

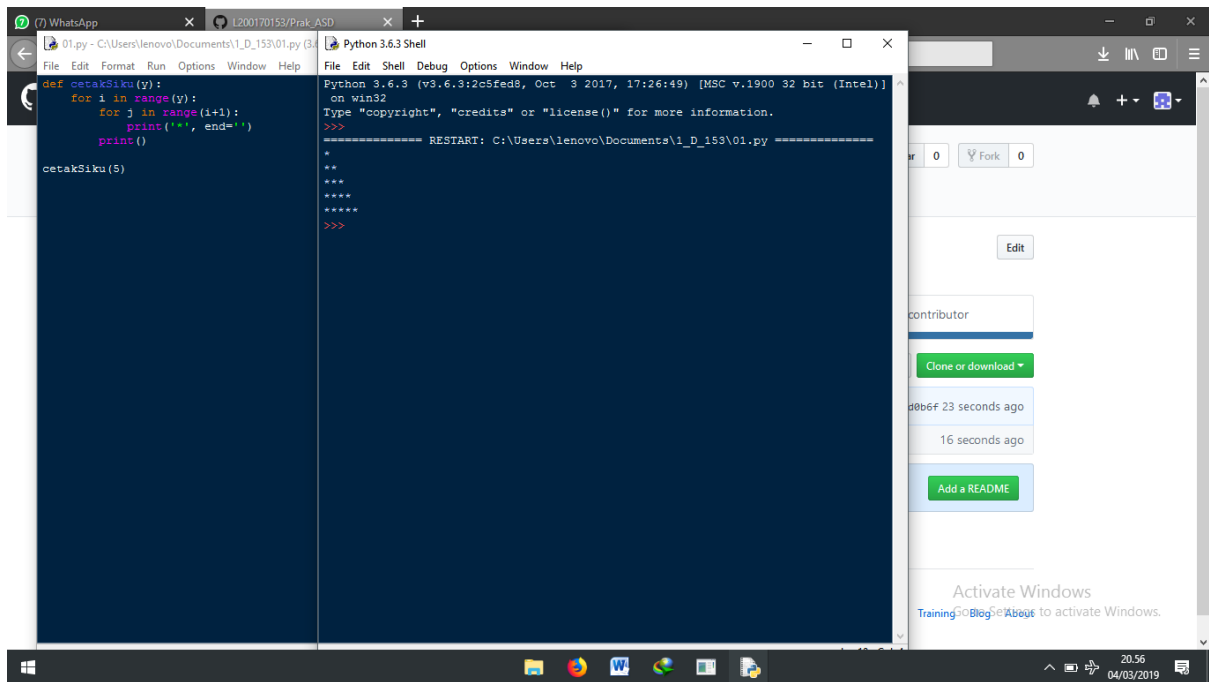
Ivanovitz A.A.R

L200170153

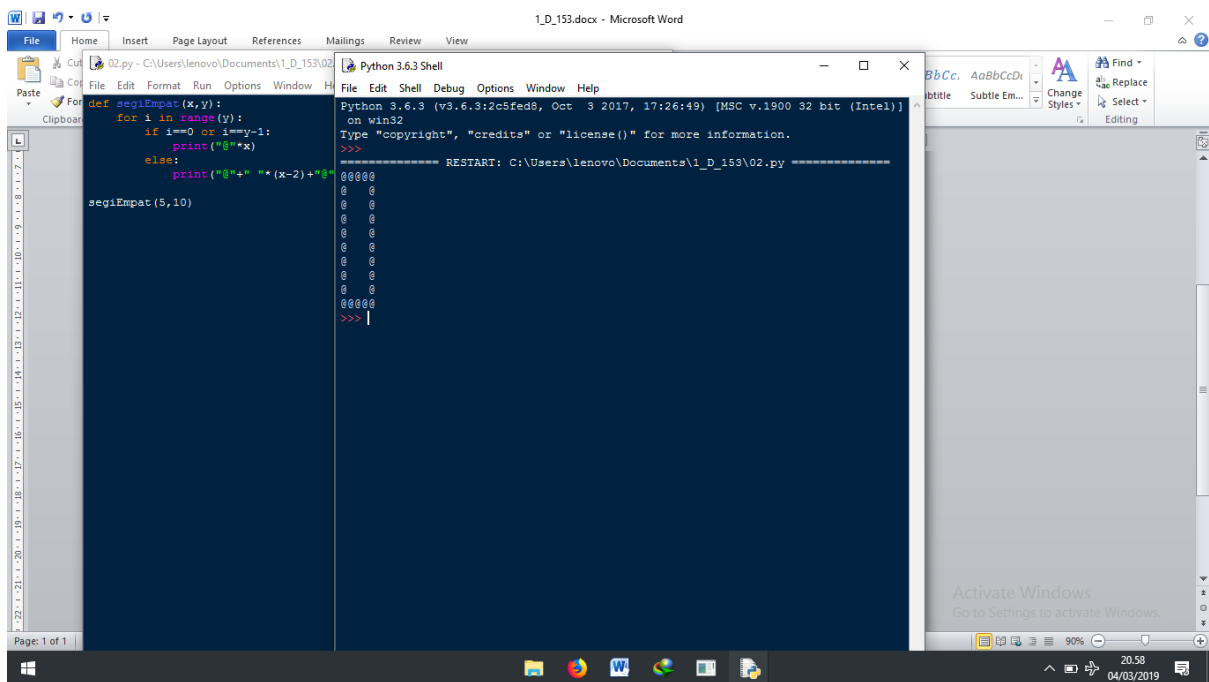
Kelas D

Modul 1

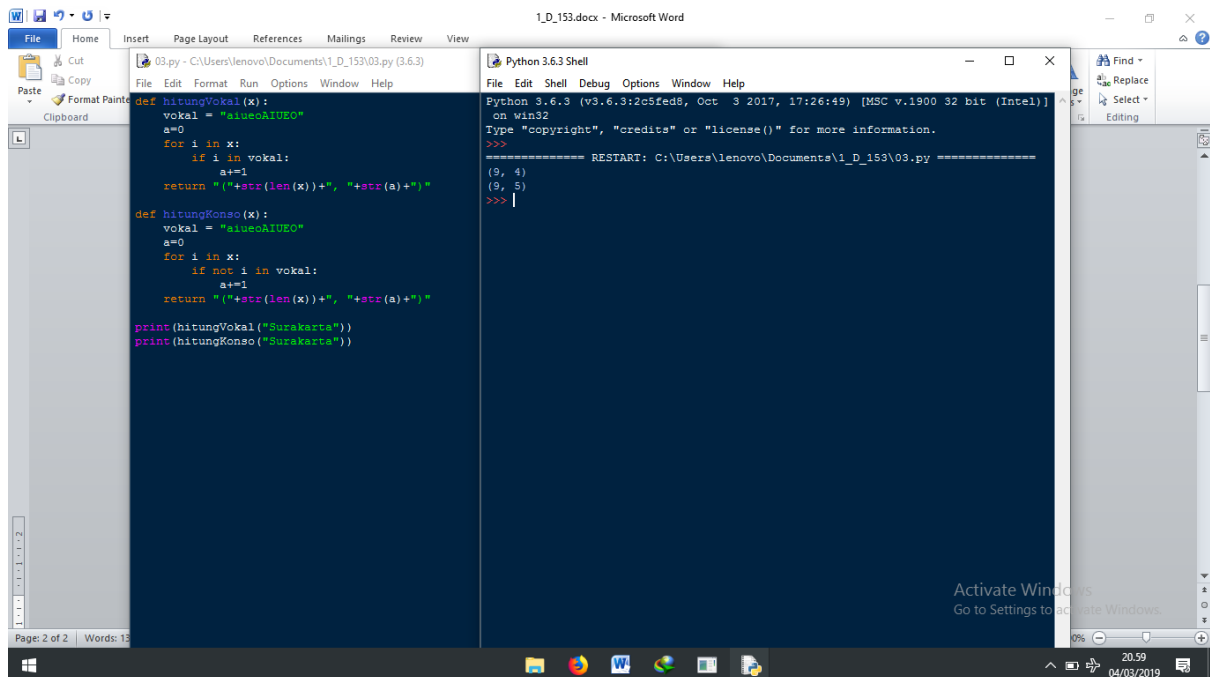
Nomor 1



Nomor 2



Nomor 3



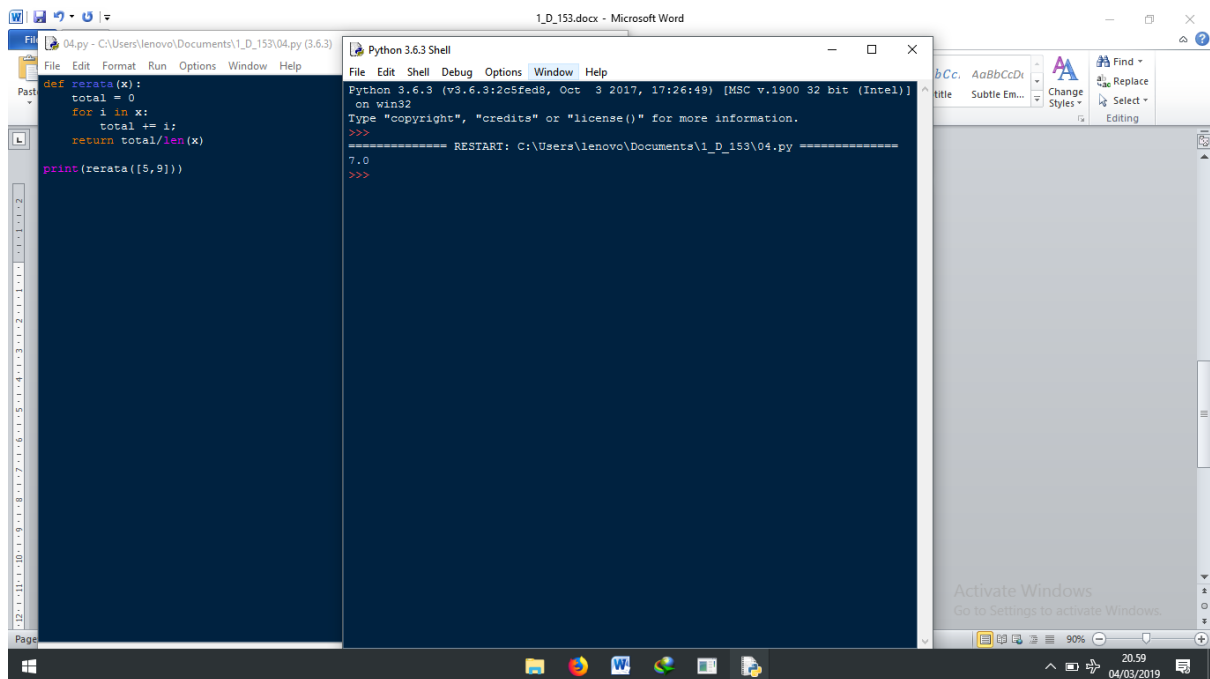
1_D_153.docx - Microsoft Word

```
def hitungVokal(x):  
    vokal = "aiueoAIUEO"  
    a=0  
    for i in x:  
        if i in vokal:  
            a+=1  
    return "("+str(len(x))+", "+str(a)+")"  
  
def hitungKonso(x):  
    vokal = "aiueoAIUEO"  
    a=0  
    for i in x:  
        if not i in vokal:  
            a+=1  
    return "("+str(len(x))+", "+str(a)+")"  
  
print(hitungVokal("Surakarta"))  
print(hitungKonso("Surakarta"))
```

Python 3.6.3 Shell

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]  
on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\lenovo\Documents\1_D_153\03.py =====  
(9, 4)  
(9, 5)  
>>>
```

Nomor 4



1_D_153.docx - Microsoft Word

```
def rerata(x):  
    total = 0  
    for i in x:  
        total += i  
    return total/len(x)  
  
print(rerata([5,9]))
```

Python 3.6.3 Shell

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]  
on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\lenovo\Documents\1_D_153\04.py =====  
7.0  
>>>
```

Nomor 5

The screenshot shows a Windows desktop with two windows. The background window is a Microsoft Word document titled '1_D_153.docx'. The foreground window is a Python 3.6.3 Shell. The Python script in the shell defines a function `tesPrima(x)` that checks if a number `x` is prime. It uses a loop from 2 to `x-1` and prints 'bilangan prima' if no divisors are found, or 'bukan bilangan prima' otherwise. The script then tests the function for a list of numbers: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41. The output in the shell shows the results for each number, with 'bilangan prima' for primes and 'bukan bilangan prima' for non-primes.

```
def tesPrima(x):  
    if x > 1:  
        for i in range(2,x):  
            if (x % i) == 0:  
                print(x," bukan bilangan prima")  
                break  
            else:  
                print(x," bilangan prima")  
                break  
        else:  
            print(x, " bukan bilangan prima")  
  
x = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41]  
for i in x:  
    tesPrima(i)
```

Output:

```
19 bilangan prima  
23 bilangan prima  
29 bilangan prima  
31 bilangan prima  
37 bilangan prima  
41 bilangan prima  
43 bilangan prima  
47 bilangan prima  
53 bilangan prima  
59 bilangan prima  
61 bilangan prima  
67 bilangan prima  
71 bilangan prima  
73 bilangan prima  
79 bilangan prima  
83 bilangan prima  
89 bilangan prima  
97 bilangan prima  
101 bilangan prima  
103 bilangan prima  
107 bilangan prima  
109 bilangan prima  
113 bilangan prima  
127 bilangan prima  
131 bilangan prima  
137 bilangan prima  
139 bilangan prima  
149 bilangan prima  
151 bilangan prima  
157 bilangan prima  
163 bilangan prima  
167 bilangan prima  
173 bilangan prima  
179 bilangan prima  
181 bilangan prima  
191 bilangan prima  
193 bilangan prima  
197 bilangan prima  
199 bilangan prima  
>>>
```

Nomor 6

The screenshot shows a Windows desktop with two windows. The background window is a Microsoft Word document titled '1_D_153.docx'. The foreground window is a Python 3.6.3 Shell. The Python script in the shell defines a function `tesPrima(x)` that checks if a number `x` is prime. It uses a loop from 2 to `x-1` and prints 'bilangan prima' if no divisors are found, or 'bukan bilangan prima' otherwise. The script then tests the function for a range of numbers from 2 to 1001. The output in the shell shows the results for each number, with 'bilangan prima' for primes and 'bukan bilangan prima' for non-primes.

```
def tesPrima(x):  
    if x > 1:  
        for i in range(2,x):  
            if (x % i) == 0:  
                print(x," bukan bilangan prima")  
                break  
            else:  
                print(x," bilangan prima")  
                break  
        else:  
            print(x, " bukan bilangan prima")  
  
for i in range(2,1001):  
    tesPrima(i)
```

Output:

```
243 bilangan prima  
244 bukan bilangan prima  
245 bilangan prima  
246 bukan bilangan prima  
247 bilangan prima  
248 bukan bilangan prima  
249 bilangan prima  
250 bukan bilangan prima  
251 bilangan prima  
252 bukan bilangan prima  
253 bilangan prima  
254 bukan bilangan prima  
255 bilangan prima  
256 bukan bilangan prima  
257 bilangan prima  
258 bukan bilangan prima  
259 bilangan prima  
260 bukan bilangan prima  
261 bilangan prima  
262 bukan bilangan prima  
263 bilangan prima  
264 bukan bilangan prima  
265 bilangan prima  
266 bukan bilangan prima  
267 bilangan prima  
268 bukan bilangan prima  
269 bilangan prima  
270 bukan bilangan prima  
271 bilangan prima  
272 bukan bilangan prima  
273 bilangan prima  
274 bukan bilangan prima  
275 bilangan prima  
276 bukan bilangan prima  
277 bilangan prima  
278 bukan bilangan prima  
279 bilangan prima  
280 bukan bilangan prima  
281 bilangan prima  
282 bukan bilangan prima
```

Nomor 7

The screenshot shows a Windows desktop with a Microsoft Word document titled "1_D_153.docx" and a Python 3.6.3 Shell window. The Word document contains a Python script for finding prime factors. The Shell window shows the script being executed, resulting in the output [11, 13].

```
def faktorprima(n):
    prima=list()
    for i in range(2,n):
        a = True
        for iter in prima:
            if(i%iter==0):
                a=False
                break
        if a and n%i==0:
            prima.append(i)
    return prima
print(faktorprima(143))
```

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\lenovo\Documents\1_D_153\07.py =====
[11, 13]
>>>
```

Nomor 8

The screenshot shows a Windows desktop with a Microsoft Word document titled "1_D_153.docx" and a Python 3.6.3 Shell window. The Word document contains a Python script for checking if a string is a pangram. The Shell window shows the script being executed, resulting in the output True and False.

