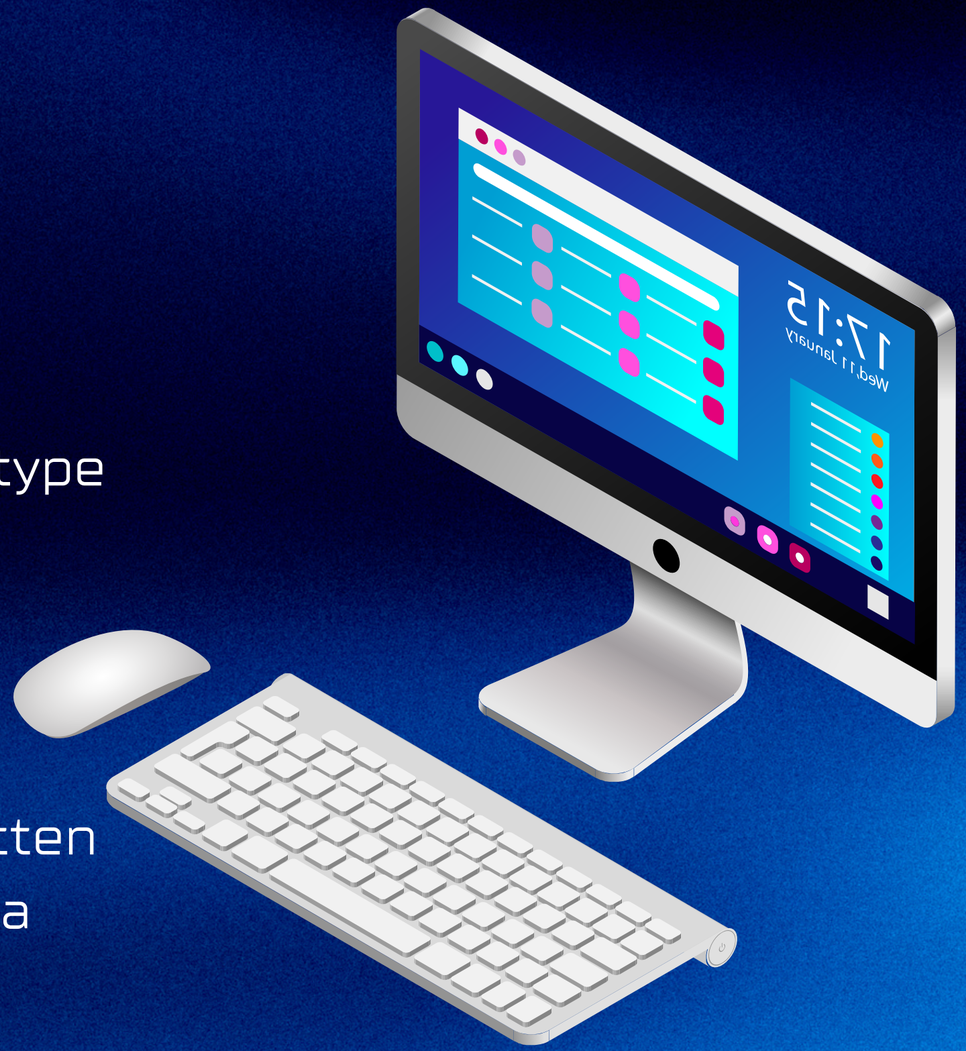
Introduction to

PYTHON PROGRAMING

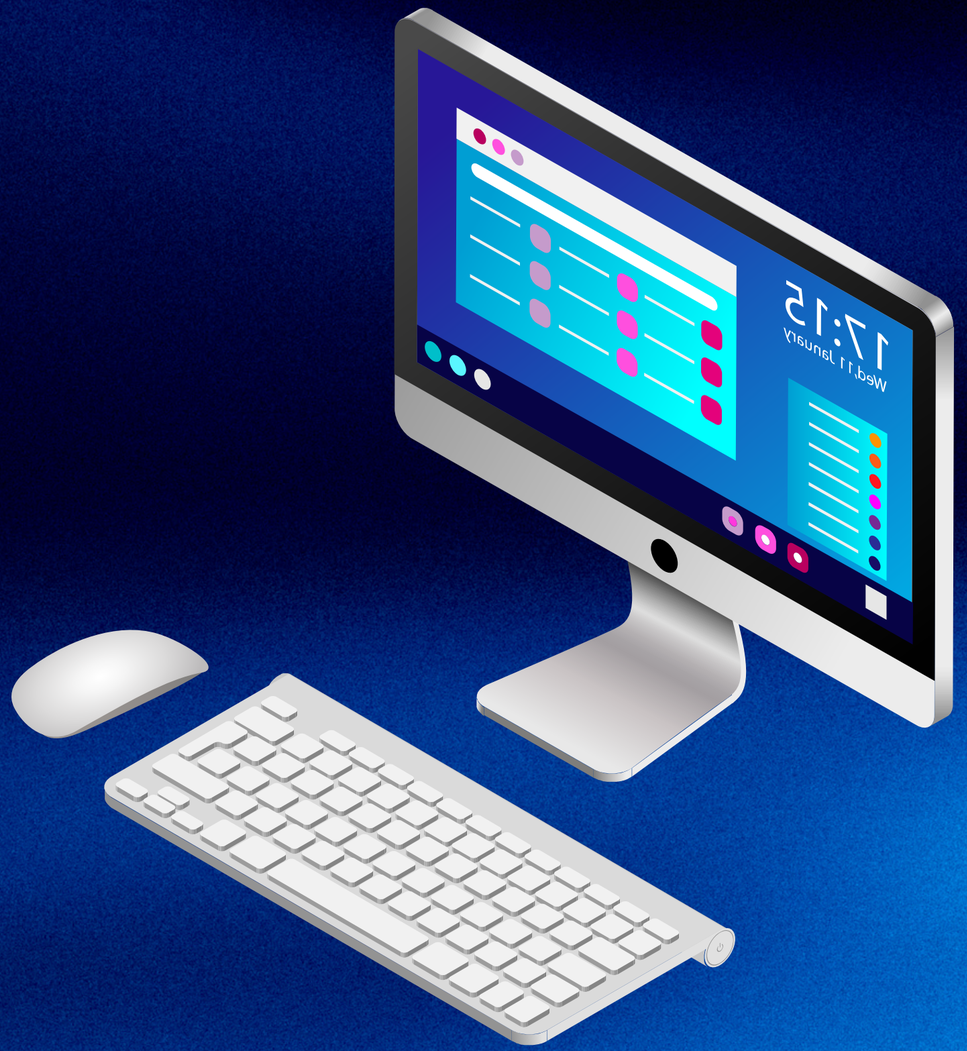
NUMBERS & MATH

1. Python has three built-in number data types: integers, floating-point numbers, and complex numbers.
2. Use the type() function to determine the data type
3. An integer is a whole number with no decimal places.
4. Create an integer by simply typing the number explicitly or using the int() function.
5. An integer literal is an integer value that is written explicitly in your code, just like a string literal is a string that is written explicitly in your code.



