In [1]:

```
versoin
```

```
---------------------------------------------------------------------
-
NameError                                 Traceback (most recent call las
t)
<ipython-input-1-0da2a7b0c4cb> in <module>
----> 1 versoin

NameError: name 'versoin' is not defined
```

In [30]:

```
print("tanoj")
```

```
tanoj
```

In [31]:

```
age = 20
print('age')
```

```
20
```

In [32]:

```
print(age)
```

```
20
```

In [7]:

```
first_name = "Tanoj"
last_name = "N"
full_name = f"{first_name} {last_name}"
print(full_name)
```

```
Tanoj N
```

In [8]:

```
## addressing whitespaces
```

In [9]:

```
print("Python")
```

```
Python
```

In [12]:

```
print("\tpython") # print using tab space
```

```
        python
```

In [13]:

```python
print("\nPython") # print in new line
```

Python

In [14]:

```python
print("Languages:\nPython\nScala\nJava")
```

Languages:
Python
Scala
Java

In [17]:

```python
x = "value"
```

In [19]:

```python
x = "value "

print(x)
```

value

In [20]:

```python
print(x.strip())     # removes the white spaces
```

value

In [21]:

```python
# Numbers
```

In [22]:

```python
2+1
```

Out[22]:

3

In [23]:

```python
3/2
```

Out[23]:

1.5

In [24]:

```python
x=1
y=2
z=3
```

In [26]:

```python
x,y,z=1,2,3  # advisible to declare a variable like this when we need to assign more va
riables
```

In [27]:

```python
import this
```

The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

In [ ]: