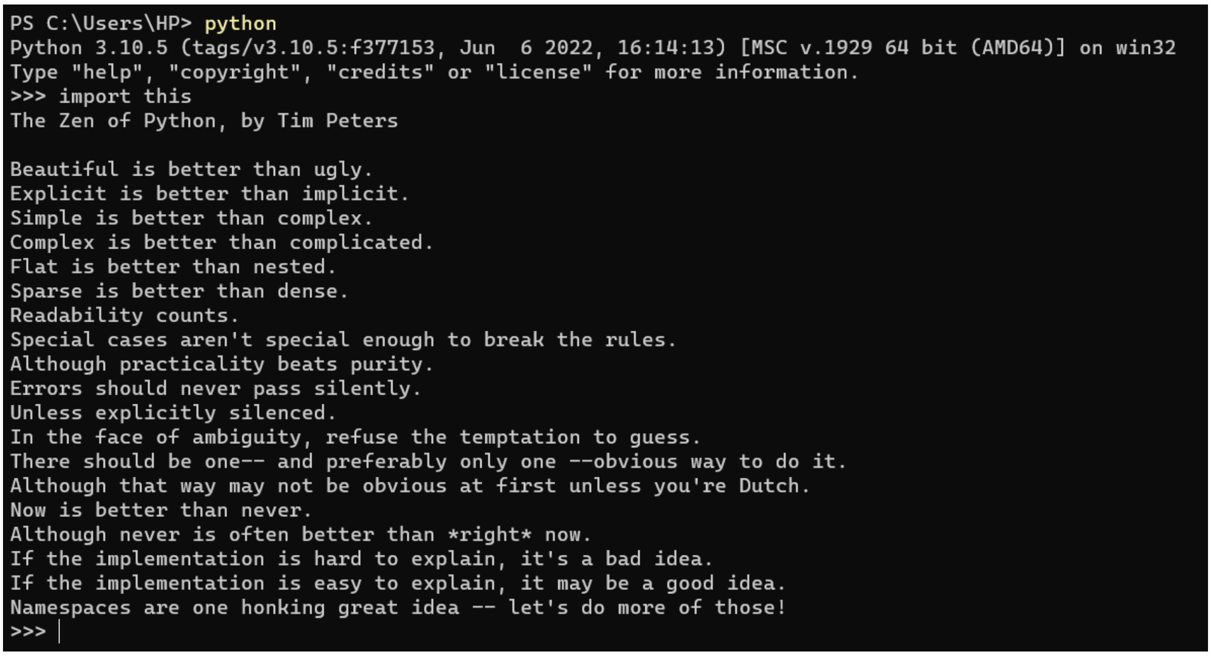
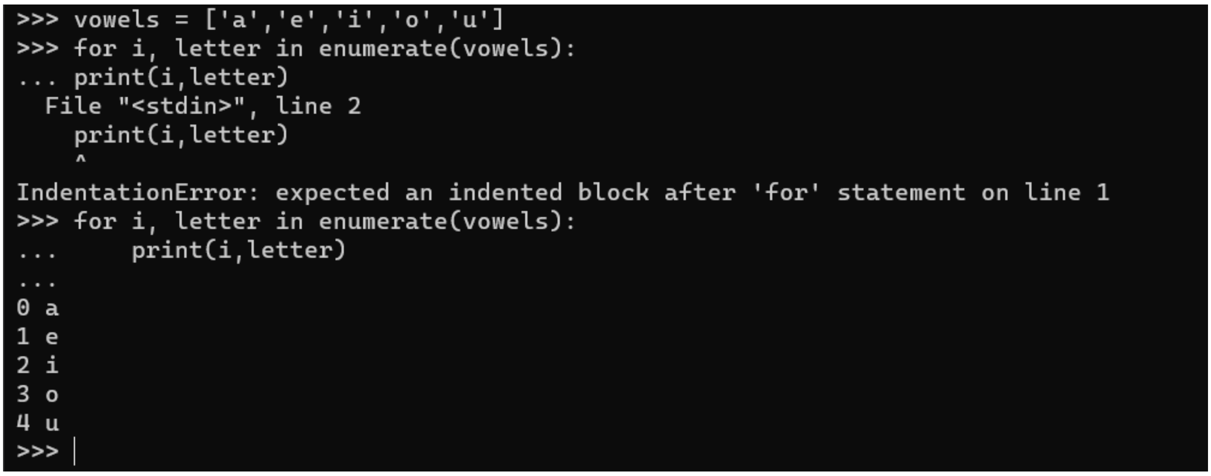
Interesting Facts about Python

1. Want to read a poem on Python by Time Peters?
   Then use the following commands -
   “import this” in the interpreter.



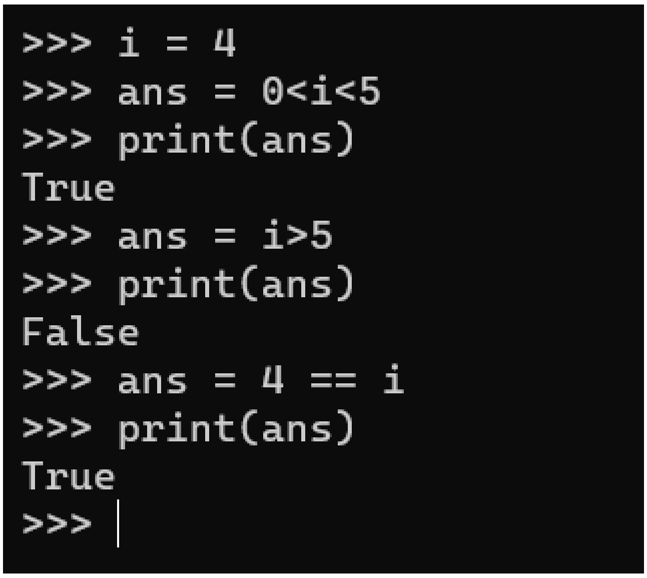
```
PS C:\Users\HP> python
Python 3.10.5 (tags/v3.10.5:f377153, Jun  6 2022, 16:14:13) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

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!
>>>
```

2. Want to find the index inside a for loop? Wrap an iterable with ‘enumerate’ and it will yield the item along with its index. See this code snippet.



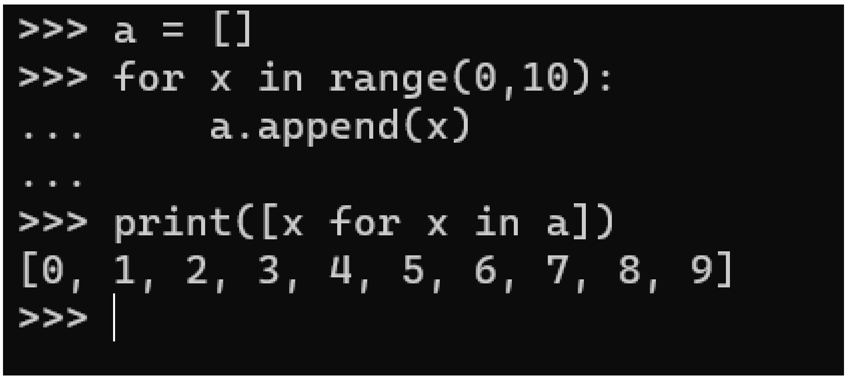
```
>>> vowels = ['a','e','i','o','u']
>>> for i, letter in enumerate(vowels):
... print(i,letter)
  File "<stdin>", line 2
    print(i,letter)
    ^
IndentationError: expected an indented block after 'for' statement on line 1
>>> for i, letter in enumerate(vowels):
...     print(i,letter)
...
0 a
1 e
2 i
3 o
4 u
>>>
```

3. Comparison operators can be used in such a way that variable can store boolean values



```
>>> i = 4
>>> ans = 0<i<5
>>> print(ans)
True
>>> ans = i>5
>>> print(ans)
False
>>> ans = 4 == i
>>> print(ans)
True
>>>
```

4. Instead of building a list with a loop, one can build it more concisely with a list comprehension. See this code for more understanding.



```
>>> a = []
>>> for x in range(0,10):
...     a.append(x)
...
>>> print([x for x in a])
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```