Try It Yourself-Chapter 2

2-2: Simple Messages

Assign a message to a variable, and print that message. Then change the value of your variable to a new message, and print the new message.

In [3]:
```
Msg="Hello World"
print(Msg)
```

```
Hello World
```

In [4]:
```
Msg="Hello World! This is New Message"
print(Msg)
```

```
Hello World! This is New Message
```

2-3. Personal Message:

Store a person's name in a variable, and print a message to that person. Your message should be simple, such as, "Hello Eric, would you like to learn some Python today?"

2-4. Name Cases: Store a person's name in a variable, and then print that person's name in lowercase, uppercase, and titlecase.

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 something like the following, including the quotation marks: Albert Einstein once said, "A person who never made a mistake never tried anything new."

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

2-7. Stripping Names: Store 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 once, so the whitespace around the name is displayed. Then print the name using each of the three stripping functions, lstrip(), rstrip(), and strip().

In [5]:
```
Person_Name='Eric'
Message= f"Hello {Person_Name} , Would you like to learn some Python today?"
Message
```

Out[5]:
```
'Hello Eric , Would you like to learn some Python today?'
```

In [6]:
```
print(Person_Name.lower())
print(Person_Name.upper())
print(Person_Name.title())
```

```
eric
ERIC
Eric
```

In [7]:
```
quote = "A person who never made a mistake never tried anything new."
author = "Albert Einstein"
print(f'{author} once said, "{quote}" ')
```

```
Albert Einstein once said, "A person who never made a mistake never tried an
ything new."
```

In [8]:
```
famous_person = "Albert Einstein"
message = famous_person + ' once said, "' + quote + '"'
print(message)
```

```
Albert Einstein once said, "A person who never made a mistake never tried an
ything new."
```

In [12]:
```
name = "\t   John Doe   \n"
print("Original Name: '" + name+ "'")
print("Using lstrip(): '" + name.lstrip() + "'")
print("Using rstrip(): '" + name.rstrip() + "'")
print("Using strip(): '" + name.strip() + "'")
```

```
Original Name: '           John Doe
'
Using lstrip(): 'John Doe
'
Using rstrip(): '           John Doe'
Using strip(): 'John Doe'
```

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 statements 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.

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

In [13]:
```
print(2+6)
print(12-4)
print(16/2)
print(2*4)
```

```
8
8
8.0
8
```

In [17]:
```
Favourite_Number=3
Msgg= "My Favourite Number is"+ " "+ str(Favourite_Number)
print(Msgg)
```

```
My Favourite Number is 3
```

2-11. Zen of Python: Enter import this into a Python terminal session and skim through the additional principles

In [21]:
```
>>> 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 [ ]:
```
```