```
def apakahTerkandung(a,b):
    return a in b
print(apakahTerkandung("db","abodcdsqwedb"))
print(apakahTerkandung("abd","abc"))
```

```
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\lenovo\Documents\1_D_153\08.py =====
True
False
>>>
```

Nomor 9

The screenshot shows a Windows desktop with two windows. The left window is a text editor titled '09.py - C:\Users\lenovo\Documents\1_D_153\09.py (3.6.3)'. It contains a Python function `def iterasi():` with a `for i in range(1,100):` loop. Inside the loop, there are conditional statements: `if (i%3)!=0 and (i%5)!=0: print(i)`, `else: if (i%15)==0: print("python UMS")`, `elif (i%3)==0: print("python")`, and `elif (i%5)==0: print("UMS")`. The function is called `iterasi()`. The right window is a 'Python 3.6.3 Shell' showing the output of the script. It prints numbers from 1 to 99, with 'python UMS' printed at line 61, 'python' at line 62, 'UMS' at line 64, and so on, following the logic of the script. The taskbar at the bottom shows the Windows logo, task view, and several application icons. The system tray shows the date and time as 21:01 on 04/03/2019.

```
def iterasi():
    for i in range(1,100):
        if (i%3)!=0 and (i%5)!=0:
            print(i)
        else:
            if (i%15)==0:
                print("python UMS")
            elif (i%3)==0:
                print("python")
            elif (i%5)==0:
                print("UMS")
    iterasi()
```

```
61
62
python
64
UMS
67
python
68
UMS
71
python
73
UMS
74
python UMS
76
77
python
79
UMS
python
82
83
python
UMS
86
python
88
89
python UMS
91
92
python
94
UMS
python
97
98
python
>>>
```

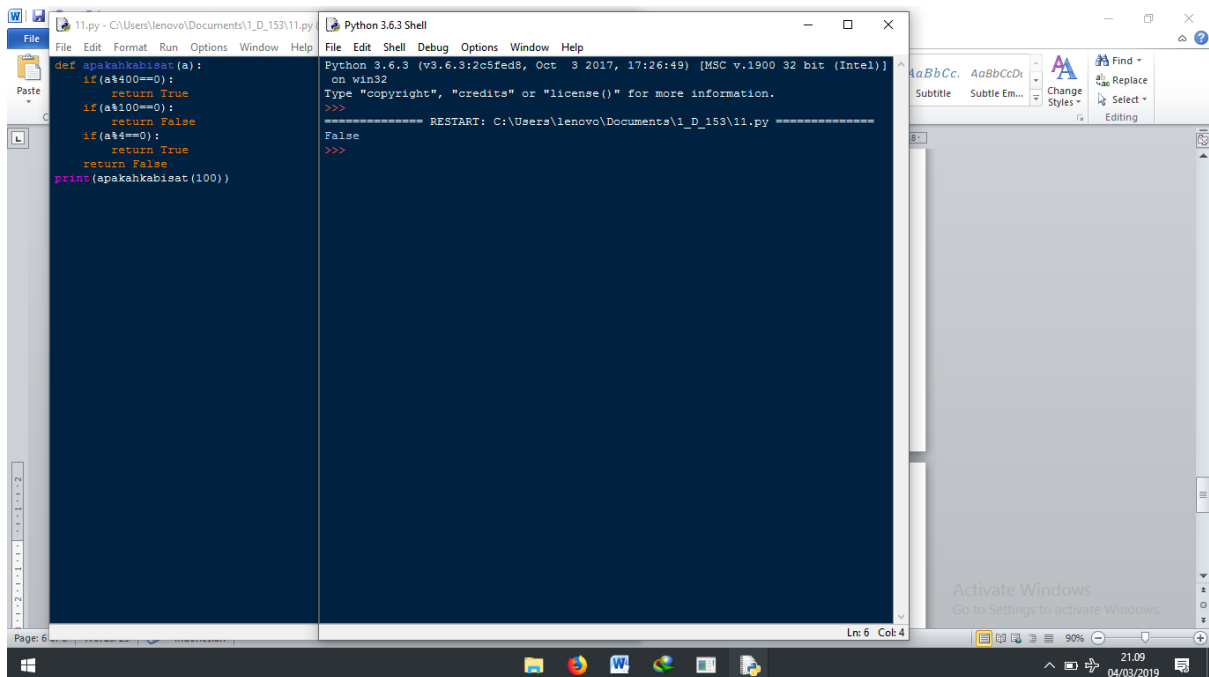
Nomor 10

The screenshot shows a Windows desktop with two windows. The left window is a text editor titled '10.py - C:\Users\lenovo\Documents\1_D_153\10.py (3.6.3)'. It contains a Python function `def selesaikanABC(a,b,c):` that calculates the discriminant `D=(b**2)-(4*a*c)`. If `D<0`, it returns "determinan negatif", and otherwise, it returns "determinan positif". The function is called `print(selesaikanABC(1,1,2))`. The right window is a 'Python 3.6.3 Shell' showing the output of the script. It displays the Python version and system information, followed by a restart message and the output 'determinan negatif'. The taskbar at the bottom shows the Windows logo, task view, and several application icons. The system tray shows the date and time as 21:02 on 04/03/2019.

