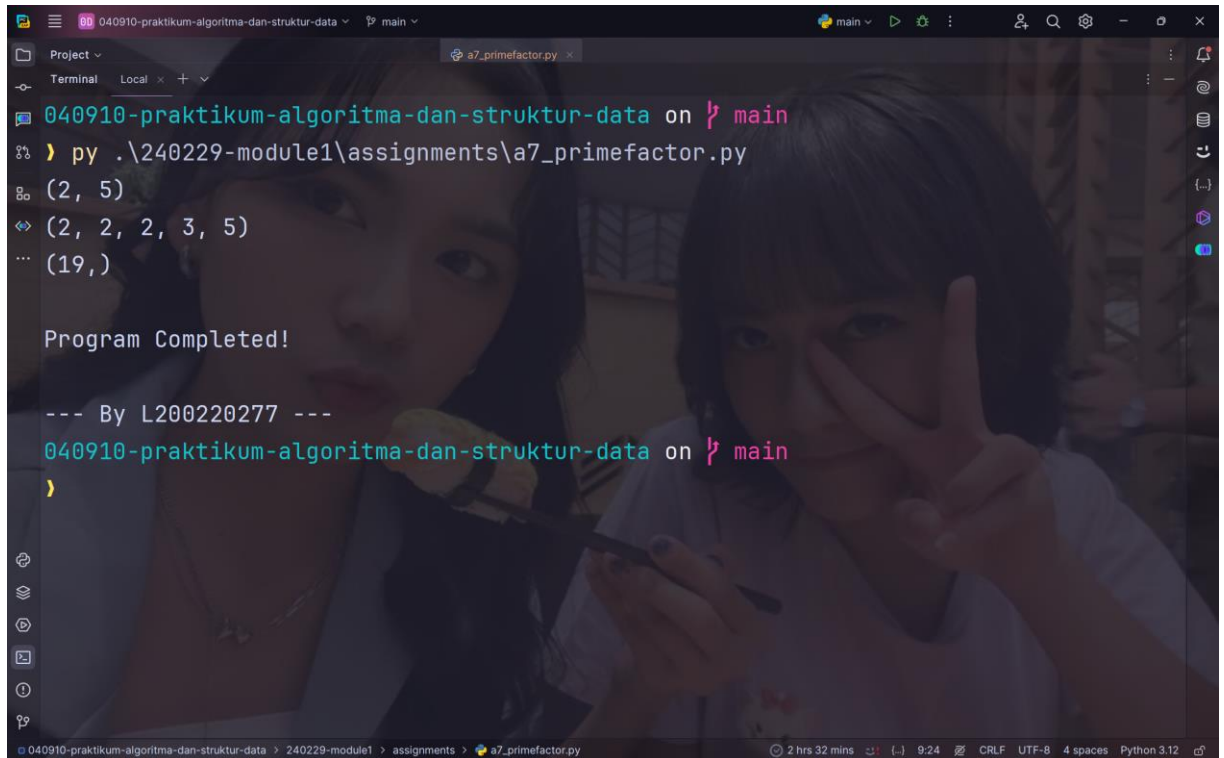
print(faktor_prima(10))
print(faktor_prima(120))
print(faktor_prima(19))
print("\nProgram Completed!\n\n--- By L200220277 ---")
```

Kode 1.7 mencari faktor angka prima dari sebuah angka

PENJELASAN:

Function ini digunakan untuk mencari faktorisasi prima pada sebuah angka. Function ini menginisialisasi variable faktor yang mengandung sebuah *list* kosong dan menginisialisasi variable pembagi dengan angka 2. Kemudian akan melakukan perulangan while yang terus berulang selama n lebih besar dari 1. Di dalam perulangan ini terdapat perulangan while lainnya yang berjalan selama n masih habis dibagi oleh pembagi. Jika kondisi ini terpenuhi, pembagi ditambahkan ke list faktor, dan n dibagi dengan pembagi kemudian nilai pembagi ditambah 1 untuk iterasi berikutnya. Terakhir, akan mengembalikan faktor prima dari n sebagai sebuah tuple.

- **Screenshot hasil praktikum**



```

040910-praktikum-algoritma-dan-struktur-data on main
py .\240229-module1\assignments\a7_primefactor.py
(2, 5)
(2, 2, 2, 3, 5)
... (19,)

Program Completed!

--- By L200220277 ---
040910-praktikum-algoritma-dan-struktur-data on main
)
  
```

Gambar 1.7 output a7_primefactor.py

8. **Buat suatu fungsi apakahTerkandung(a,b) yang menerima dua string a dan b, lalu menentukan apakah string a terkandung dalam string b.**

Eksekusinya seperti ini:

```

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

- **Kode Program**

```

def apakah_terkandung(a,b): return a.lower() in b.lower()

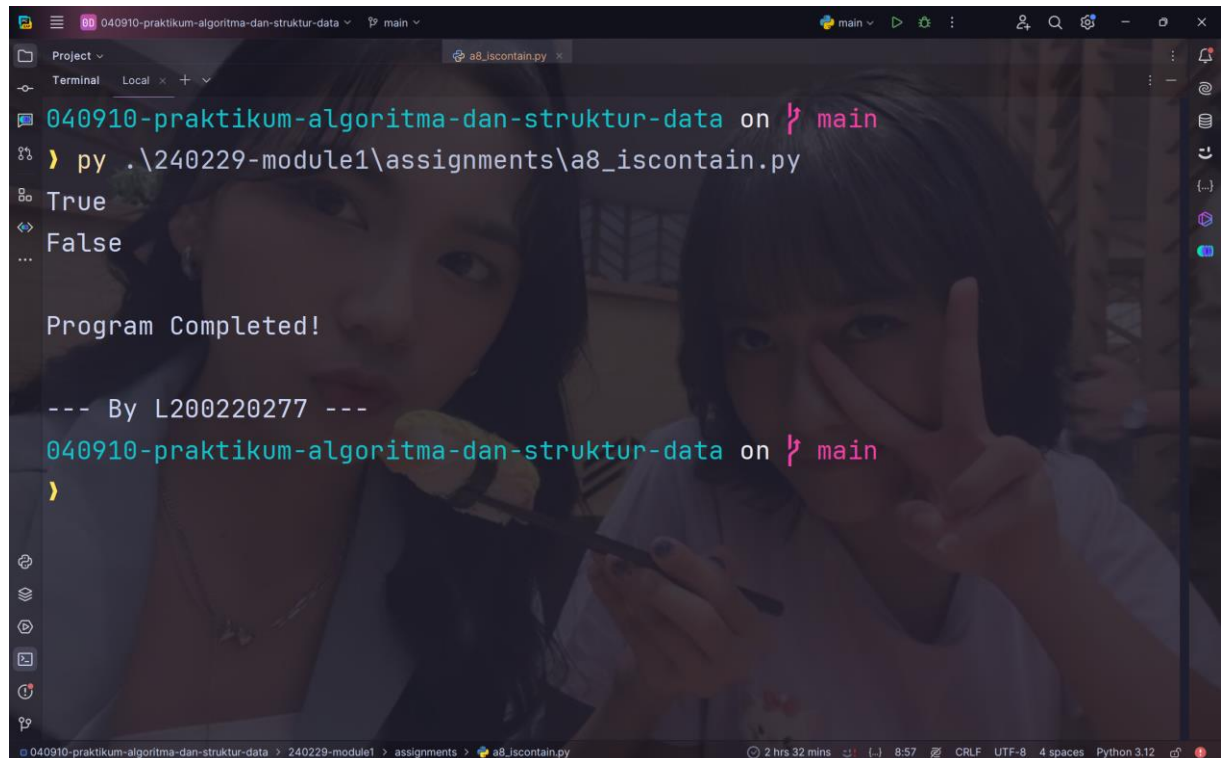
k = 'Indonesia tanah air beta'
print(apakah_terkandung('Do', k))
print(apakah_terkandung('pusaka', k))
print("\nProgram Completed!\n\n--- By L200220277 ---")
  
```

Kode 1.8 mengecek keberadaan suatu string pada string lain

PENJELASAN:

function bernama apakah_terkandung ini menerima string a dan b sebagai argumen. Kemudian memeriksa apakah versi huruf kecil dari string a terdapat di dalam versi huruf kecil dari string b, dan mengembalikan nilai boolean sebagai hasilnya.

- **Screenshot hasil praktikum**



```
040910-praktikum-algoritma-dan-struktur-data on main
> py .\240229-module1\assignments\a8_iscontain.py
True
False
...
Program Completed!
--- By L200220277 ---
040910-praktikum-algoritma-dan-struktur-data on main
>
```

Gambar 1.8 output a8_iscontain.py

9. Buat program untuk mencetak angka dari 1 sampai 100.

Kalau angkanya pas kelipatan 3, cetak 'Python'. Kalau pas kelipatan 5, cetak 'UMS'. Kalau pas kelipatan 3 sekaligus kelipatan 5, cetak 'Python UMS'. Jadi hasilnya:

```
1
2
Python
4
UMS
Python
7
8
Python
UMS
11
Python
13
14
```


Python UMS

16

17

...

