CS215 Ch 3F Lab – Using Type Decimal **Name:** JAke

- F-strings can be used to nicely align output. Here's the rules.
  - < means L Align
  - <8 means L Align in a field width of 8

```
a = 1.23
print(f'{a:<8}***')
```

  - > means R Align
  - >10 means R align in a field width of 10

```
a = 1.23
print(f'{a:>10f}***')
print(f'{a:>10.4f}***')
```

1.23000***
1.2300***

- These ideas can be helpful when lining up money amounts. Create a Python script called "decimals.py" and use the above formatter rules to print the following out.

```
1325.48
   5.17
 328.48
```

****

- The rest of this lab will help us explore floating point numbers. We open with an example. In the blank below, add code so that the temperature variable is printed out with 20 decimal places of precision. Then type this code into decimals.py. Run the code and type its output below.

```
temperature = 98.6
print(temperature)
print (f'_____')
```

Output:

98.6

98.600000

- What is going on?

  Well, representing floating point numbers in a binary format (as required by a computer) can be difficult. So some floating point values are represented only approximately when they're converted to binary. This is fine for many applications. For example, recording someone's temperature as 98.6 versus 98.59994 is not a big

deal. However, you can imagine in some settings like banking, these fractional differences are gravely important.

To help you achieve better precision when dealing with floating point numbers, the Python Standard Library includes a module (called the decimal module) that provides a type called Decimal, which uses a special coding scheme to achieve higher levels of precision, so you need to have this line at the top of your Python scripts.

```
from decimal import Decimal
```

- To create a decimal type of 98.6 type the following. What is printed out? 98.600000000000000000
(Remember that you need the statement above in your code!)

```
#Below, we pass a string of 98.6 to the constructor of a Decimal type
temperature = Decimal('98.6')

print (f'{temperature:.20f}')
```

- We can perform arithmetic between Decimal objects as usual. Type the following code into your script. Run your script and then print what is written out below.

x = Decimal ('10.5')
y = Decimal ('2')
print(x + y) 12.5
print(x // y) 5.25
*x *= y*
print(x) 21.0
x %= 6
print(x) 3.0
print(type(x)) <class 'decimal.Decimal'>

- NOTE: The last line above reminds us that x is <u>not</u> simply a floating point number of a special type, but rather, that x is a <u>Decimal type</u> number.

- You <u>can</u> perform calculations between integers and Decimal values. Add the code below to your script and type what is printed out.

*print(x**2)* 9.00

- However, you cannot multiply Decimal values by floating point numbers. Why not? Well, as we discussed, regular floating point numbers are imprecise and a value of the Decimal type is trying to prevent this imprecision.

  Example: Add the code below to your script and type what error you receive.

  *print(x**2.0)* Type Error _____

- At the bottom of decimals.py, create a program that does the following.

A person invests a certain amount in a savings account where
- p is the original amount invested (i.e., the principal),
- r is the annual interest rate, and
- n is the number of years the money has been invested.

Assuming that the person leaves all of the earned interest in the account, then the following formula calculates the amount at the end of the nth year (denoted by "a").

$$a = p(1 + r)^n$$

Ask the user for the principal, annual interest rate (as a whole number), and the number of years the money has been invested. Then calculate and display the amount of money in the account at the end of each year. For example, if the user invests $1000 at 5% interest for 10 years then the table below should display.

```
 1   1050.00
 2   1102.50
 3   1157.62
 4   1215.51
 5   1276.28
 6   1340.10
 7   1407.10
 8   1477.46
 9   1551.33
10   1628.89
```

Use field width and aligns to make sure the output is nicely aligned.

SUBMISSION INFO

TO GET CREDIT FOR THIS LAB, UPLOAD THESE 2 DOCUMENTS TO THE SUBMISSION AREA.

- LabCh3F.pdf (should have your guess work)
- decimals.py (make sure you have the usual header at the top: file name, description, your name)