IT Academy - Data Science Itinerary

Sprint 2 - Introduction to Python

Assignment by: Kat Weissman

Python Learning Objectives:

- Create variables
- Math operations
- Transform with casting
- String techniques
- Booleans

Jupyter Notebook Learning Objectives:

- Use Jupyter Notebook to practice python
- Use Markdown Language
  - Titles
  - Lists
  - Font styles
  - Images
- Export the Notebook as pdf and as html.
- Install Nbextensions the Jupyter Notebook

Recommended learning resources:

- https://www.w3schools.com/python/default.asp
- https://learn.datacamp.com/courses/intro-to-python-for-data-science
- https://cibernarium.barcelonactiva.cat/web/guest/ficha-actividad?activityId=1019541

Getting Started

In [127…

```
me = "Kat"
myDog = "Logan"
```

```
myDogBirthYear = 2013
currentYear = 2021
myDogAge = currentYear - myDogBirthYear
myDogBreed = "galgo"
myString = myDog + " is " + str(myDogAge) + " years old, and he is a " + myDogBreed + "."

print ("I am", me + ".")
print ("My dog's name is", myDog + ".")
print (myString)
```

```
I am Kat.
My dog's name is Logan.
Logan is 8 years old, and he is a galgo.
```

Logan likes to play with stuffed animal squeaky toys. His favorite toys are a raccoon 🦡and a crocodile 🐊

This cell is automatically translated from english to spanish by Google translate using the Nbextension nbTranslate.

A Logan le gusta jugar con juguetes chirriantes de animales de peluche. Sus juguetes favoritos son un mapache y un cocodrilo.

Esta celda es traducida automáticamente del inglés al español por el traductor de Google utilizando Nbextension nbTranslate.

Nbextensions

I installed Nbextensions using the following command in the terminal.

```
conda install -c conda-forge jupyter_contrib_nbextensions
```

I enabled the following Nbextensions:

- Variable Inspector
- ExecuteTime
- jupyter-js-widgets/extension
- Nbextensions dashboard tab
- table_beautifier
- contrib_nbextensions_help_item
- Nbextensions edit menu item
- nbTranslate

Python Variables & Math Operations

In [128…

```
#assigning variables, performing some simple math, and displaying the results.
x = 3.4
y = 7
sumXY = x + y
diffXY = x - y
productXY = x * y
ratioXY = x / y

print ("Variable x is:", x)
print ("Variable y is:", y)
print ("The sum of x and y is:", sumXY)
print ("The difference of x and y is:", diffXY)
print ("The product of x and y is:", productXY)
print ("The ratio of x and y is:", ratioXY)
print ("x raised to the power of y is:", x**y)
```

```
Variable x is: 3.4
```

```
Variable y is: 7
The sum of x and y is: 10.4
The difference of x and y is: -3.6
The product of x and y is: 23.8
The ratio of x and y is: 0.4857142857142857
x raised to the power of y is: 5252.335014399999
```

In [129…

```
#checking the type of variables
print (type(x))
print (type(y))

#variables from earlier cells can be used as long as the code has been executed.
print (type(myDog))
```

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

Transforming variables with typecasting

The + operation can be used on numbers or strings. It will sum numbers or it will concatenate strings. An error will occur if you use the operation on a string with a number unless you change the type of variable. An example of a TypeError and successful typecasting is shown below.

In [130…

```
#this code produces a TypeError since a number cannot be concatenated with a string.
myDog + "is" + myDogAge + "years old."
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-130-d52a6a05c0ab> in <module>
      1 #this code produces a TypeError since a number cannot be concatenated with a string.
----> 2 myDog + "is" + myDogAge + "years old."

TypeError: can only concatenate str (not "int") to str
```

In [131…

```
#typecast a number to a string in order to concatenate.
myDog + " is " + str(myDogAge) + " years old."
```

Out[131…

```
'Logan is 8 years old.'
```

Even though the previous codeblock executed succesfully, the variable is not permanently changed with typecasting unless it is overwritten or saved as a new variable.

In [132…
```
#confirm that the variable type was not changed in the previous code.
type(myDogAge)
```

Out[132…
```
int
```

In [133…
```
#save the variable as a new type
myDogAge = str(myDogAge)
```

In [134…
```
#check the variable type to confirm the change
type(myDogAge)
```

Out[134…
```
str
```

Math cannot be performed on strings unless they are typecasted to a number as shown below.

