

Python
1

(Beta) Compliment Generator



Introduction:

Learn how to use lists, to store lots of data in 1 variable.



Activity Checklist

Follow these **INSTRUCTIONS** one by one



Test your Project

Click on the green flag to **TEST** your code



Save your Project

Make sure to **SAVE** your work now

Step 1: It's nice to be nice

In this project, you'll make a program to give the user a randomly generated compliment!

Activity Checklist

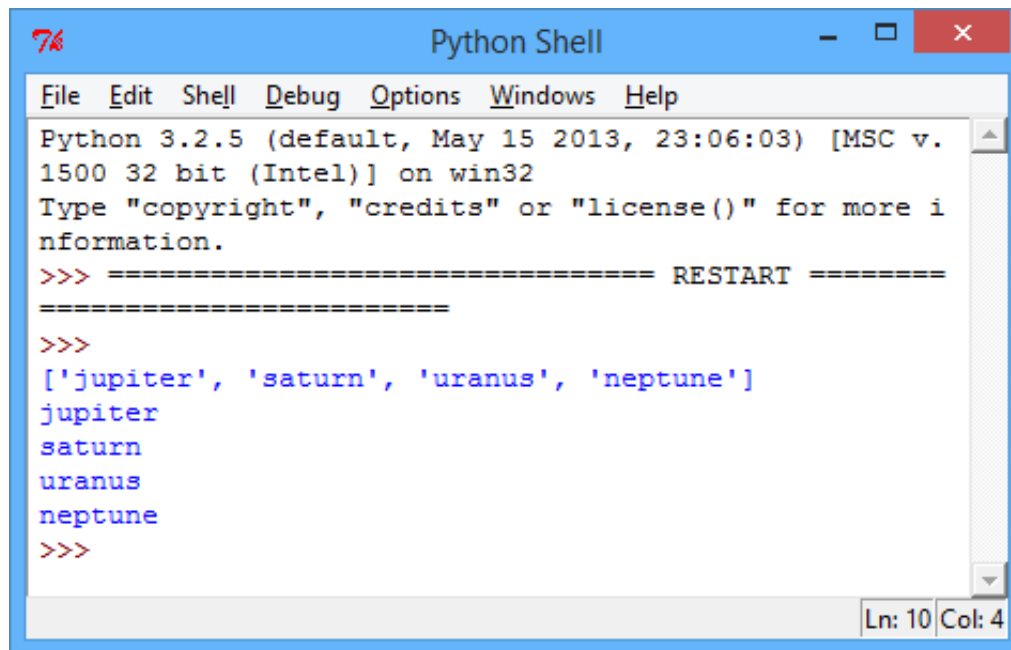
1. In your projects so far, you've used a variable to store a single piece of data, such as a name or a score. But what if you want to store lots of data? In Python, you can use a list to store lots of data in 1 variable:



```
bigPlanets = [ "jupiter" , "saturn" , "uranus" ,  
              "neptune"]
```

This list of text is also known as an array of text. To access items in the list, you just need to know the position of the item. Run this program to give yourself a better idea of how lists work:

```
bigPlanets = [ "jupiter" , "saturn" , "uranus" ,  
              "neptune"]  
print( bigPlanets )  
print( bigPlanets[0] )  
print( bigPlanets[1] )  
print( bigPlanets[2] )  
print( bigPlanets[3] )
```

A screenshot of a Python Shell window titled "Python Shell". The window has a menu bar with "File", "Edit", "Shell", "Debug", "Options", "Windows", and "Help". The main text area shows the following text: "Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v. 1500 32 bit (Intel)] on win32", "Type 'copyright', 'credits' or 'license()' for more information.", a prompt ">>>" followed by "===== RESTART =====", another prompt ">>>" followed by a list of planet names: ['jupiter', 'saturn', 'uranus', 'neptune'], and then the names 'jupiter', 'saturn', 'uranus', and 'neptune' printed on separate lines. The status bar at the bottom right shows "Ln: 10 Col: 4".

```
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.
1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more i
nformation.
>>> ===== RESTART =====
>>>
['jupiter', 'saturn', 'uranus', 'neptune']
jupiter
saturn
uranus
neptune
>>>
```

As you can see, positions start at 0 and not 1, so

`bigPlanets[1]` is "saturn" (the second item) and not "jupiter".

