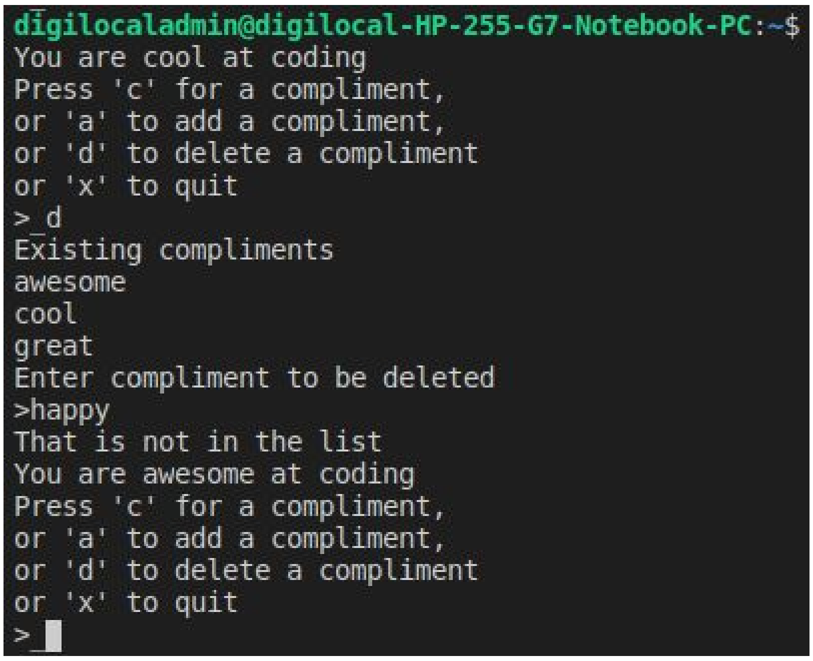
Complimentary Code

MAKING OUR DIGITAL FUTURE

INTRODUCTION

In this project guide we'll use lists to generate random compliments for the user. We'll also be able to add compliments and remove them from our final programme.

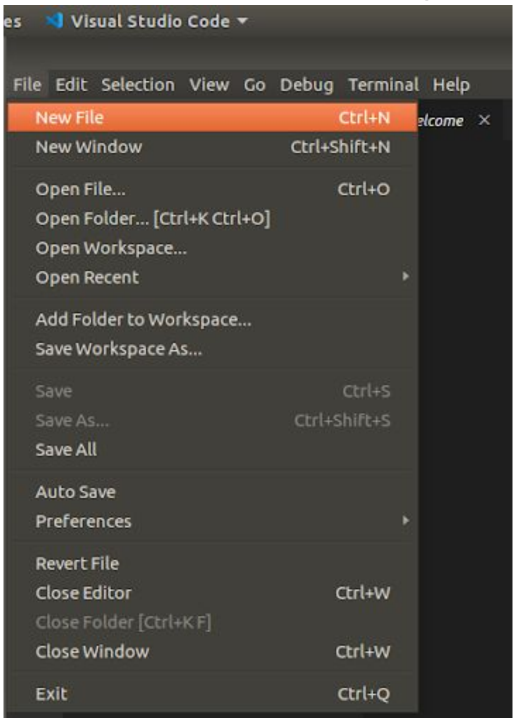


```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
You are cool at coding
Press 'c' for a compliment,
or 'a' to add a compliment,
or 'd' to delete a compliment
or 'x' to quit
>_d
Existing compliments
awesome
cool
great
Enter compliment to be deleted
>happy
That is not in the list
You are awesome at coding
Press 'c' for a compliment,
or 'a' to add a compliment,
or 'd' to delete a compliment
or 'x' to quit
>_
```

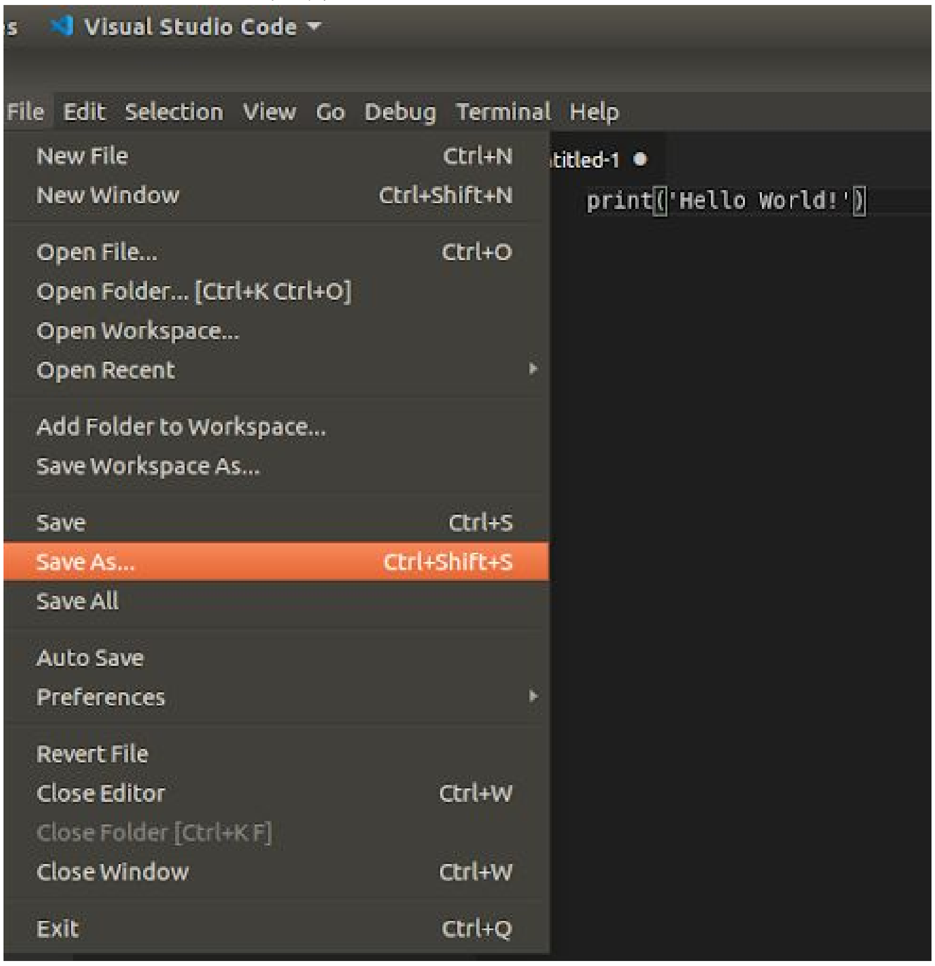
Step One: Getting started

✓ Open VS Code from the icon in your side bar.