- **Kode Program**

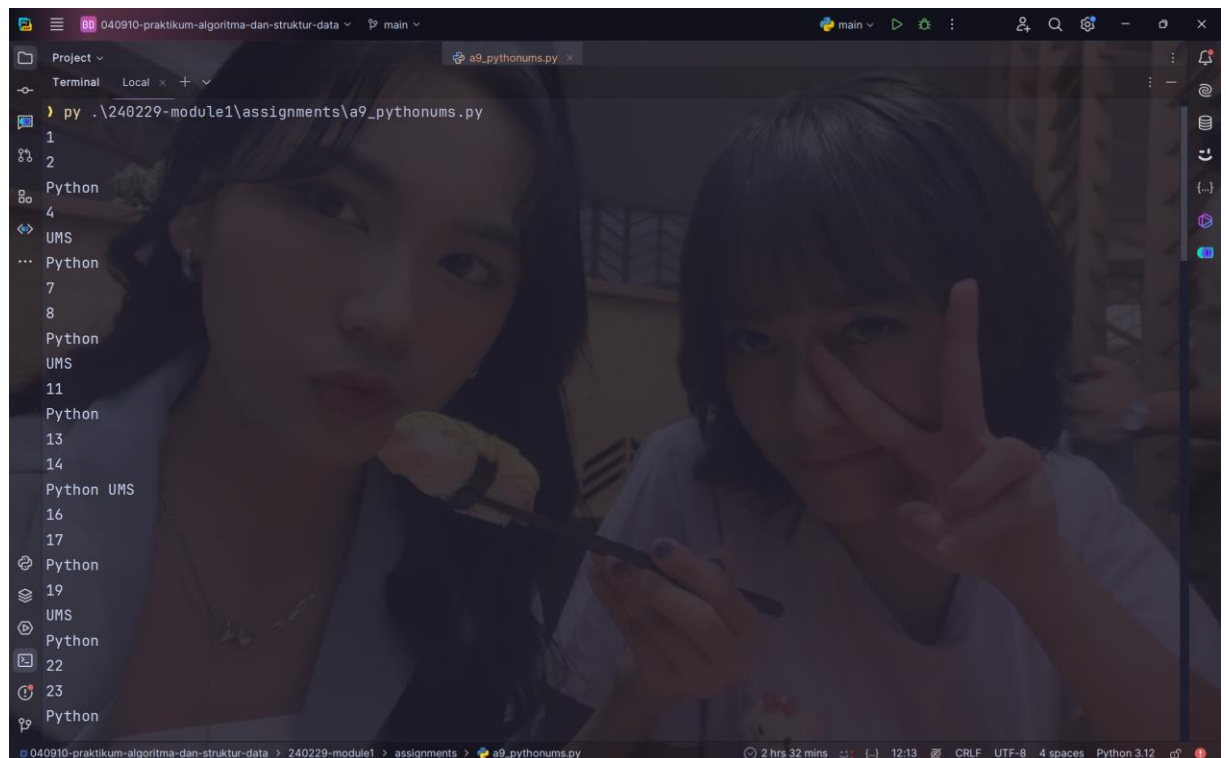
```
def python_ums():  
    for i in range(1,100):  
        if i % 3 == 0 and i % 5 == 0:  
            print("Python UMS")  
        elif i % 3 == 0:  
            print("Python")  
        elif i % 5 == 0:  
            print("UMS")  
        else:  
            print(i)  
  
python_ums()  
print("\nProgram Completed!\n\n--- By L200220277 ---")
```

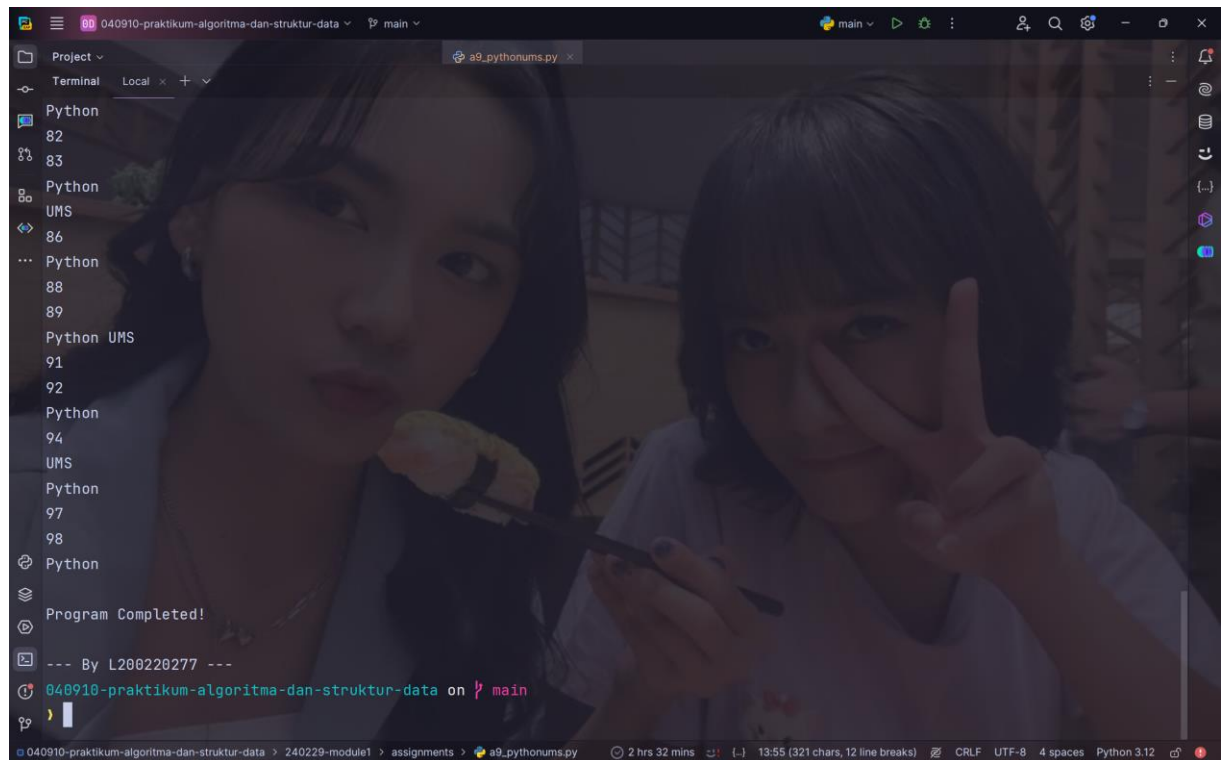
Kode 1.9 program cek python ums

PENJELASAN:

function bernama python_ums akan mengulang angka dari 1 hingga 99 dimana untuk setiap angkanya akan diperiksa apakah angka tersebut habis dibagi 3 dan 5, dan mencetak "Python UMS" jika benar. Jika angka tersebut hanya habis dibagi 3, maka akan mencetak "Python". Jika hanya habis dibagi 5, maka akan dicetak "UMS" dan Jika tidak ada satu pun dari kondisi sebelumnya terpenuhi, maka akan mencetak angka itu sendiri.

- **Screenshot hasil praktikum**





Gambar 1.9 output a9_pythonums.py

10. Buat modifikasi pada Contoh 1.4, agar bisa menangkap kasus di mana determinannya kurang dari nol.

Jika ini terjadi, tampilkan peringatan di layar seperti ini:

```
>>> selesaikanABC(1,2,3)
Determinannya negatif. Persamaan tidak mempunyai akar real.
>>>
```

- **Kode Program**

```
from math import sqrt

def selesaikan_abc(a,b,c):
    a = float(a)
    b = float(b)
    c = float(c)
    D = b**2 - 4*a*c
    if D < 0:
        return "Determinannya negatif. Persamaan tidak
mempunyai akar real."
    x1 = (-b + sqrt(D))/(2*a)
    x2 = (-b - sqrt(D))/(2*a)
    hasil = (x1, x2)
    return hasil

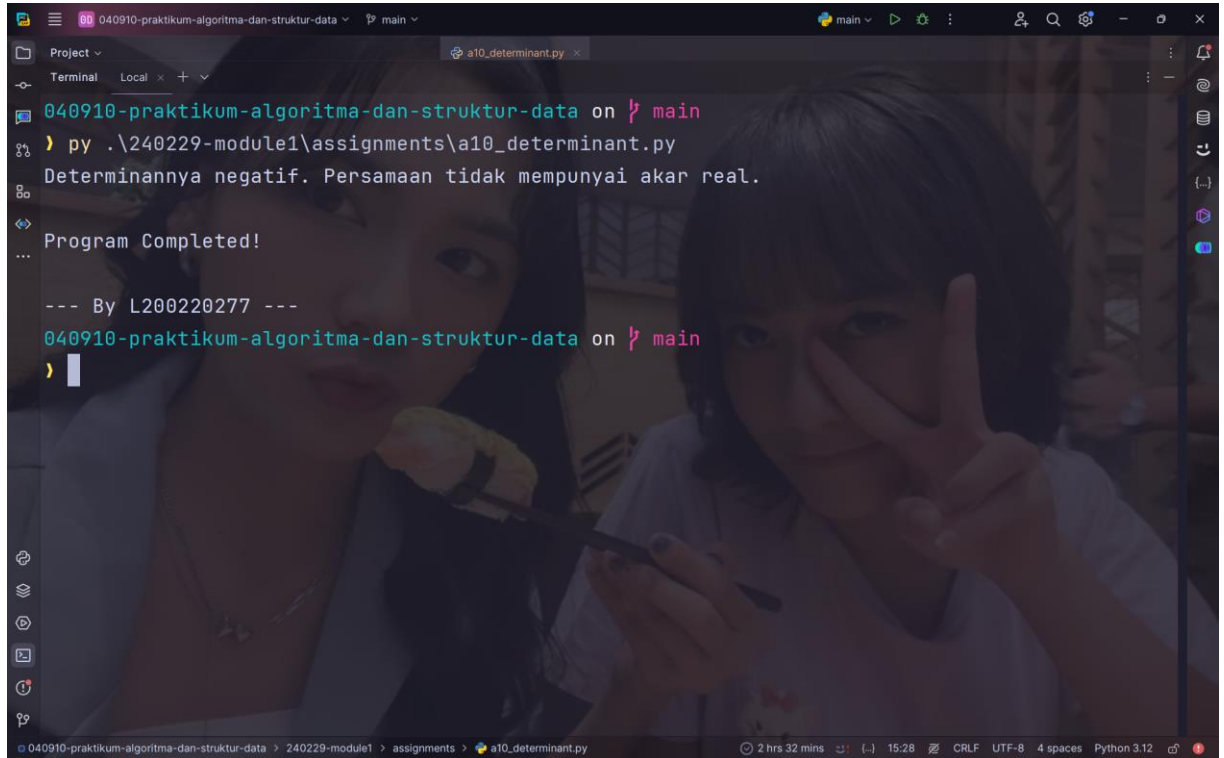
print(selesaikan_abc(1, 2, 3))
print("\nProgram Completed!\n\n--- By L200220277 ---")
```

Kode 1.10 mengecek determinan dan menentukan akarnya

PENJELASAN:

function ini memiliki tiga parameter a, b, dan c yang mewakili koefisien persamaan kuadrat. dalam function ini akan dihitung diskriminan D nya serta diperiksa apakah diskriminan tersebut negatif, jika iya akan mengembalikan pesan yang menunjukkan tidak adanya akar nyata. Jika diskriminan non-negatif, akan dihitung dua akarnya menggunakan rumus kuadrat dan mengembalikannya sebagai *tuple*.

- **Screenshot hasil praktikum**



```
040910-praktikum-algoritma-dan-struktur-data on main
> py .\240229-module1\assignments\a10_determinant.py
Determinannya negatif. Persamaan tidak mempunyai akar real.
Program Completed!
--- By L200220277 ---
040910-praktikum-algoritma-dan-struktur-data on main
>
```

Gambar 1.10 output a10_determinant

11. Buat suatu fungsi apakahKabisat() yang menerima suatu angka (tahun). Jika tahun itu kabisat, kembalikan True. Jika bukan kabisat, kembalikan False.

Tahun kabisat – tahun yang memiliki tanggal 29 Februari – adalah tahun yang habis dibagi 4, kecuali dia habis dibagi 100 (maka dia bukan tahun kabisat). Tapi kalau dia habis dibagi 400, dia adalah tahun kabisat (meski habis dibagi 100).