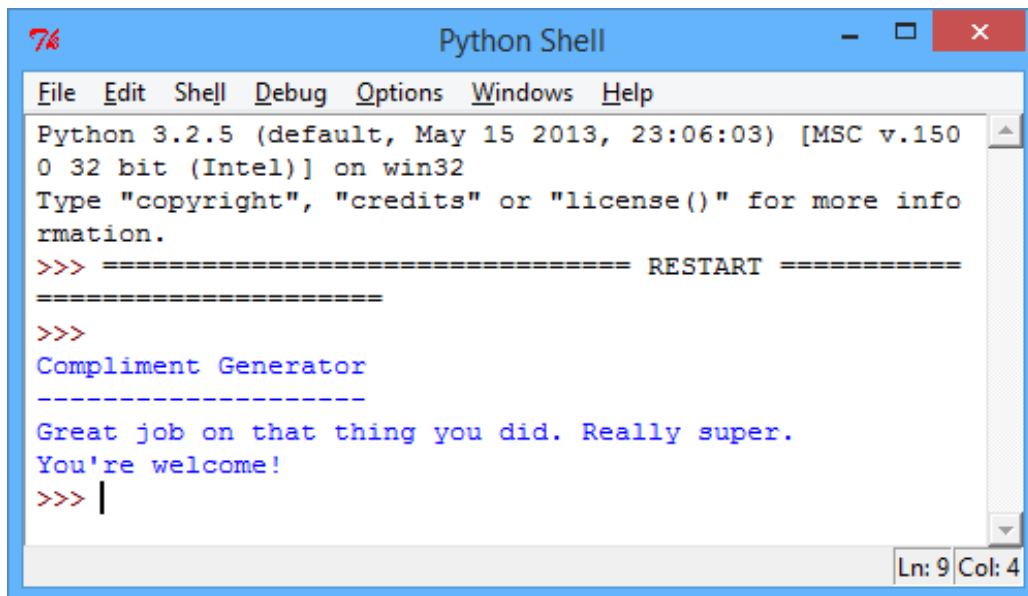
2. You can use a list called `compliments` to store all of the possible compliments for your compliment generator program, and then use `choice(compliments)` to choose a random compliment for the user:

```
from random import *

print("Compliment Generator")
print("-----")

compliments = [ "Great job on that thing you did.
Really super." ,
                "You have really really nice
programming skills." ,
                "You make an excellent human."
              ]

#print a random item in the 'compliments' list
print(choice(compliments))
print("You're welcome!")
```



```
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Compliment Generator
-----
Great job on that thing you did. Really super.
You're welcome!
>>> |
```

3. You could make your compliments a little more interesting, by combining random items from 2 different lists:

```
from random import *

print("Compliment Generator")
print("-----")

adjectives = [ "amazing" , "above-average" ,
"excellent" ]
hobbies = [ "riding a bike" , "programming" , "making a
cup of tea" ]

name = input("What is your name?: ")
print( "Here is your compliment" , name , ":" )

#get a random item from both lists, and add them to the
compliment
print( name , "you are" , choice(adjectives) , "at" ,
choice(hobbies) )
print( "You're welcome!" )
```



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.150
0 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more info
rmation.
>>> ===== RESTART =====
>>>
Compliment Generator
-----
What is your name?: Dylan
Here is your compliment Dylan :
Dylan you are amazing at riding a bike
You're welcome!
>>> |
```

Challenge: Adding more compliments

Try to think of some more compliments, and add them to your program! Remember that you need to add a comma (,) between the items in your lists.

Step 2: Endless compliments

✓ Activity Checklist

1. Using what you know about `while` loops and `if` statements, you could modify your program to keep giving out compliments until the user decides to quit:

```
from random import *

#the program loops as long as this variable is 'True'
running = True

adjectives = [ "amazing" , "above-average" ,
"excellent" ]
hobbies = [ "riding a bike" , "programming" , "making a
```

```

cup of tea" ]

print("Compliment Generator")
print("-----")

name = input("What is your name?: ")

print('''
Menu
    c = get compliment
    q = quit
''')

while running == True:

    menuChoice = input("\n>_").lower()

    #'c' for a compliment
    if menuChoice == 'c':

        print( "Here is your compliment" , name , ":" )

        #get a random item from both lists, and add
        them to the compliment
        print( name , "you are" , choice(adjectives) ,
        "at" , choice(hobbies) )
        print( "You're welcome!" )