```
def selesaikanABC(a,b,c):
    a=float(a)
    b=float(b)
    c=float(c)
    D=(b**2)-(4*a*c)
    if D<0:
        return "determinan negatif"
    return "determinan positif"
print(selesaikanABC(1,1,2))
```

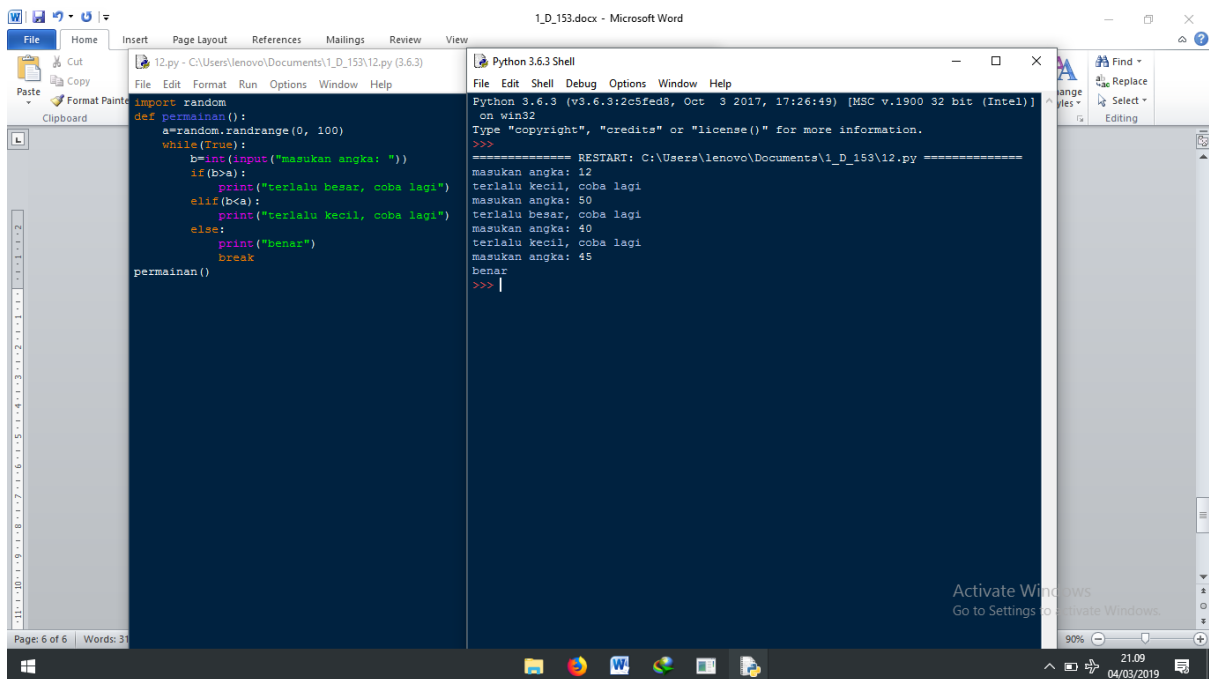
```
Python 3.6.3 (tags/v3.6.3:2c5fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\lenovo\Documents\1_D_153\10.py =====
determinan negatif
>>>
```

Nomor 11



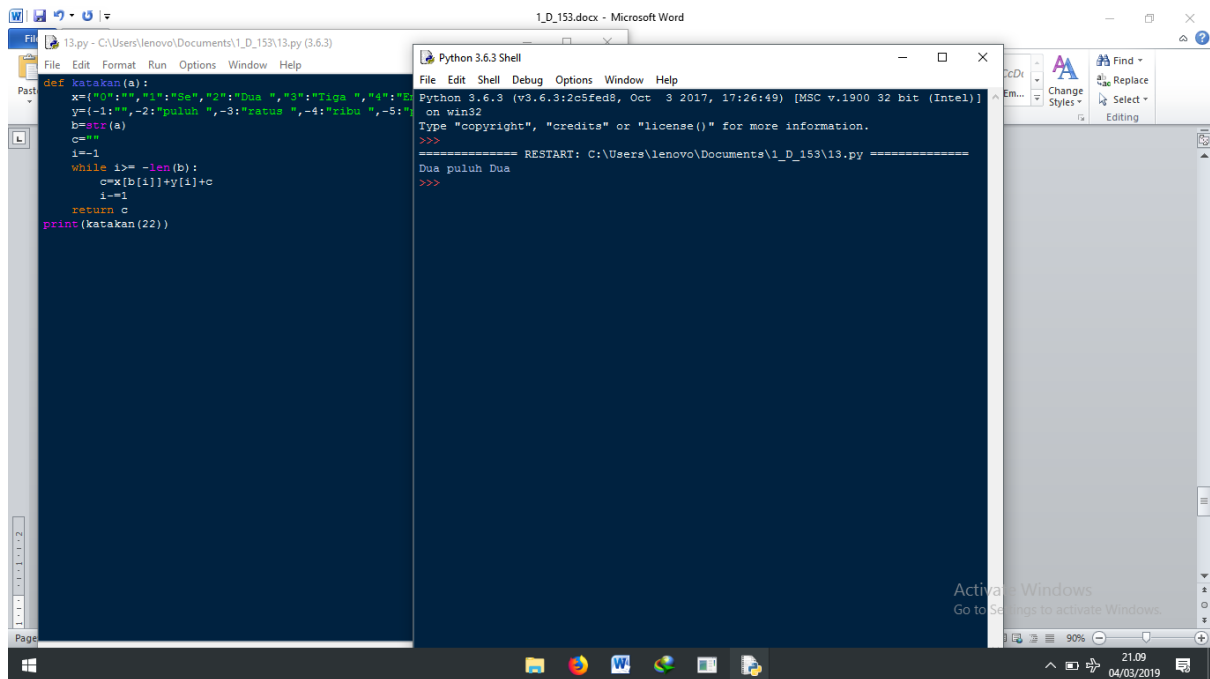
```
def apakahkabisat(a):  
    if(a%400==0):  
        return True  
    if(a%100==0):  
        return False  
    if(a%4==0):  
        return True  
    return False  
print(apakahkabisat(100))
```

Nomor 12



