Homework Problems

2-1. Simple Message: Assign a message to a variable and then print that message

In [1]:
```
message = "this is a message created in python"
print(message)
```

```
this is a message created in python
```

2-2 Simple Messages: Assign a message to a variable and then print that message. Then change the value of the variable to a new message, and print the new message

In [2]:
```
message = "this is a message created in python"
print(message)
message = "i now changed the value of the message variable"
print(message)
```

```
this is a message created in python
i now changed the value of the message variable
```

2-3. Personal Message: Use a variable to represent a person's name, and print a message to that person. Your message should be simple, such as, "Hello Eric, would you like to learn some Python today?"

In [3]:
```
# using f-strings!

name = "Andrew"
print(f"Hello {name}, how are you doing today?")
```

```
Hello Andrew, how are you doing today?
```

2-4. Name Cases: Use a variable to represent a person's name and then print that peron's name in lowercase, uppercase, and title case.

In [4]:
```
name = "aNdRew"
print(f"lowercase: {name.lower()} \nuppercase: {name.upper()} \ntitle case: {
```

```
lowercase: andrew
uppercase: ANDREW
title case: Andrew
```

2-5. Famous Quote: Find a quote from a famous person you admire. Print the quote and the name of its author. Your output should look somethink like the following, including quotation marks:

```
    Albert Einstein once said, "A person who never made a mistake nev
er tried anything new"
```

In [5]:
```
author = "Neil Armstrong"
quote = "One small step for man, one giant leap for mankind"
print(f'{author} once said, "{quote}"')
```

```
Neil Armstrong once said, "One small step for man, one giant leap for mankin
d"
```

2-6. Famous Quote 2: Repeat Exercise 2-5, but this time, represent the famous person's name using a variable called `famous_person`. Then compose your message and represent it with a new variable called `message`. Print your message

In [6]:
```
famous_person = "Neil Armstrong"
message = f'{famous_person} once said, "One small step for man, one giant lea
print(message)
```

```
Neil Armstrong once said, "One small step for man, one giant leap for mankin
d"
```

2-7. Stripping Names: Use a variable to represent a person's name, and include some whitespace characters at the beginning and end of the name. Make sure you use each character combination, "\t" and "\n", at least once. Print the name using each of the three stripping functions, `lstrip()`, `rstrip()`, and `strip()`.

In [7]:
```
name = "\t     Andrew      \n"
print(f"lstrip(): {name.lstrip()} \nrstrip(): {name.rstrip()} \nstrip(): {nam
```

```
lstrip(): Andrew

rstrip():            Andrew
strip(): Andrew
```

2-8. Number Eight: Write addition, subtraction, multiplication, and division operations that each result in the number 8. Be sure to enclose your operations in `print()` calls to see the results. You should create four lines that look like this:

```
print(5+3)
```

Your output should simply be four lines with the number 8 appearing once on each line.

In [8]:
```
# demonstrating mathematical operations in python...

print(4+4)
print(10-2)
print(4*2)
print(int(8/1))
```

```
8
8
8
8
```

2-9. Favorite Number: Use a variable to represent your favorite number. Then, using that variable, create a message that reveals your favorite number. Print that message.

In [9]:
```
favorite_number = 1
print(f"my favorite number is {favorite_number}:")
```

```
my favorite number is 1:
```

2-10. Adding Comments: Choose two of the previous problems and at at least one comment to each.

2-11. Zen of Python: Enter `import this` and skim through the additional principles.

In [10]:
```
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.
```