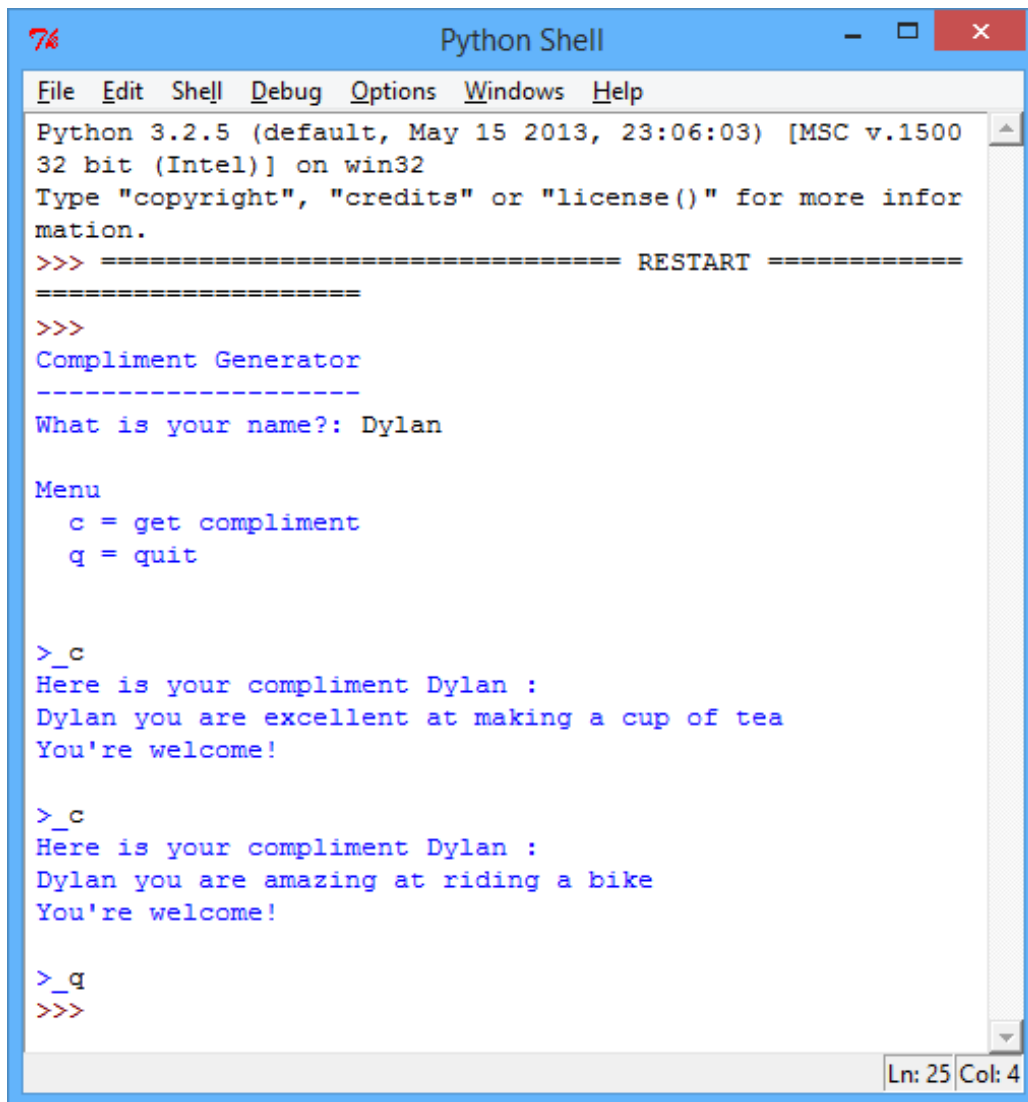
    #'q' to quit
    elif menuChoice == 'q':

        running = False

    else:

        print("Please choose a valid option!")

```



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more infor
mation.
>>> ===== RESTART =====
>>>
Compliment Generator
-----
What is your name?: Dylan

Menu
  c = get compliment
  q = quit

>_c
Here is your compliment Dylan :
Dylan you are excellent at making a cup of tea
You're welcome!

>_c
Here is your compliment Dylan :
Dylan you are amazing at riding a bike
You're welcome!

>_q
>>>
```

Remember that the `while` loop continues to run as long as the variable `running` is set to `True`. If the user inputs `q` to quit, `running` is set to `False`.