```
import random  
def permainan():  
    a=random.randrange(0, 100)  
    while(True):  
        b=int(input("masukan angka: "))  
        if(b>a):  
            print("terlalu besar, coba lagi")  
        elif(b<a):  
            print("terlalu kecil, coba lagi")  
        else:  
            print("benar")  
            break  
    permainan()
```

Nomor 13

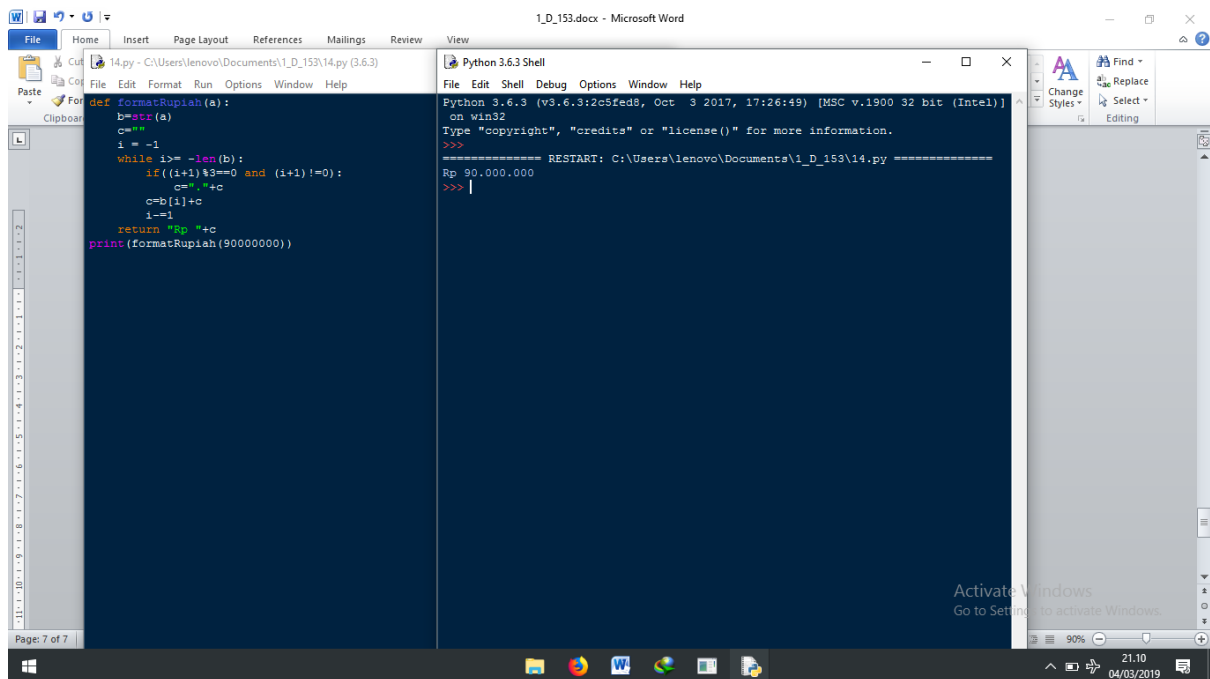


The screenshot shows a Windows desktop with a Microsoft Word document titled "1_D_153.docx" and a Python 3.6.3 Shell window. The Word document contains a Python script named 13.py, which defines a function `katakan(a)` that takes a string `a` and returns a string `c` based on the length of `a`. The Shell window shows the execution of the script, which outputs "Dua puluh Dua".