✓ We need a new file to begin editing. Start one from the 'File' menu.



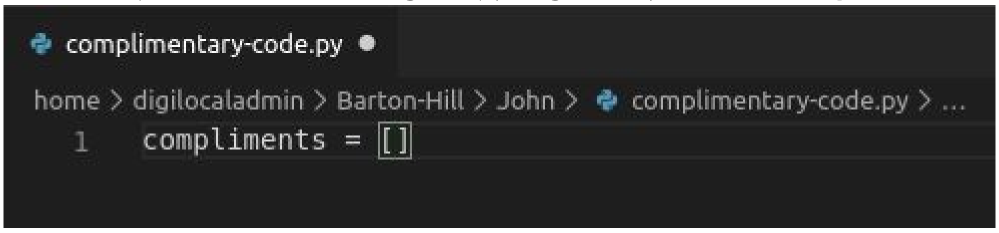
- Let’s tell VS Code we’re using python by saving our file (even though it’s still empty).



```
print('Hello World!')
```

- Call your programme something like complimentary-code.py it’s a good idea not to use spaces in file names for programming as it can cause problems later on.

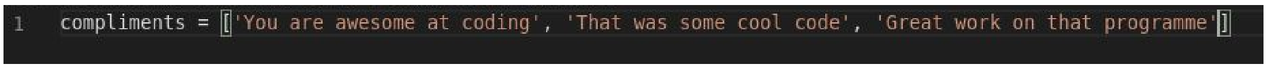
- In your new file begin typing compliments = [


complimentary-code.py
home > digilocaladmin > Barton-Hill > John > complimentary-code.py > ...

```
compliments = []
```

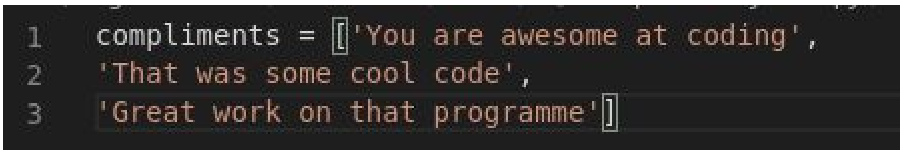
- Notice how even though you only typed the first square bracket the programme has added the second]

- Add the rest of our variable definition



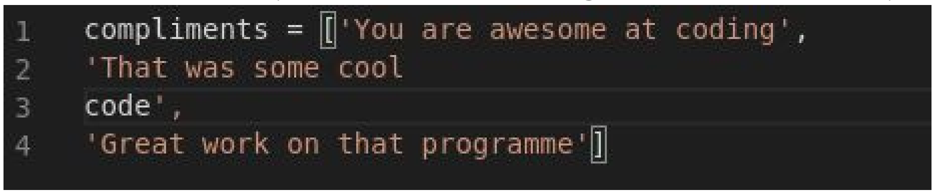
```
compliments = ['You are awesome at coding', 'That was some cool code', 'Great work on that programme']
```

- You can also split your list across lines so it's easier to read.



```
compliments = ['You are awesome at coding',
'That was some cool code',
'Great work on that programme']
```

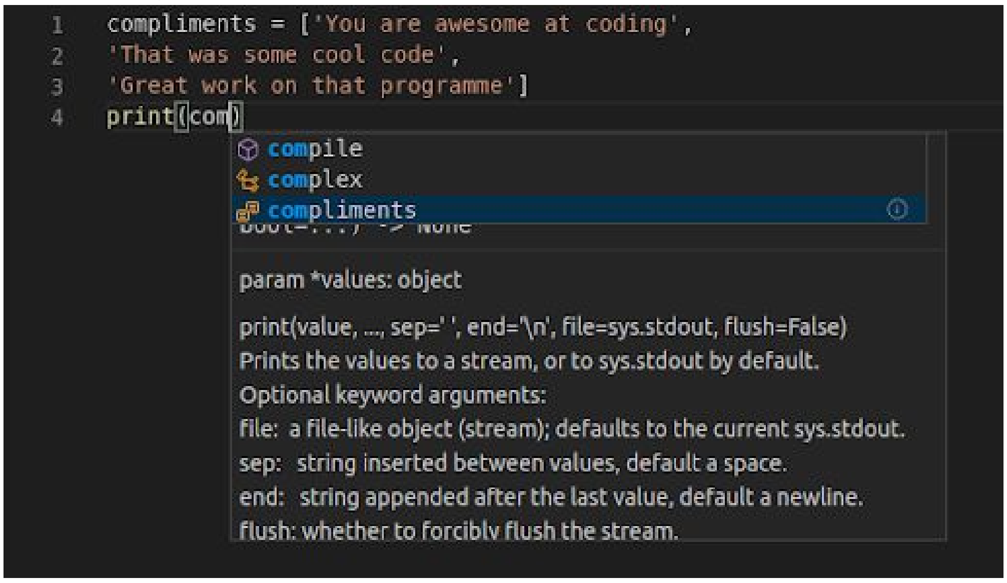
- But don't split individual strings, that will cause problems.



```
compliments = ['You are awesome at coding',
'That was some cool
code',
'Great work on that programme']
```

- See how the word 'code' is now white - that means VS Code things it's a python command (which it isn't) so will produce an error if we try and run our programme. Let's put everything right, and check our string has been created.

- As you begin to type print(compliments) a new window will pop up to offer suggestions on which variables you have already used. This is very handy as for python, Compliments is not the same as compliments, and obviously different to complements. Spelling differences are one of the most common causes of errors for python programmers, so this is a very useful feature!



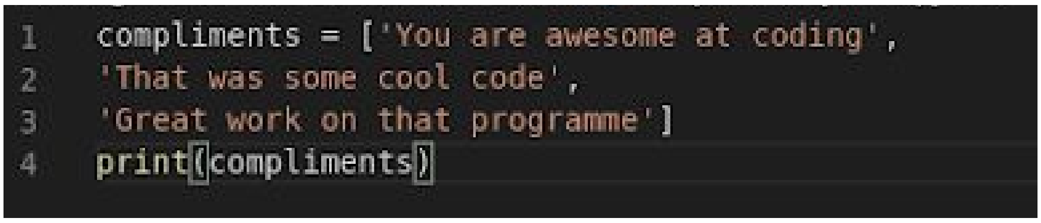
```
compliments = ['You are awesome at coding',
'That was some cool code',
'Great work on that programme']
print(com)
```

compile
complex
compliments

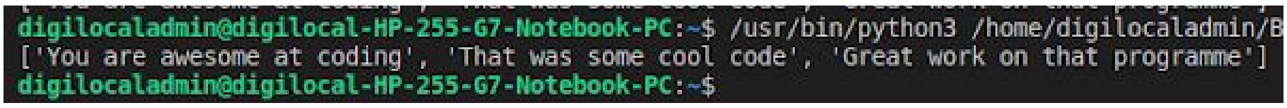
param *values: object

print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
Prints the values to a stream, or to sys.stdout by default.
Optional keyword arguments:
file: a file-like object (stream); defaults to the current sys.stdout.
sep: string inserted between values, default a space.
end: string appended after the last value, default a newline.
flush: whether to forcibly flush the stream.

- Our code should now look like this.



```
compliments = ['You are awesome at coding',
'That was some cool code',
'Great work on that programme']
print(compliments)
```



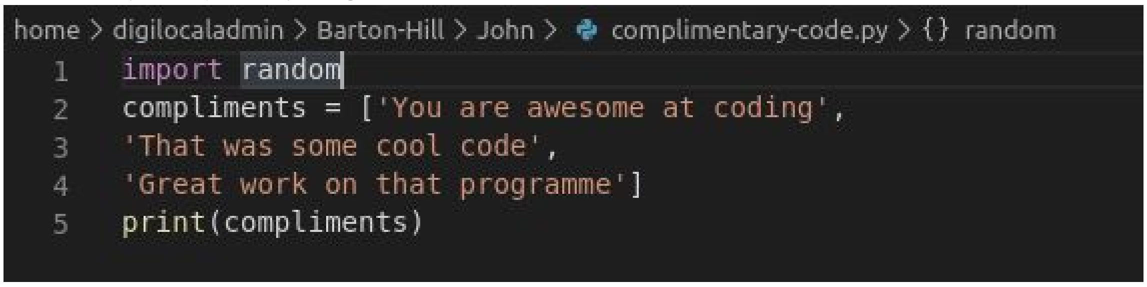
```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$ /usr/bin/python3 /home/digilocaladmin/B
['You are awesome at coding', 'That was some cool code', 'Great work on that programme']
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

Great, it worked!

Step Two: Mixing things up

- Computers are great at doing the same thing over and over again. So picking random numbers is really quite tricky. Helpfully, there is a python library to do this. A software library is just like a regular library, it has lots of bits of code we can use without having to write everything ourselves.

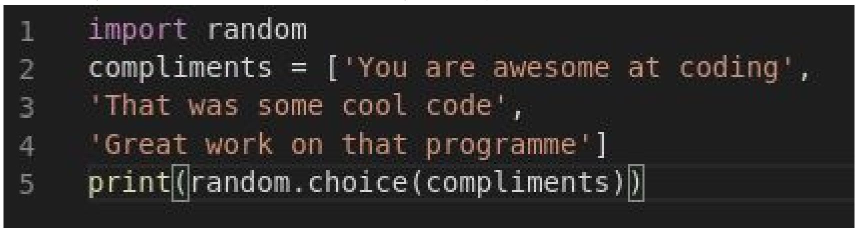
- First, let's tell python to use the random library. Start a new line at the top of your programme.



```
home > digilocaladmin > Barton-Hill > John > complimentary-code.py > {} random
import random
compliments = ['You are awesome at coding',
'That was some cool code',
'Great work on that programme']
print(compliments)
```

- It's a good idea to add things like import statements to the top of your code so you can quickly see which libraries you're using (and so can anyone else that needs to read your code).

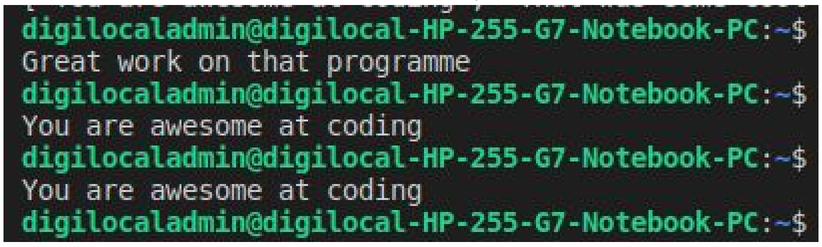
- Now instead of printing the whole list of compliments, let's change our print statement to print just one, at random.



```
import random
compliments = ['You are awesome at coding',
'That was some cool code',
'Great work on that programme']
print(random.choice(compliments))
```

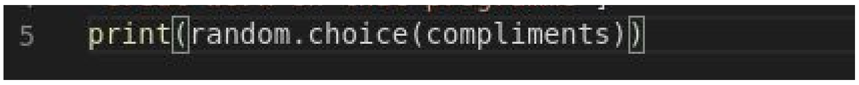
Test your code!
A few times.

You should see the output below.



```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
Great work on that programme
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
You are awesome at coding
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
You are awesome at coding
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

Let’s take another look at line 5 of our code.



```
5    print(random.choice(compliments))
```

There are three parts to our print statement now:

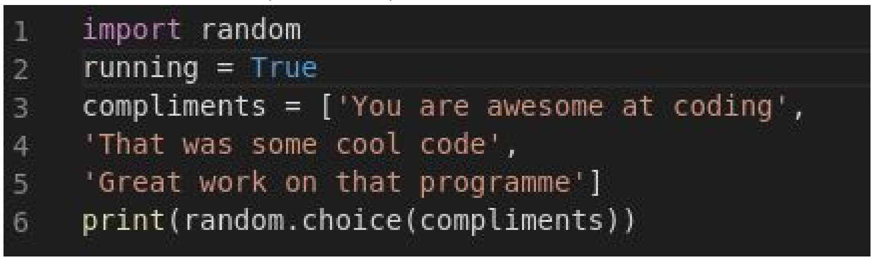
1. random - this tells us we’re using the random code library
2. .choice - this tells us we’re using the choice function within that library
3. (compliments) is the list of things we want to pick something from

There is actually a fourth part, but you can’t really see it. The function .choice returns whichever thing in the list it picked and that is what gets printed out. Function’s don’t always have to return something, but this one does.

Step Three: Endless compliments

Right now we have to run our programme every time we want a compliment. Let’s change it so that we can have endless compliments!

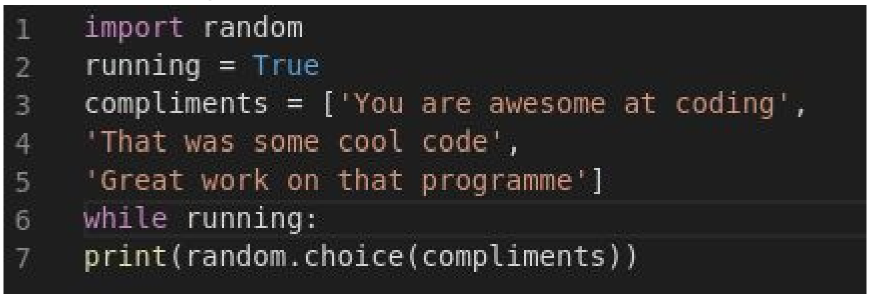
We’ll create a new variable called running and set it to True (the capital letter is important).



```
1    import random
2    running = True
3    compliments = ['You are awesome at coding',
4    'That was some cool code',
5    'Great work on that programme']
6    print(random.choice(compliments))
```

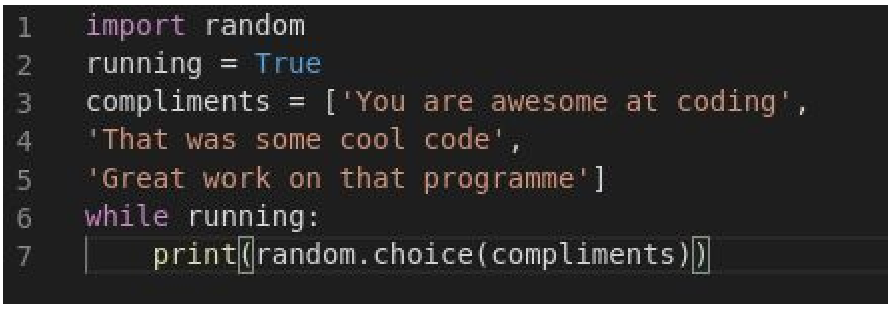
- Just as we should put all our import statements at the top of our programme, variable definitions should also be near the start.

- Now we can add our loop statement. We don't know how many compliments we will want, so we can't use a 'for i in range(x):' because we don't know what to set x = to. What we can use is loop while running is True.



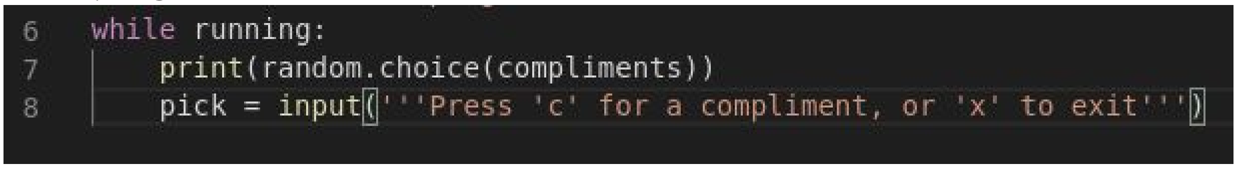
```python
import random
running = True
compliments = ['You are awesome at coding',
'That was some cool code',
'Great work on that programme']
while running:
print(random.choice(compliments))
```

- But our print statement is outside the loop. Use the <TAB> key to indent your print statement once.



```python
import random
running = True
compliments = ['You are awesome at coding',
'That was some cool code',
'Great work on that programme']
while running:
    print(random.choice(compliments))
```

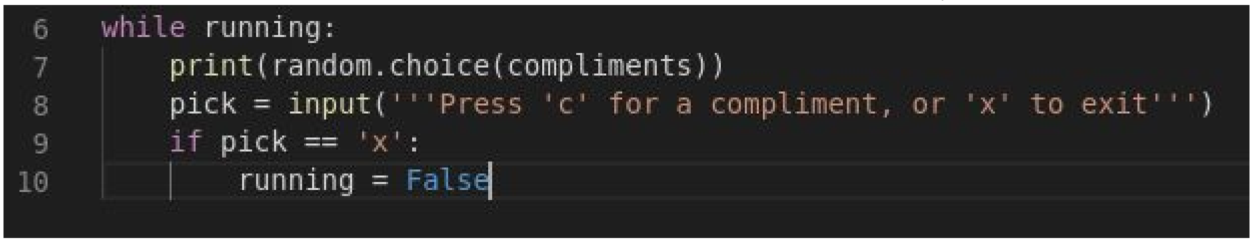
- Our programme has nothing to stop it running! Let's add a new line to ask the user if they would like another compliment, or to exit the programme.



```python
while running:
    print(random.choice(compliments))
    pick = input('''Press 'c' for a compliment, or 'x' to exit''')
```

- We've used the 3 quotes for print, because otherwise the quotes around our 'c' and 'x' would cause problems.