Berikut ini adalah beberapa contoh:

- 1896 tahun kabisat (habis dibagi 4)
- 1897 bukan tahun kabisat (sudah jelas)
- 1900 bukan tahun kabisat (meski habis dibagi 4, tapi habis dibagi 100, dan tidak habis dibagi 400)
- 2000 tahun kabisat (habis dibagi 400)
- 2004, 2008, 2012, 2016, ..., 2096 tahun kabisat
- 2100, 2200, 2300 bukan tahun kabisat
- 2400 tahun kabisat

- **Kode Program**

```
def apakah_kabisat(tahun):
    if (tahun % 4 == 0 and tahun % 100 != 0) or (tahun %
400 == 0):
        return True
```

```

    return False

print(apakah_kabisat(1896))
print(apakah_kabisat(1897))
print(apakah_kabisat(1900))
print(apakah_kabisat(2000))
print(apakah_kabisat(2004))
print(apakah_kabisat(2008))
print(apakah_kabisat(2012))
print(apakah_kabisat(2016))
print(apakah_kabisat(2096))
print(apakah_kabisat(2100))
print(apakah_kabisat(2200))
print(apakah_kabisat(2300))
print(apakah_kabisat(2400))
print("\nProgram Completed!\n\n--- By L200220277 ---")

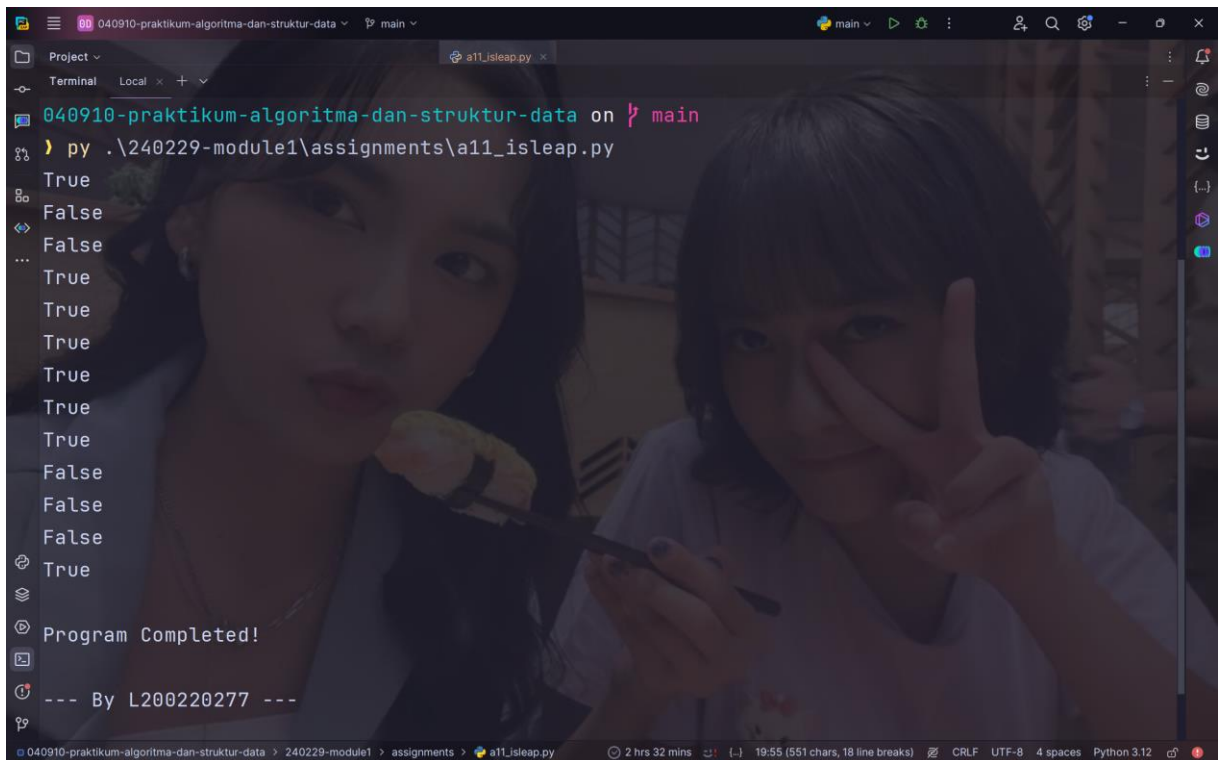
```

Kode 1.11 mengecek tahun kabisat

PENJELASAN:

function `apakah_kabisat` mengambil sebuah tahun sebagai argumen dan memeriksa apakah tahun tersebut adalah tahun kabisat. Function ini menentukan apakah tahun tersebut merupakan tahun kabisat dengan memeriksa apakah tahun tersebut habis dibagi 4 dan tidak habis dibagi 100, atau jika habis dibagi 400. Jika salah satu dari kondisi ini terpenuhi, akan dikembalikan nilai `True`, yang menunjukkan bahwa tahun tersebut adalah tahun kabisat; jika tidak, akan mengembalikan nilai `False`.

- **Screenshot hasil praktikum**



Gambar 1.11 output `a11_isleap.py`

12. Program permainan tebak angka.

Buat program yang alurnya secara global seperti ini:

- Komputer membangkitkan bilangan bulat random antara 1 sampai 100. Nilainya disimpan di suatu variabel dan tidak ditampilkan ke pengguna.
- Pengguna diminta menebak angka itu, diinputkan lewat keyboard.
- Jika angka yang diinputkan terlalu kecil atau terlalu besar, pengguna mendapatkan umpan balik dari komputer ("Angka itu terlalu kecil. Coba lagi")
- Proses diulangi sampai angka itu tertebak atau sampai sekian tebakan meleset

Ketika programnya dilarikan, prosesnya kurang lebih seperti di bawah ini
Permainan tebak angka.

Saya menyimpan sebuah angka bulat antara 1 sampai 100. Coba tebak.

Masukkan tebakan ke-1:> 50

Itu terlalu kecil. Coba lagi.

Masukkan tebakan ke-2:> 75

Itu terlalu besar. Coba lagi.

Masukkan tebakan ke-3:> 58

Ya. Anda benar

- **Kode Program**

```
from random import randint

def tebak_angka():
    angka_acak = randint(1, 100)
    tebakan = 0
    print("Permainan tebak angka.")
    print("Saya menyimpan sebuah angka bulat antara 1
sampai 100. Coba tebak.")
    while True:
        tebakan += 1
        angka_tebakan = int(input(f"Masukkan tebakan ke-
{tebakan}> "))
        if angka_tebakan < angka_acak:
            print("Itu terlalu kecil. Coba lagi.")
        elif angka_tebakan > angka_acak:
            print("Itu terlalu besar. Coba lagi.")
        else:
            print(f"Ya. Anda benar!")
            break

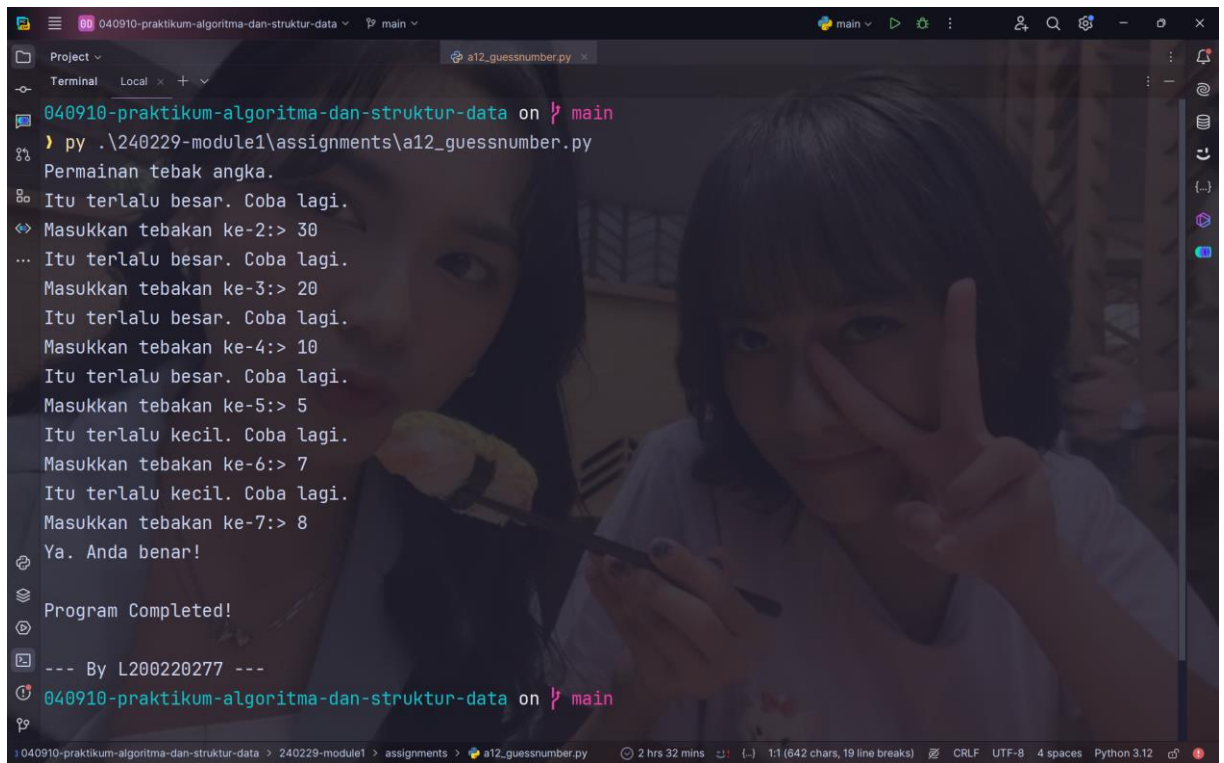
tebak_angka()
print("\nProgram Completed!\n\n--- By L200220277 ---")
```

Kode 1.12 *permainan tebak angka*

PENJELASAN:

function `tebak_angka` akan menghasilkan angka acak antara 1 dan 100 untuk ditebak oleh user. Function ini dimulai dengan user memasukkan tebakannya, dan program ini memberikan petunjuk jika tebakan tersebut terlalu kecil atau terlalu besar. Permainan berlanjut sampai user menebak dengan benar angka yang dihasilkan secara acak, sehingga program akan menampilkan pesan sukses dan mengakhiri permainan.

