Simon Ilisevich

Data Bootcamp, Prof. Waugh

Code Practice #1

Question 1:

```
In [1]:
```

```
2+5
```

```
Out[1]:

7
```

Adds two values

```
In [42]:
```

```
2 + 5
```

```
Out[42]:

7
```

Same as above but with extra spaces, i.e. spaces make no difference in basic functions

```
In [4]:
```

```
2*5
```

```
Out[4]:

10
```

Multiplies values

```
In [41]:
```

```
2/5
```

```
Out[41]:

0.4
```

Divides values

```
In [40]:
```

```
2**5
```

```
Out[40]:

32
```

" ** " serves as 'to the power of', same as 2^5 in Excel

Question 2:

In [18]:

```
x = 7
x = x + 3
```

In [19]:

```
print(x)
```

```
10
```

The first input assigns the number 7 to x, so whenever you type "x" it's the same as typing 7. When assigning (or in this case re-assigning) values in Python, the right-hand side of the equation is evaluated first, thus using x = 7 from the previous line, 7 + 3 = 10

Question 3:

In [22]:

```
x = 3
print(x)
```

```
3
```

In [24]:

```
y = x
print(y)
```

```
3
```

In [27]:

```
x = 10
print(x)
```

```
10
```

In [28]:

```
print(y)
```

```
3
```

Question 4:

In [45]:

```
x = 3
x = x/2
print(x)
```

```
1.5
```

```
In [48]:
```

```
y = 'abc'
print(y)
```

```
abc
```

```
In [50]:
```

```
z = y + y
print(z)
```

```
abcabc
```

```
In [51]:
```

```
print (x, z)
```

```
1.5 abcabc
```

This code runs without error, with the result shown in the final output

Question 5:

```
In [52]:
```

```
x = 3
x = x/2
print(x)
```

```
1.5
```

```
In [53]:
```

```
y = 'abc'
print(y)
```

```
abc
```

```
In [59]:
```

```
z = x + y
print (x, z)
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-59-57024a8455a0> in <module>()
----> 1 z = x + y
      2 print (x, z)

TypeError: unsupported operand type(s) for +: 'float' and 'str'
```

```
In [55]:
```

```
whos
```

```
Variable   Type     Data/Info
-----------------------------
x          float    1.5
y          str      abc
z          str      abcabc
```

As shown above, this code runs with error because of z=x+y - 'x' is a float but 'y' is a string, so they can't be added. Question 4 worked fine because z was adding two strings.

Question 6:

```
In [60]:
```

```
x = 3
y = 24
z = y/x
print(x, y, z, sep='|')
```

```
3|24|8.0
```

Runs without error

Question 7:

```
In [61]:
```

```
x = 3
y = '24'
z = y/x
print(x, z)
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-61-54c746c62a46> in <module>()
      1 x = 3
      2 y = '24'
----> 3 z = y/x
      4 print(x, z)

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

```
In [62]:
```

```
whos
```

```
Variable   Type     Data/Info
-----------------------------
x          int      3
y          str      24
z          float    8.0
```

This code runs with an error for the same reason as Question (5) - z is attempting to divide two different 'types', as In [62] shows

Question 8:

In [64]:

```
x = "I am a #string" # Whoa, a string!
x
```

Out[64]:

```
'I am a #string'
```

Runs without error - everything after a '#' in a code line, unless part of the string between quotation marks, just creates a 'comment'

Question 9:

In [65]:

```
x = [1, 2, 3]
y = [42, 43]
z = x + y
print (z)
```

```
[1, 2, 3, 42, 43]
```

Runs without error - just combining two lists

Question 10:

In [67]:

```
x = [1, 2, 3]
y = 42
z = x + y
```

```
--------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-67-6ce2df3c1c18> in <module>()
      1 x = [1, 2, 3]
      2 y = 42
----> 3 z = x + y

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

In [68]:

```
whos
```

```
Variable   Type    Data/Info
----------------------------
x          list    n=3
y          int     42
z          list    n=5
```

Yes, this code runs with an error because z is once again attempting to combine two different types, in this case a list and an integer

Question 11:

In [70]:

```
x1 = 12
x2 = 12.0
x3 = '12.0'
x4 = [12]
x5 = [12, 12.0, '12.0']
```

In [71]:

```
whos
```

```
Variable   Type     Data/Info
-----------------------------
x          list     n=3
x1         int      12
x2         float    12.0
x3         str      12.0
x4         list     n=1
x5         list     n=3
y          int      42
z          list     n=5
```

In [71] explains what type x1-x5 are (described in detail in the next question)

Question 12:

In [72]:

```
type(42)
```

Out[72]:

```
int
```

'int' = integer, i.e. a whole real number

In [73]:

```
type(42.0)
```

Out[73]:

```
float
```

'float' = floating point real value, i.e. necessarily a real number which the decimal point signifies, and this case also a whole number

In [74]:

```
type('42.0')
```

Out[74]:

```
str
```

str = string

In [75]:

```
type("42.0")
```

Out[75]:

```
str
```

Same as In [74] - doesn't matter if you use single or double quotation marks

In [76]:

```
type("""42.0""")
```

Out[76]:

```
str
```

Still a string, doesn't matter how many quotation markets you use

In [84]:

```
type([1, 2])
```

Out[84]:

```
list
```

'list' = list/series of numbers

In [80]:

```
type([1] + [2])
```

Out[80]:

```
list
```

Still a list - numbers can be separated so long as they're within brackets

In [81]:

```
type(1 + 2)
```

Out[81]:

```
int
```

Without brackets, this is just adding two integers and thus is an integer

In [85]:

```
type(print)
```

Out[85]:

```
builtin_function_or_method
```

'builtin_finction_or_method' - a built in command functio of Python

Question 13:

In [86]:

```
type(float(str(int('1234'))))
```

Out[86]:

```
float
```

Same logic as Question 2 - Python evaluates inputs from right to left, so float was the final type which we labeled

Question 14:

In [87]:

```
type(int(float('12.34')))
```

Out[87]:

```
int
```

In this case, int was the final type, and although 12.34 isn't an integer, we told Python to recognize it as one

Question 15:

In [91]:

```
len([1234])
```

Out[91]:

```
1
```

Lists, denoted by brackets, are a series of numbers. When applying a length to a list, it calculates the number of whole numbers, separated by commas, in that list. Since 1234 is a whole number and the only one between the brackets, 'len' is telling us that the length of the list is 1.

```
In [93]:
```

```
len("1234")
```

```
Out[93]:

4
```

Strings, denoted by quotation marks, are a series of characters. Since there are four characters in the list, the 'len' is measuring the number of characters in the list, of which there are four.

```
In [97]:
```

```
len(1234)
```

```
---------------------------------------------------------------------
-------
TypeError                                 Traceback (most recent cal
l last)
<ipython-input-97-924e0f4453b7> in <module>()
----> 1 len(1234)

TypeError: object of type 'int' has no len()
```

1234 is an integer, which you can't measure the length of

Question 16:

```
In [99]:
```

```
x =[]
type(x)
```

```
Out[99]:

list
```

```
In [100]:
```

```
len(x)
```

```
Out[100]:

0
```

Question 17:

In [102]:

```
x = 'abcde'
x
```

Out[102]:

```
'abcde'
```

In [105]:

```
list(str(x))
```

Out[105]:

```
['a', 'b', 'c', 'd', 'e']
```

Question 18:

a.

In [106]:

```
x = 1234
float(x)
```

Out[106]:

```
1234.0
```

b.

In [116]:

```
str(x)
```

Out[116]:

```
'1234'
```

In [117]:

```
list(str(x))
```

Out[117]:

```
['1', '2', '3', '4']
```

Question 19:

In [118]:

```
x = 'luke, i am your father'
```

In [120]:

```
x.title()
```

Out[120]:

```
'Luke, I Am Your Father'
```

Question 20:

In [123]:

```
x = "How many characters and words are in this string?"
```

a.

In [126]:

```
len(str(x))
```

Out[126]:

```
49
```

b.