Step 3: Personalising compliments

✓ Activity Checklist

1. Your compliment generator is starting to take shape, but it has a problem: what if your user can't ride a bike or make a cup of tea? In that case, your compliments won't be true, and won't cheer them up!
Let's modify your program, so that the user can choose to add or remove items from the `hobbies` list, to allow them to



personalise the compliments they receive:

```
from random import *

running = True
adjectives = [ "amazing" , "above-average" ,
"excellent" ]
hobbies = [ "riding a bike" , "programming" , "making a
cup of tea" ]

print("Compliment Generator")
print("-----")

name = input("What is your name?: ")

print('''
Menu
    c = get compliment
    a = add hobby to list
    d = delete hobby from list
    p = print hobbies
    q = quit
''')

while running == True:

    menuChoice = input("\n>_").lower()

    #'c' for a compliment
    if menuChoice == 'c':

        print( "Here is your compliment" , name , ":" )

        #get a random item from both lists, and add
        them to the compliment
        print( name , "you are" , choice(adjectives) ,
        "at" , choice(hobbies) )
        print( "You're welcome!" )

    #'a' to add a hobby
    elif menuChoice == 'a':
```



```

        itemToAdd = input("Please enter the hobby to
add: ")
        hobbies.append(itemToAdd)

    #'d' to delete a hobby
    elif menuChoice == 'd':

        itemToDelete = input("Please enter the hobby to
remove: ")
        hobbies.remove(itemToDelete)

    #'p' to print the hobbies list
    elif menuChoice == 'p':
        print(hobbies)

    #'q' to quit
    elif menuChoice == 'q':

        running = False

    else:

        print("Please choose a valid option!")

```

As you can see, you can use `append()` to add to a list, and `remove()` to remove an item. Run this program, and personalise the hobbies in the list to suit you. Ask the program for compliments until you're in a good mood!

2. When testing the program above, did you run into any problems? At the moment, your compliment generator crashes if you try and remove a compliment that isn't in the list:



```

>_p
['riding a bike', 'programming', 'making a cup of tea']

>_d
Please enter the hobby to remove: speaking spanish
Traceback (most recent call last):
  File "C:/Users/Rik Cross/Desktop/comp.py", line 44, in <module>
    hobbies.remove(itemToDelete)
ValueError: list.remove(x): x not in list
>>> |

```

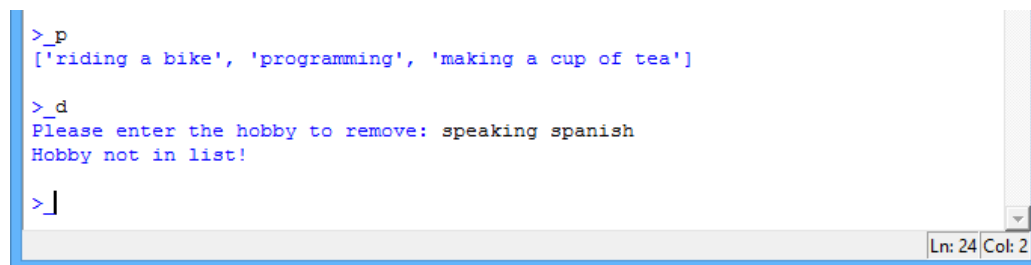
You can fix this problem, by first checking that the item to

remove exists in the list. Replace your code to remove a hobby with this code:

```
#'d' to delete a hobby
elif menuChoice == 'd':

    itemToDelete = input("Please enter the hobby to
remove: ")
    #only remove an item if it's in the list
    if itemToDelete in hobbies:
        hobbies.remove(itemToDelete)
    else:
        print("Hobby not in list!")
```

Now run the program and try to delete a hobby that isn't in the list:

A screenshot of a Python REPL window. The prompt is '>_p' and the input is '['riding a bike', 'programming', 'making a cup of tea']'. The prompt changes to '>_d' and the user enters 'speaking spanish'. The program outputs 'Hobby not in list!'. The prompt returns to '>_'. The bottom right corner shows 'Ln: 24 Col: 2'.

```
>_p
['riding a bike', 'programming', 'making a cup of tea']

>_d
Please enter the hobby to remove: speaking spanish
Hobby not in list!

>_
```

Challenge: Duplicate hobbies

Another problem with the program is that it is possible to add the same hobby more than once:

```
>_a
Please enter the hobby to add: programming

>_p
['riding a bike', 'programming', 'making a cup of tea', 'programming']

>_]
```

Ln: 23 Col: 2

Can you fix this problem, so that a hobby can only be added if it isn't already in the list:

```
if itemToAdd not in hobbies:
    #add code here...
```

Challenge: Pet naming service

Write a program to help a new pet owner to name their pet:



```
Python Shell
File Edit Shell Debug Options Windows Help
Python 3.2.5 (default, May 15 2013, 23:06:03) [MSC v.1500
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more inform
ation.
>>> ===== RESTART =====
>>>
Pet naming service
-----
You should name your new pet Martin
>>> |
```

Your program could:

- ☐ allow the user to add and remove names from the list;
- ☐ give different names for male and female pets, or different types of animal;
- ☐ ask the user how many names they need, in case they have more than 1 pet to name.