```
def katakan(a):  
    x=("0":"", "1":"Se", "2":"Dua ", "3":"Tiga ", "4":"E",  
      "-1":"", "-2":"puluh ", "-3":"ratus ", "-4":"ribu ", "-5":"")  
    b=str(a)  
    c=""  
    i=-1  
    while i >= -len(b):  
        c=x[b[i]]+y[i]+c  
        i=i-1  
    return c  
print(katakan(22))
```

```
Python 3.6.3 (v3.6.3:205fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]  
on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\lenovo\Documents\1_D_153\13.py =====  
Dua puluh Dua  
>>>
```

Nomor 14



The screenshot shows a Windows desktop with a Microsoft Word document titled "1_D_153.docx" and a Python 3.6.3 Shell window. The Word document contains a Python script named 14.py, which defines a function `formatRupiah(a)` that takes a string `a` and returns a string `c` based on the length of `a`. The Shell window shows the execution of the script, which outputs "Rp 90.000.000".

```
def formatRupiah(a):  
    b=str(a)  
    c=""  
    i = -1  
    while i >= -len(b):  
        if ((i+1)%3==0 and (i+1)!=0):  
            c="."+c  
        c=b[i]+c  
        i=i-1  
    return "Rp "+c  
print(formatRupiah(90000000))
```

```
Python 3.6.3 (v3.6.3:205fed8, Oct 3 2017, 17:26:49) [MSC v.1900 32 bit (Intel)]  
on win32  
Type "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\lenovo\Documents\1_D_153\14.py =====  
Rp 90.000.000  
>>>
```