In [135…
```
#This code produces an error because myDogAge is now a string.
myDogAge + y
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-135-9b8ede084382> in <module>
      1 #This code produces an error because myDogAge is now a string.
----> 2 myDogAge + y

TypeError: can only concatenate str (not "int") to str
```

In [ ]:
```
#multiplying a string by the integer y concatenates the string to itself y times.
myDogAge * y
```

In [ ]:
```
#typecast the variable to a float, multiply the numbers and the result will be a float.
float(myDogAge) * y
```

In [136…
```
#typecast the variable to an int and the result will be an int.
int(myDogAge) * y
```

Out[136…
```
56
```

In [137…
```
#myDogAge is still a string, because it hasn't been saved as a variable.
type(myDogAge)
```

Out[137…
```
str
```

String Techniques

Some string techniques have already been demonstrated, like concatenation. More techniques are demonstrated in the code blocks below.

In [138…
```
#check the length of a string using len().
print (myDog, "is", len(myDog), "characaters long.")

#print the characters at a certain index of the string using [].
print (myDog, "has the letter", myDog[0], "as the first character.")
print (myDog, "has the letter", myDog[-1], "as the last character.")
print (myDog, "has the series of letters", myDog[1:4], "as the second to fourth characters.")
```

```
Logan is 5 characaters long.
Logan has the letter L as the first character.
Logan has the letter n as the last character.
Logan has the series of letters oga as the second to fourth characters.
```

In [139…
```
#print each character of a string individually using a for loop.
for char in myDog: print(char)
```

```
L
o
g
a
n
```

In [140…
```
#check presence of a character in a string. The result will be a boolean.
print ("The letter 'L' as uppercase is present in", myDog + ":")
print ('L' in myDog)


#Uppercase and lowercase are checked exactly.
print ("The letter 'l' as lowercase is present in", myDog + ":")
print ('l' in myDog)
```

```
#A series of characters can also be checked.
print ("The letters 'oga' as lowercase are present in", myDog + ":")
print ('oga' in myDog)
```

```
The letter 'L' as uppercase is present in Logan:
True
The letter 'l' as lowercase is present in Logan:
False
The letters 'oga' as lowercase are present in Logan:
True
```

String Methods

In [141…

```
print(myString)
# upper() converts all characters to uppercase.
print(myString.upper())
# lower() converts all characters to lowercase.
print(myString.lower())
# title() converts the first character of each word to uppercase.
print(myString.title())
```

```
Logan is 8 years old, and he is a galgo.
LOGAN IS 8 YEARS OLD, AND HE IS A GALGO.
logan is 8 years old, and he is a galgo.
Logan Is 8 Years Old, And He Is A Galgo.
```

In [142…

```
# split() returns a list of strings that is separated at the split argument.
print(myString.split(','))
myWordList = myString.split(' ')
print(myWordList)

# join() combines the elements of an iterable as a string with a separator.
print(' '.join(myWordList))
print('-'.join(myWordList))
```

```
['Logan is 8 years old', ' and he is a galgo.']
['Logan', 'is', '8', 'years', 'old,', 'and', 'he', 'is', 'a', 'galgo.']
Logan is 8 years old, and he is a galgo.
Logan-is-8-years-old,-and-he-is-a-galgo.
```

Booleans

In [143…

```
#returns true or false upon evaluating the statement.
print (x > y)
```

```
print (x < y)
print (x == y)
print (x != y)
```

```
False
True
False
True
```

In [144…

```
#When the if statements are true, the code will be executed.
if (x > y): print (x, "is greater than", y)
if (x < y): print (x, "is less than", y)
if (x == y): print (x, "equals", y)
if (x != y): print (x, "does not equal", y)
```

```
3.4 is less than 7
3.4 does not equal 7
```

In [145…

```
#booleans evaluate most variables to true, unless they are 0 or empty.
print (bool(x))
print (bool (y))
print (bool (myDog))
print (bool (''))
print (bool (0))
```

```
True
True
True
False
False
```

In [146…

```
#booleans are useful for checking conditions, then using them to execute some code if the condition is true.

#typecasting the variable to a float
x = float(x)

# math methods will be used for numbers
if (isinstance(x,int) | isinstance(x,float)):
    print (x * 8)
    print (x + x)

#typecasting the variable to a string
x = str(x)

# string methods will be used for strings
```

```
if (isinstance(x, str)):
    print (x * 8)
    print (x + x)
```

```
27.2
6.8
3.43.43.43.43.43.43.43.4
3.43.4
```

In [147…

```
# when the statements evaluate to False, then the code will not run, and there will be no output unless
# there is an else statement included.

#typecasting the variable to a string
x = str(x)

# math methods will be used for numbers
if (isinstance(x,int) | isinstance(x,float)):
    print (x * 8)
    print (x + x)
else:
    print ((isinstance(x,int) | isinstance(x,float)))

#typecasting the variable to a float
x = float(x)

# string methods will be used for strings
if (isinstance(x, str)):
    print (x * 8)
    print (x + x)
else:
    print (isinstance(x, str))
```

```
False
False
```

In [ ]: