Rex Jones II

How To Work With Python Numbers

Python Video = https://youtu.be/NNLj3nP2o2k

Working With Numbers

In this session, we are going to work with numbers. For Python, we can convert numbers or use a function. Let's start by converting a number to a string.

If we want to concatenate an integer such as cost_of_water that is assigned 3. We can print the value but need the string form. For example, print("Water cost ") then the variable ("Water cost " + cost_of_water + " Dollars"). Do you see how cost_of_water has a brown background? That's because there is a problem.

```
cost_of_water = 3
print("Water cost " + cost_of_water + " Dollars")
```

Run and the console shows TypeError: can only concatenate str (not "int") to str. This exception has 2 parts. First part is the type of error and the second part is the message. TypeError means there is a data type mismatch. The message is saying, we can only concatenate string to string and not use integer.

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

We must convert the integer to a string. You know how a number can be an integer or floating-point. A similar exception shows up if we update cost_of_water from 3 to 3.25. Run again and the console returns (not float).

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

We convert the value to a string by passing in cost_of_water to str(). This time in the console we see Water cost 3.25 Dollars.

```
cost_of_water = 3.25
print("Water cost " + str(cost_of_water) + " Dollars")
```

```
Water cost 3.25 Dollars
```

The int and float functions help when converting a string to a number. Whenever we use input(“”), Python makes all of the information a string. So if I write “How Much Does Candy Cost? ”. The information inside the quotes is a string. Also, the information we type as a user will be a string. Assign this value to price =. Watch what happens when I print(price).

ValueError: invalid literal for int() with base 10. ValueError is the type of error meaning it is the right data type but the value is inappropriate. The message “invalid literal for int() with base 10” means the int() method will not allow us to pass in a floating-point number represented as a string.

```
ValueError: invalid literal for int() with base 10:
```

We can keep the value in quotes but it has to be an integer like 12. Run and we no longer see the error or exception.

```
number = "12"
print(int(number))
```

```
12
```

We can also change the value from a string to a floating-point number. By changing to a floating-point, we convert 12.97. Remove the quotes then add back .97. Run and the console only shows 12 which is a whole number.

```
number = 12.97
print(int(number))
```

```
12
```

There are more math functions we can use with our numbers. We can use a round function. The purpose of round is to return the nearest integer value. Just replace int() with round() and the console shows 13 because it rounds up.

```
number = 12.97
print(round(number))
```

```
13
```

We can also use abs. This is short for absolute which returns the absolute value of a number. It always return a positive number. If I make this a negative -12.97 then play and we see positive 12.97 because 12.97 is the absolute value of negative 12.97.

```
number = -12.97
print(abs(number))
```

```
12.97
```

We can also use another function like pow(). This is pow and it receives 2 arguments. The 1st argument is a number like 10 and the 2nd argument is a number like 3. This reads 10 to the power of 3. The answer should be 1,000. Just like our Arithmetic Operator session when we used an exponent to calculate 10 to the power of 3. print() then Run and we see 1000.

```
print(pow(10, 3))
```

```
1000
```

Another function is min. The purpose of min is to return the smallest argument. It will return the smallest of these 2 arguments (10 and 3). The console returns 3. max is the opposite of min. It will return the largest of 10 and 3. We see 10.

```
print(min(10, 3))
```

```
3
```

```
print(max(10, 3))
```

```
10
```

In addition to these functions, we have some complex functions from the math module. The math module allows us to access reusable functions math after we import *. This statement brings in outside code to our file so we can perform a math operation. You can find a list of the math module functions online. For example, sqrt(25). This is the square root function. It returns the square root of a number. Square root is the value of a number multiplied by itself. The square root of 25 is 5 because 5 times 5 is 25. The console shows 5.0.

```
from math import *
print(sqrt(25))
```

```
5.0
```

Contact Info

- ✔ Email Rex.Jones@Test4Success.org
- ✔ YouTube https://www.youtube.com/c/RexJonesII/videos
- ✔ Facebook https://facebook.com/JonesRexII
- ✔ Twitter https://twitter.com/RexJonesII
- ✔ GitHub https://github.com/RexJonesII/Free-Videos
- ✔ LinkedIn https://www.linkedin.com/in/rexjones34/