Next we need to make a decision what to do if our pick is ‘c’ or ‘x’



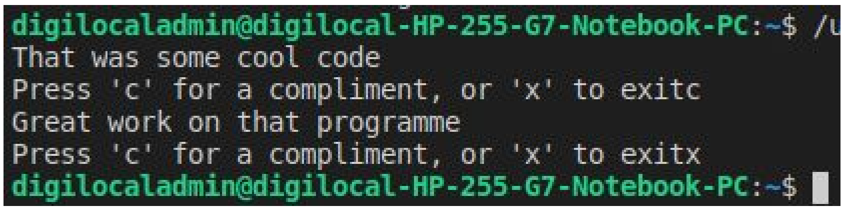
```
6   while running:
7       print(random.choice(compliments))
8       pick = input('''Press 'c' for a compliment, or 'x' to exit''')
9       if pick == 'x':
10          running = False
```

We can actually stop here, as there are only two options. To quit or keep going. Notice that our if statement uses == and not just a single =.

1. = means ‘set the value to’
2. == means ‘has the same value as’

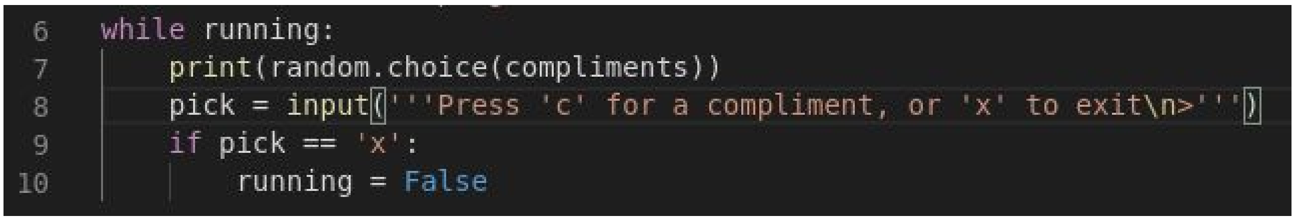
Notice also that our line 10 has two levels of indent. It is within the if loop, that is within the while loop.

Test your code!



```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$ /u
That was some cool code
Press 'c' for a compliment, or 'x' to exitc
Great work on that programme
Press 'c' for a compliment, or 'x' to exitx
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$ 
```

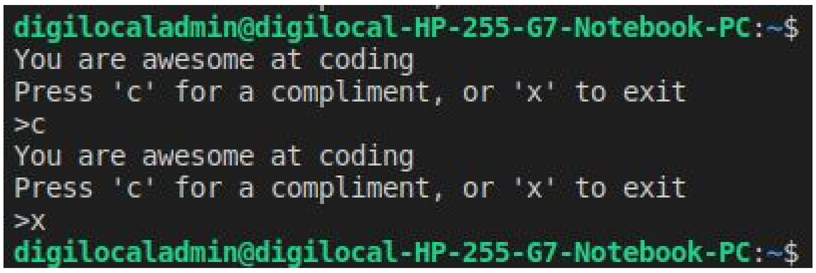
You will need to click on the terminal window at the bottom of VS Code to be able to type there. And your input is right against the input line. Let’s tidy things up a little.



```
while running:
    print(random.choice(compliments))
    pick = input('''Press 'c' for a compliment, or 'x' to exit\n>''')
    if pick == 'x':
        running = False
```

We’ve added a \n to our print statement. This is a special command that means, add a new line here. We’ve then put a > sign in to show where the user should type.

Test your code!



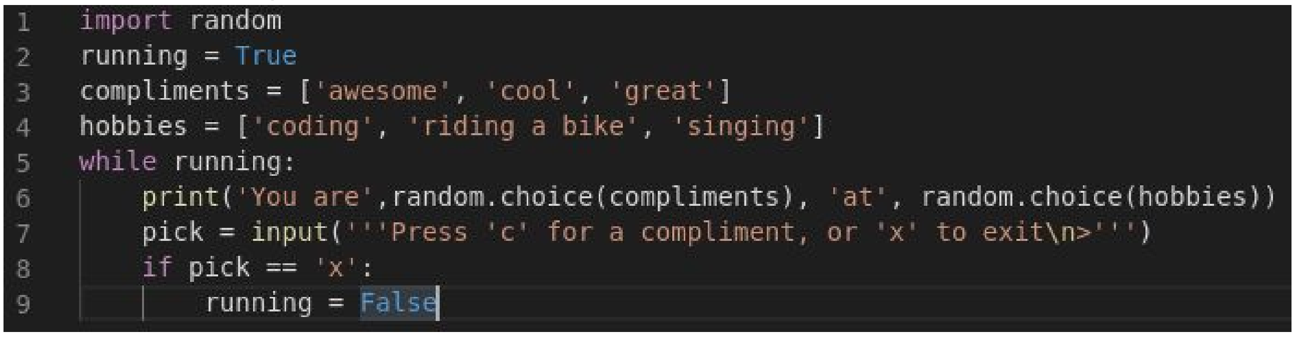
```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
You are awesome at coding
Press 'c' for a compliment, or 'x' to exit
>c
You are awesome at coding
Press 'c' for a compliment, or 'x' to exit
>x
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

Much neater.

Step Four: More variation

- At the moment all our compliments are held in one list. If we want to add more, or change them, we have to edit the whole list. How about we break it up to have a compliment, and then the thing you're good at.

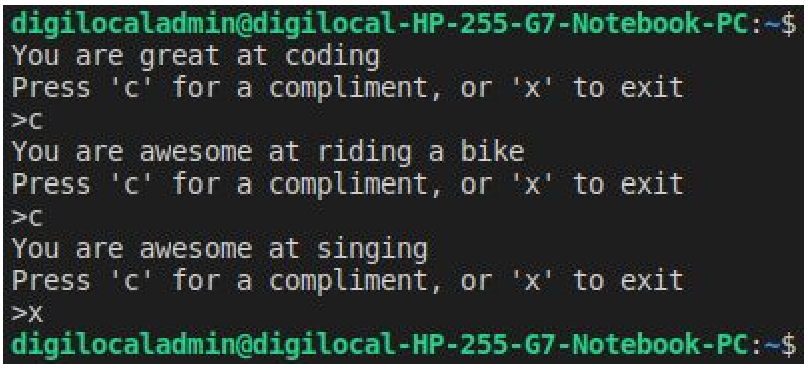
- Let's change our list of compliments, and add a new list of hobbies:



```python
import random
running = True
compliments = ['awesome', 'cool', 'great']
hobbies = ['coding', 'riding a bike', 'singing']
while running:
    print('You are',random.choice(compliments), 'at', random.choice(hobbies))
    pick = input('''Press 'c' for a compliment, or 'x' to exit\n>''')
    if pick == 'x':
        running = False
```

- Our print statement has also got a little more complex! Now we print some text 'You are', then our random.choice of compliment, then some text 'at', then our random.choice of hobby.

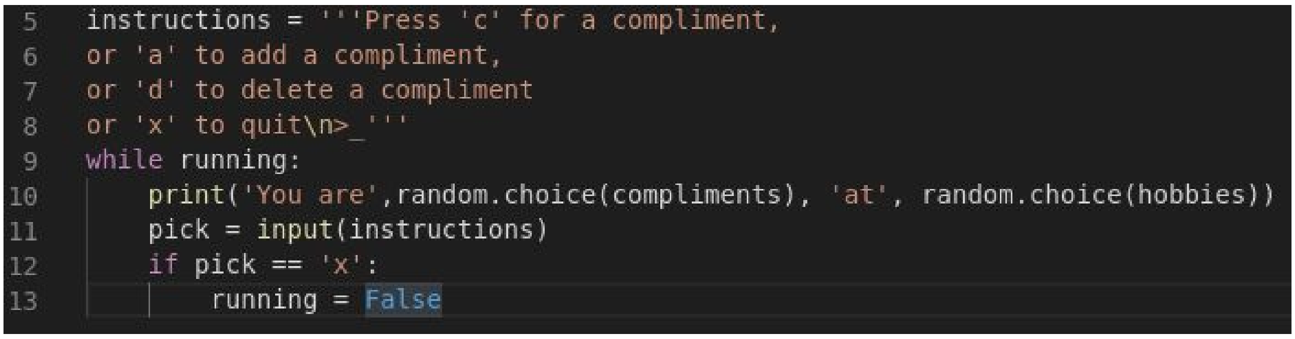
Test your code!



```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
You are great at coding
Press 'c' for a compliment, or 'x' to exit
>c
You are awesome at riding a bike
Press 'c' for a compliment, or 'x' to exit
>c
You are awesome at singing
Press 'c' for a compliment, or 'x' to exit
>x
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

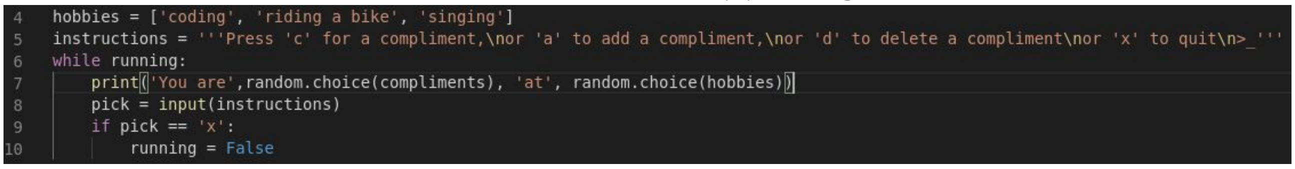
- Now that we have our compliments and hobbies in separate lists, we can begin to think about adding to them.

- First let’s change our instructions, and rather than keep adding to our print statement (which will start to get messy), let’s make a new variable for our instructions.



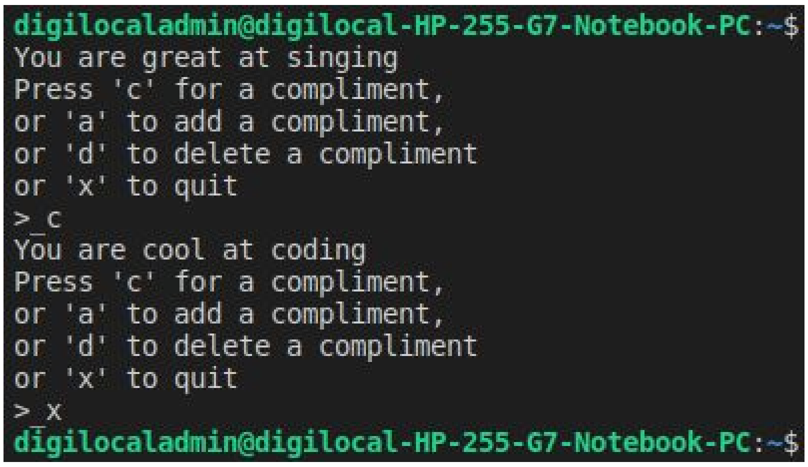
```python
instructions = '''Press 'c' for a compliment,
or 'a' to add a compliment,
or 'd' to delete a compliment
or 'x' to quit\n>_'''
while running:
    print('You are',random.choice(compliments), 'at', random.choice(hobbies))
    pick = input(instructions)
    if pick == 'x':
        running = False
```

- We could type it all out on one line, but that’s not easy to read. You don’t need to do this, both examples do the same thing and the one above is much easier to see what’s happening.



```python
hobbies = ['coding', 'riding a bike', 'singing']
instructions = '''Press 'c' for a compliment,\nor 'a' to add a compliment,\nor 'd' to delete a compliment\nor 'x' to quit\n>_'''
while running:
    print('You are',random.choice(compliments), 'at', random.choice(hobbies))
    pick = input(instructions)
    if pick == 'x':
        running = False
```

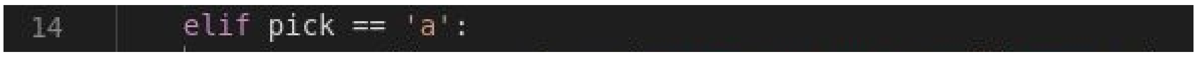
Test your code!



```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
You are great at singing
Press 'c' for a compliment,
or 'a' to add a compliment,
or 'd' to delete a compliment
or 'x' to quit
>_c
You are cool at coding
Press 'c' for a compliment,
or 'a' to add a compliment,
or 'd' to delete a compliment
or 'x' to quit
>_x
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
```

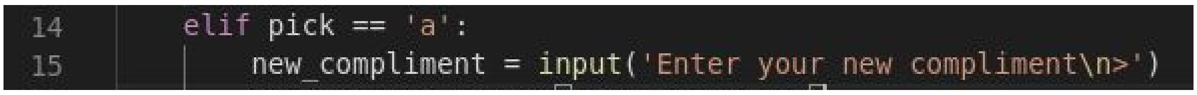
- You could also press a, or d, or anything (other than x) to get a new compliment. Let’s add the code to add a compliment to our list.

- The first thing is to see if an ‘a’ has been typed. elif is python shorthand for else if.



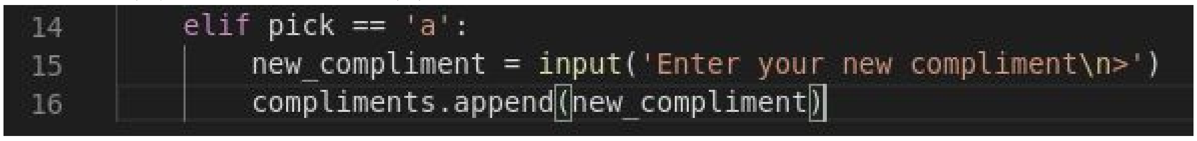
```python
    elif pick == 'a':
```

- Then we need to ask for the new compliment.



```python
    elif pick == 'a':
        new_compliment = input('Enter your new compliment\n>')
```

- Finally we need to add it to our list. There is a special command for lists in python called .append which does this for us.



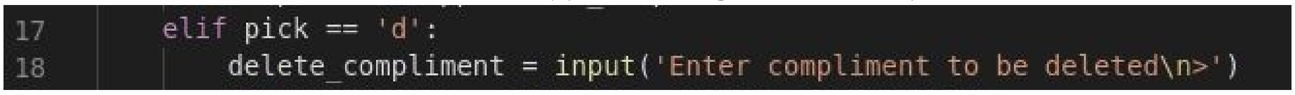
```python
    elif pick == 'a':
        new_compliment = input('Enter your new compliment\n>')
        compliments.append(new_compliment)
```

Test your code!

- Remember that because you create your list of compliments each time you run the programme, it won’t remember any you have added.

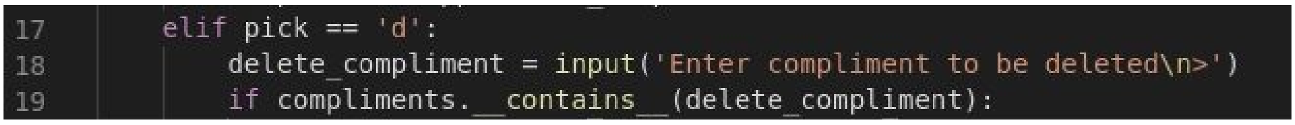
- We can do a similar thing to delete compliments. However, deleting is a little more complex because we need to check that the compliment entered is actually in the list to be deleted.

- First see if an ‘d’ has been typed and get the compliment to be deleted.



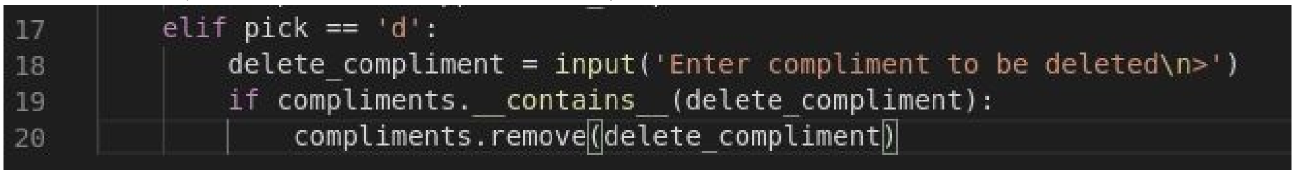
```python
    elif pick == 'd':
        delete_compliment = input('Enter compliment to be deleted\n>')
```

- Now we can use a special python command to see if our list contains our compliment to be deleted.



```python
    elif pick == 'd':
        delete_compliment = input('Enter compliment to be deleted\n>')
        if compliments.__contains__(delete_compliment):
```

- Be careful - there are two _ _ on either side of the word ‘contains’. Finally we can remove the compliment to be deleted

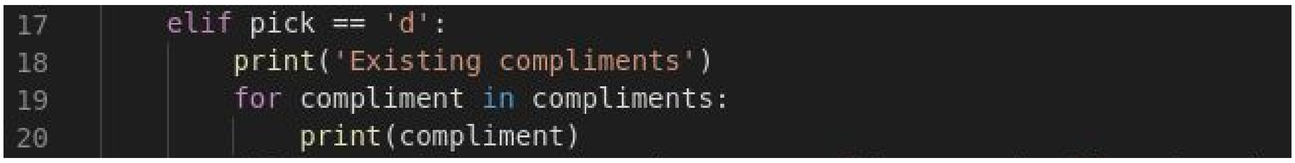


```
17     elif pick == 'd':
18         delete_compliment = input('Enter compliment to be deleted\n>')
19         if compliments.__contains__(delete_compliment):
20             compliments.remove(delete_compliment)
```

Test your code!