- **Screenshot hasil praktikum**



```

040910-praktikum-algoritma-dan-struktur-data on main
> py .\240229-module1\assignments\al2_guessnumber.py
Permainan tebak angka.
Itu terlalu besar. Coba lagi.
Masukkan tebakan ke-2:> 30
... Itu terlalu besar. Coba lagi.
Masukkan tebakan ke-3:> 20
Itu terlalu besar. Coba lagi.
Masukkan tebakan ke-4:> 10
Itu terlalu besar. Coba lagi.
Masukkan tebakan ke-5:> 5
Itu terlalu kecil. Coba lagi.
Masukkan tebakan ke-6:> 7
Itu terlalu kecil. Coba lagi.
Masukkan tebakan ke-7:> 8
Ya. Anda benar!
Program Completed!
--- By L200220277 ---
040910-praktikum-algoritma-dan-struktur-data on main

```

Gambar 1.12 output a12_guessnumber.py

13. Buat suatu fungsi katakan() yang menerima bilangan bulat positif dan mengembalikan suatu string yang merupakan pengucapan angka itu dalam Bahasa Indonesia.

Contoh:

```
>>> katakan(3125750)
'Tiga juta seratus dua puluh lima ribu tujuh ratus lima puluh'
```

Batasi inputnya agar lebih kecil dari satu milyar. Extra credit: gunakan rekursi.

- **Kode Program**

```

def katakan(n):
    if n == 0:
        return "nol"
    elif n < 10:
        return ['satu', 'dua', 'tiga', 'empat', 'lima',
'enam', 'tujuh', 'delapan', 'sembilan'][n-1]
    elif n < 20:
        return "belas " + katakan(n % 10)
    elif n < 100:
        return ['dua puluh', 'tiga puluh', 'empat puluh',
'lima puluh', 'enam puluh', 'tujuh puluh', 'delapan
puluh', 'sembilan puluh'][n // 10 - 2] + (" " + katakan(n
% 10) if n % 10 != 0 else "")
    elif n < 1000:
        return katakan(n // 100) + " ratus" + (" " +
katakan(n % 100) if n % 100 != 0 else "")
    elif n < 1000000:
        return katakan(n // 1000) + " ribu" + (" " +
katakan(n % 1000) if n % 1000 != 0 else "")

```

```

elif n < 1000000000:
    return katakan(n // 1000000) + " juta" + (" " +
katakan(n % 1000000) if n % 1000000 != 0 else "")
else:
    return "Angka terlalu besar"

print(katakan(3125750))
print("\nProgram Completed!\n\n--- By L200220277 ---")

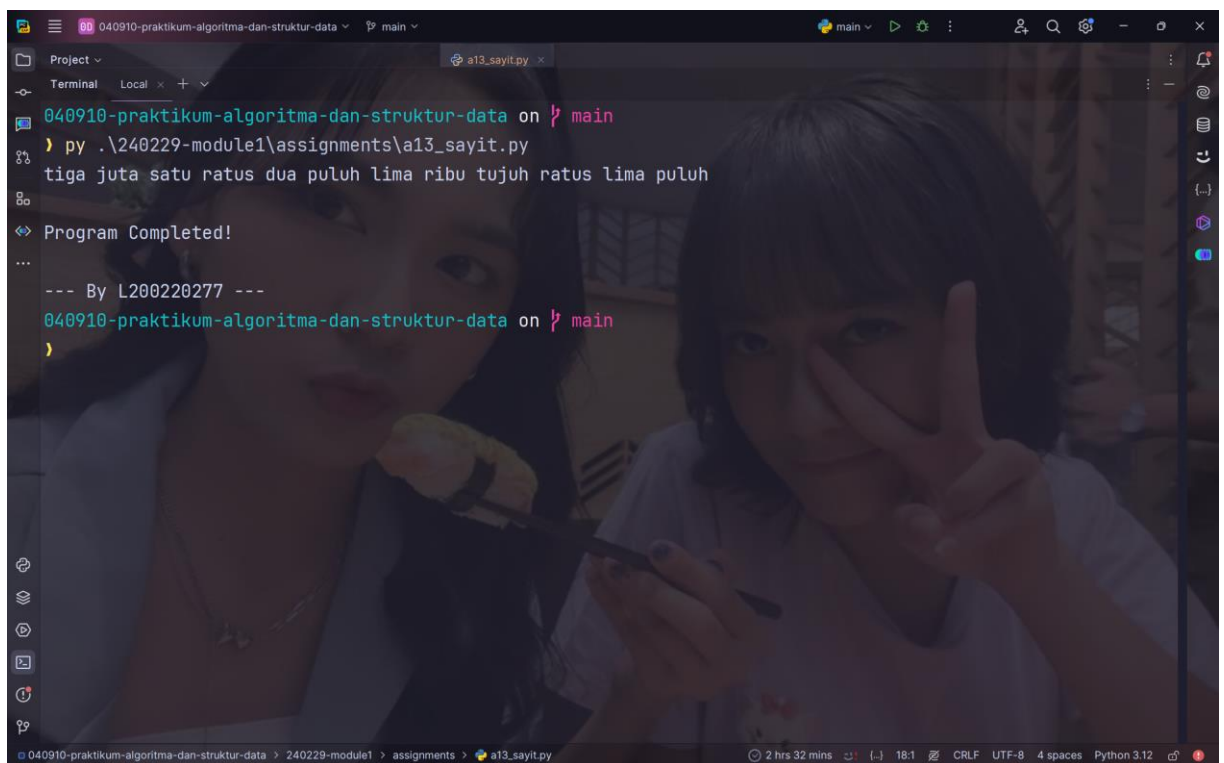
```

Kode 1.13 mengkonversi angka menjadi kata representasi

PENJELASAN:

function katakan ini mengubah bilangan bulat n yang diberikan menjadi representasi kata dalam bahasa Indonesia dengan memecah angka tersebut secara rekursif menjadi satuan, puluhan, ratusan, ribuan, jutaan, dan milyaran, serta mengubah setiap bagiannya menjadi kata. Fungsi ini berisi kondisi yang menangani rentang angka yang berbeda untuk membuat representasi kata. Jika angka *input* melebihi satu miliar, function ini mengembalikan "Angka terlalu besar" untuk memberitahu bahwa angka tersebut terlalu besar untuk dikonversi.

