In [ ]:

```
Variables

Variables are containers for storing data values.
```

In [1]:

```
x = 5
y = "John"
print(x)
print(y)
```

```
5
John
```

In [2]:

```
x = 4
x = "Sally"
print(x)
```

```
Sally
```

In [ ]:

```
Casting

If you want to specify the data type of a variable, this can be done with casting.
```

In [6]:

```
x = str(3)
y = int(3)
z = float(3.8)
print(type(x))

print(x)
print(y)
print(z)
```

```
<class 'str'>
3
3
3.8
```

In [ ]:

```
Get the Type

You can get the data type of a variable with the type() function.
```

In [4]:

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

```
<class 'int'>
<class 'str'>
```

In [ ]:

```
Single or Double Quotes?

String variables can be declared either by using single or double quotes
```

In [5]:

```
x = "John"
print(x)
#double quotes are the same as single quotes:
x = 'John'
print(x)
```

```
John
John
```

In [ ]:

```
Case-Sensitive

Variable names are case-sensitive.
```

In [6]:

```
a = 4
A = "Sally"

print(a)
print(A)
```

```
4
Sally
```

In [8]:

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"


print(myvar)
print(my_var)
print(_my_var)
print(myVar)
print(MYVAR)
print(myvar2)
```

```
John
John
John
John
John
John
```

In [1]:

```
2myvar = "John"
my-var = "John"
my var = "John"


#This example will produce an error in the result
```

```
  File "C:\Users\SPC\AppData\Local\Temp/ipykernel_2200/143763948.py", line 1
    2myvar = "John"
     ^
SyntaxError: invalid syntax
```

In [2]:

```
x, y, z = "Orange", "Banana", "Cherry"

print(x)
print(y)
print(z)
```

```
Orange
Banana
Cherry
```

In [10]:

```
x = y = z = "Orange"

print(x)
print(y)
print(z)
```

```
Orange
Orange
Orange
```

In [1]:

```
fruits = ["apple", "banana", "cherry"]
x, Y, z = fruits

print(x)
print(Y)
print(z)
```

```
apple
banana
cherry
```

In [12]:

```
x = "awesome"
print("Python is " + x)
```

```
Python is awesome
```

In [13]:

```
x = "Python is "
y = "awesome"
z = x + y
print(z)
```

```
Python is awesome
```

In [14]:

```
x = 5
y = 10
print(x + y)
```

```
15
```

In [ ]:

```
Global Variables

Variables that are created outside of a function (as in all of the examples above) are know

Global variables can be used by everyone, both inside of functions and outside.
```

In [15]:

```
x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

```
Python is awesome
```

In [16]:

```
x = "awesome"

def myfunc():
  x = "fantastic"
  print("Python is " + x)

myfunc()

print("Python is " + x)
```

```
Python is fantastic
Python is awesome
```

In [ ]:

```
The global Keyword

Normally, when you create a variable inside a function, that variable is local, and can onl
used inside that function.

To create a global variable inside a function, you can use the global keyword.
```

In [17]:

```
def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

```
Python is fantastic
```

In [18]:

```
x = "awesome"

def myfunc():
  global x
  x = "fantastic"

myfunc()

print("Python is " + x)
```

```
Python is fantastic
```