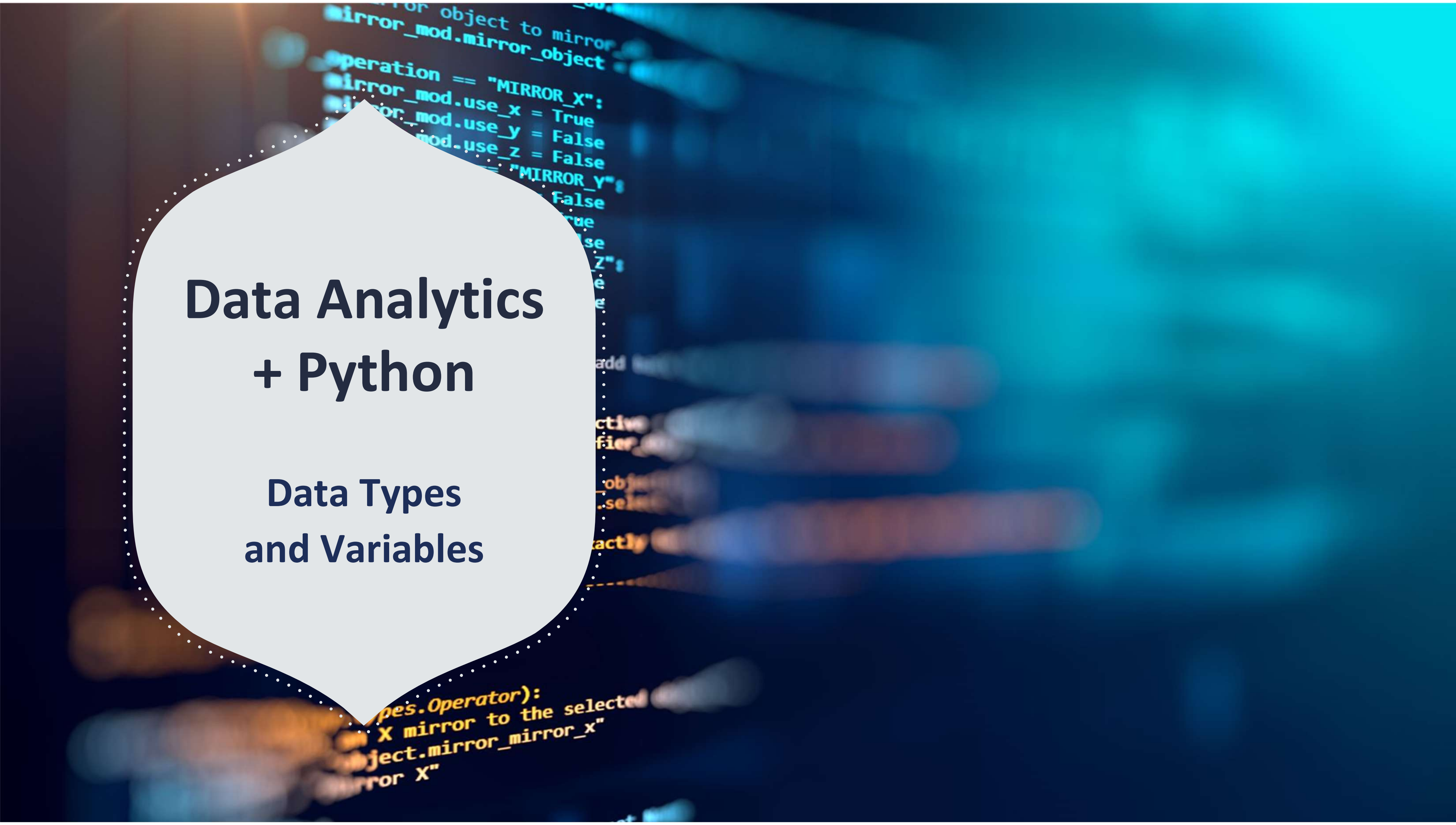

Data Analytics + Python

Data Types and Variables

Variable Names

- Containers for storing data values
- No command for declaring a variable
- YOU create it as soon as you assign a value to it!

```
x = 8
y = brothers
print(x)
print(y)
```

Multi Word Variable Names

```
fruits = [“apple”, “banana”, “cherry”]
x = y = z = fruits
print(x)
print(y)
print(z)
```

```
x, y, z = “Orange”, “Banana”, “Cherry”

print(x, y, z)

g = h = i = “Orange”
print(g, h, i)
```

Output and Variables

```
x = “awesome”
print(“Python is “ + x)

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

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

x = 5
y = “John”
Print(x + y)
```

Global Variables

It holds its value throughout the lifetime of the program.

```
x = "awesome"

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

myfunc()
```

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

myfunc()
```

Data Types

x = “Hello World”	string (str)
x = 20	integer (int)
x = 20.5	float
x = [“apple”, “banana”, “cherry”]	list
x = (“apple”, “banana”, “cherry”)	tuple
x = range(5)	range
x = {“name” : “John”, “age” : 36}	dictionary (dict)
x = True	boolean

You can put the data type in front if you want to specify

```
x = str("hello, world!")
print(x)
y = int(20.5)
print(y)
z = float(20)
print(z)
```

Also known as Casting!

- You can check the data type of a value by writing this:

```
x = 1
y = 4.5
print(type(x))
print(type(y))
```

Strings

- are surrounded by either single or double quotation marks
- You can assign a variable to them a = “Hello”

  print(a)
- Multi-line strings are designated with “”””” /”””” or ‘’’ /’’’

Finding the length of a string

```
a = “Hello World!”
print(len(a))
```

Counting in Python starts with 0
It includes ALL spaces inside the “ “

Exemption: when you are counting backwards.

Getting the character position of a string

```
a = “Hello World!”
print(a[1])
print(a[-1])
```

Try with another number!

Checking a string for a sub-string

```
txt = “The best class in Saint Louis”

if “best” in txt:
    print(“Yes, ‘best’ is present.”)
```

Slicing a string:

Return a range of characters

```
b = “Hello World!”
print(b[2:5])
```

Ranges start at the number on the left but doesn’t include the number on the right.

No number on the left? Means count starts at the beginning.

Slicing a string

From the start

```
b = “Hello World!”
print(b[:5])
```

Slicing to the end

```
b=“Hello World!”
print(b[2:])
```

Output starts at position 2

Negative Indexing

```
b=“Hello World!”
print(b[-5:-2])
```

Count back 5
DO NOT include the last 2

Replace in a string

```
c = “Hello World!”
print(c.replace(“H” , “J”))
```

Concatenation

```
a = "Hello"
b = "World"
c = a + b
print(c)
```

Formatting a String

```
age = ”36”
txt = "My name is John, I am " + age
print(txt)


age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))


quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))
```

Logical Operators

Operator	Name
==	Equal
!=	Not equal
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

Booleans represent one of two values: True or False

- You often need to know if an expression is true or false

```
print(10 > 9)
```

- You can evaluate any expression and get one of two answers

```
print(10 == 9)
```

- When you compare two values, the expression is evaluated and Python returns the BOOLEAN answer.

```
print(10 < 9)
```