- **Screenshot hasil praktikum**



Gambar 1.13 output a13_sayit.py

14. Buat suatu fungsi formatRupiah() yang menerima suatu bilangan bulat positif dan mengembalikan suatu string yang merupakan bilangan itu tapi dengan 'format rupiah'.

Contoh:

```

>>> formatRupiah(1500)
'Rp 1.500'
>>> formatRupiah(2560000)

```


'Rp 2.560.000'

- **Kode Program**

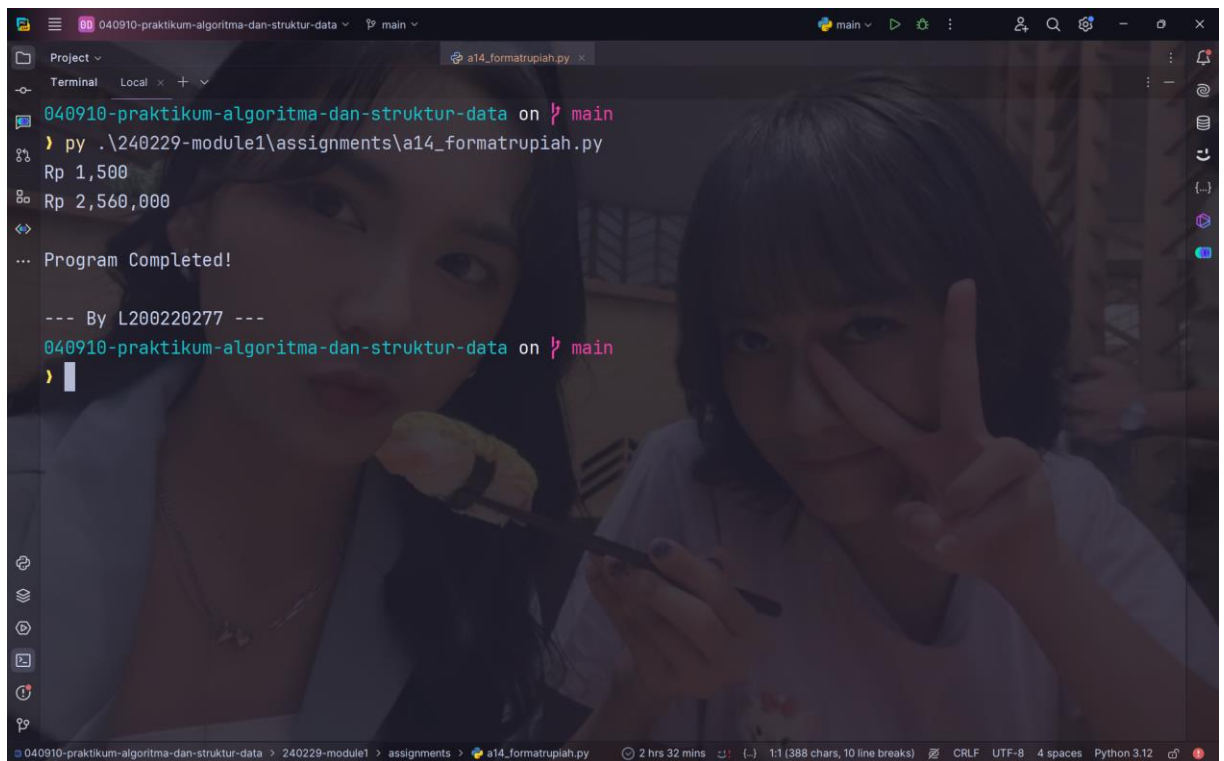
```
def format_rupiah(nominal):  
    if type(nominal) not in [int, float]:  
        return "Harus bilangan bulat atau float!"  
    elif nominal < 0:  
        return "bilangan bulat harus positif!"  
    formatted_nominal = "{:,.0f}".format(nominal)  
    return f'Rp {formatted_nominal}'  
  
print(format_rupiah(1500))  
print(format_rupiah(2560000))  
print("\nProgram Completed!\n\n--- By L200220277 ---")
```

Kode 1.14 memformat bilangan bulat menjadi rupiah

PENJELASAN:

function `format_rupiah` akan mengambil `nominal` angka sebagai argumen dan memformatnya menjadi mata uang Rupiah. pertama-tama akan diperiksa apakah inputnya adalah angka yang valid, dan jika tidak, akan mengembalikan pesan kesalahan. Kemudian kode ini juga akan memeriksa apakah angkanya negatif, dan jika iya, kode ini akan mengembalikan pesan bahwa angkanya harus positif. Jika inputnya valid dan positif, angka tersebut akan diformat menggunakan penentu `format "{:,.0f}"` untuk menambahkan koma sebagai pemisah ribuan dan menghapus bagian desimalnya. Terakhir, akan mengembalikan string yang telah diformat dengan simbol mata uang "Rp" di bagian awal.

- **Screenshot hasil praktikum**



Gambar 1.14 output `a14_formatrupiah.py`