- This works but it’s kind of hard to remember what compliments are in the list and which aren’t. Also, if you type something in that isn’t in the list, it just goes around and you’re not sure why. So lets add a bit more to remind us what’s in the list, and give a helpful note if we enter something that isn’t.

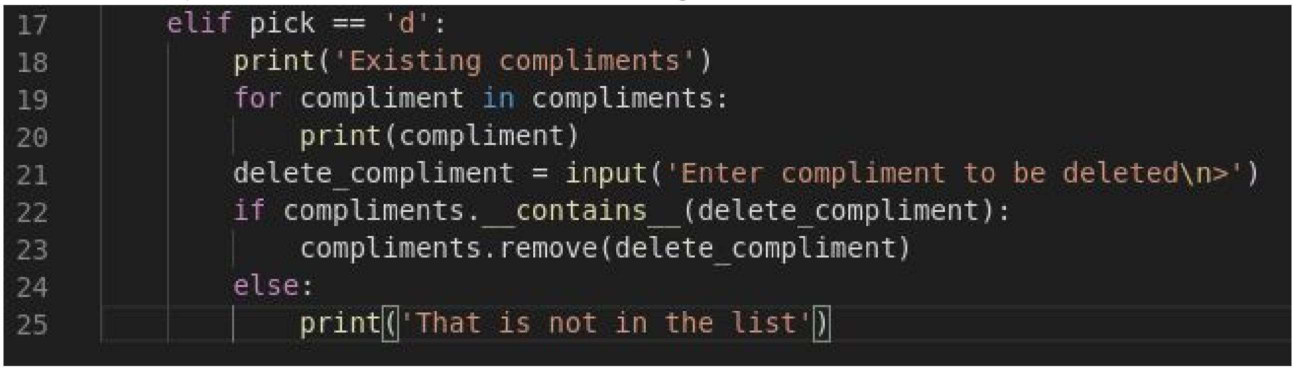
- We begin by printing out the list of compliments. We could just print(compliments) but that wouldn’t look nice as it would include the square brackets and quote marks. Instead, we can print each compliment out on it’s own line. Add the 3 new lines (18 to 20).



```
17     elif pick == 'd':
18         print('Existing compliments')
19         for compliment in compliments:
20             print(compliment)
```

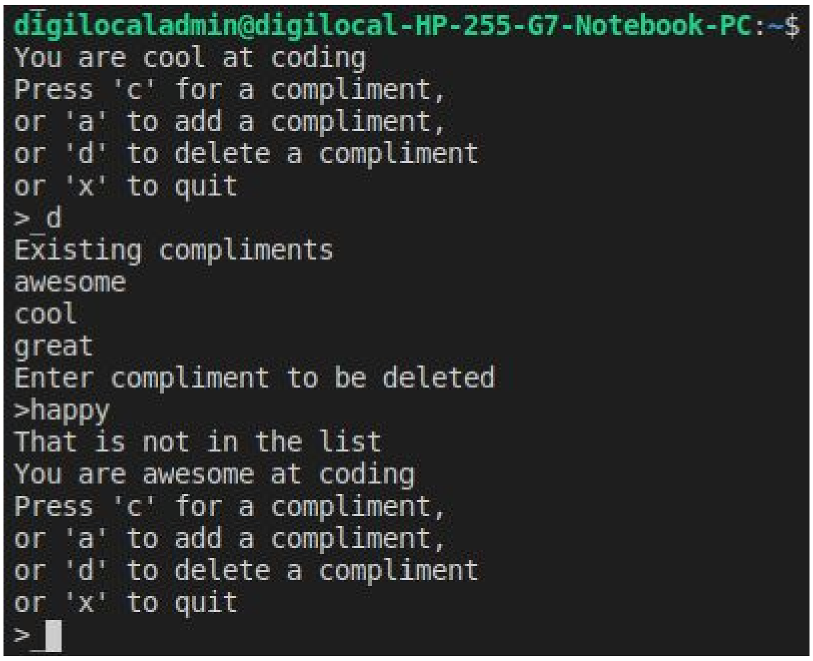
- The command ‘for compliment in compliments’ takes each item in our list of compliments and stores it in a new variable compliment, which we then print.

Finally we can add the error message.



```
    elif pick == 'd':
        print('Existing compliments')
        for compliment in compliments:
            print(compliment)
        delete_compliment = input('Enter compliment to be deleted\n>')
        if compliments.__contains__(delete_compliment):
            compliments.remove(delete_compliment)
        else:
            print('That is not in the list')
```

Test your code!



```
digilocaladmin@digilocal-HP-255-G7-Notebook-PC:~$
You are cool at coding
Press 'c' for a compliment,
or 'a' to add a compliment,
or 'd' to delete a compliment
or 'x' to quit
> d
Existing compliments
awesome
cool
great
Enter compliment to be deleted
>happy
That is not in the list
You are awesome at coding
Press 'c' for a compliment,
or 'a' to add a compliment,
or 'd' to delete a compliment
or 'x' to quit
> 
```

Challenge time!

Can you add 'n' new hobbies, and 'r' remove old ones?