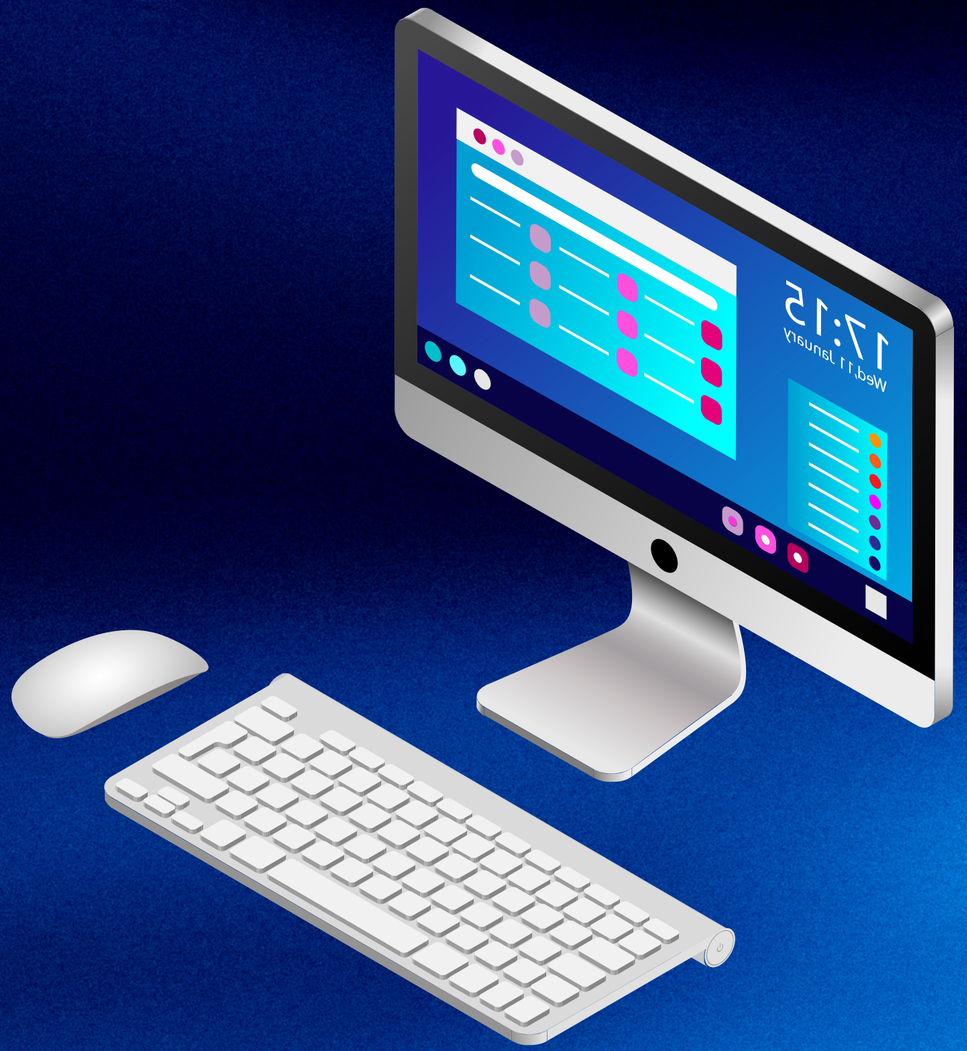
NUMBERS & MATH

1. A floating-point number, or float, is a number with a decimal place. It can be created by typing a number directly into your code, or by using the float() function.
2. A floating-point literal is a floating-point value that is written explicitly in your code.
3. For really large numbers, you can use: E-notation. This is short for exponential notation, and is the more common name for how many calculators and programming languages display large numbers.



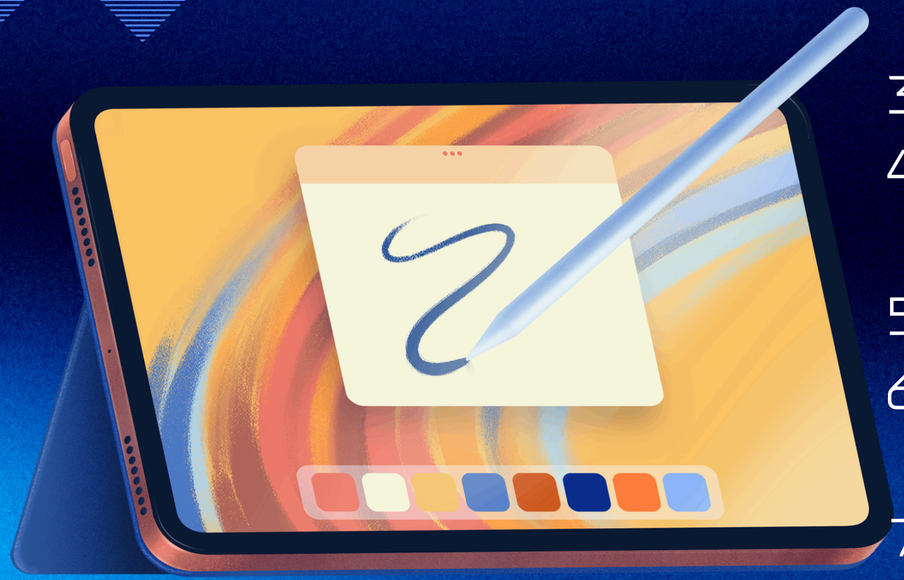
NUMBERS & MATH

1. When you reach the maximum floating-point number, Python returns a special float value inf, or -inf, stands for infinity, and it just means that the number you've tried to create is beyond the maximum floating-point value allowed on your computer.



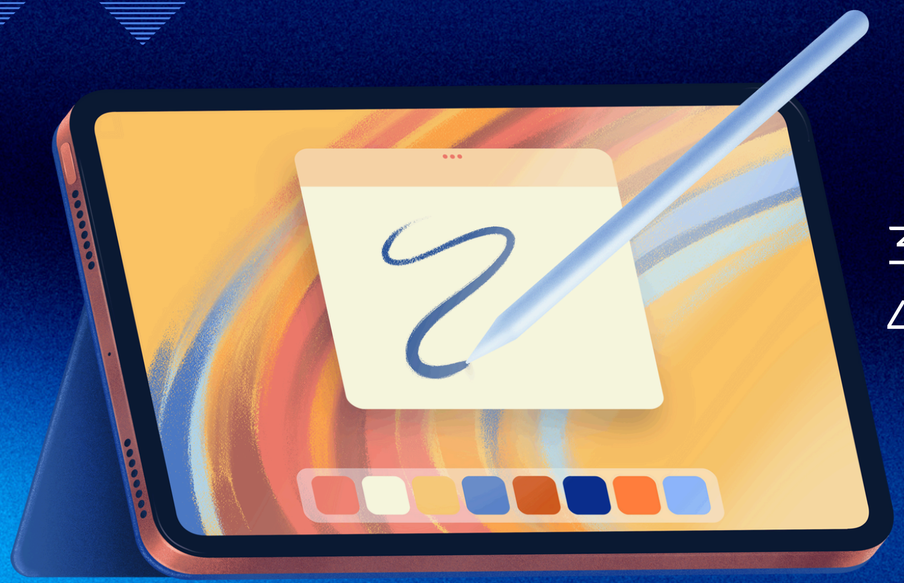
REVIEW QUESTIONS:

1. Write a script that creates the two variables, num1 and num2. Both num1 and num2 should be assigned the integer literal 25,000,000, one written with underscored and one without. Print num1 and num2 on two separate lines.
2. Write a script that assigns the floating-point literal 175000.0 to the variable num using exponential notation, and then prints num.
3. Try and find the smallest exponent N so that 2e<N>, where <N> is replaced with your number, returns inf.



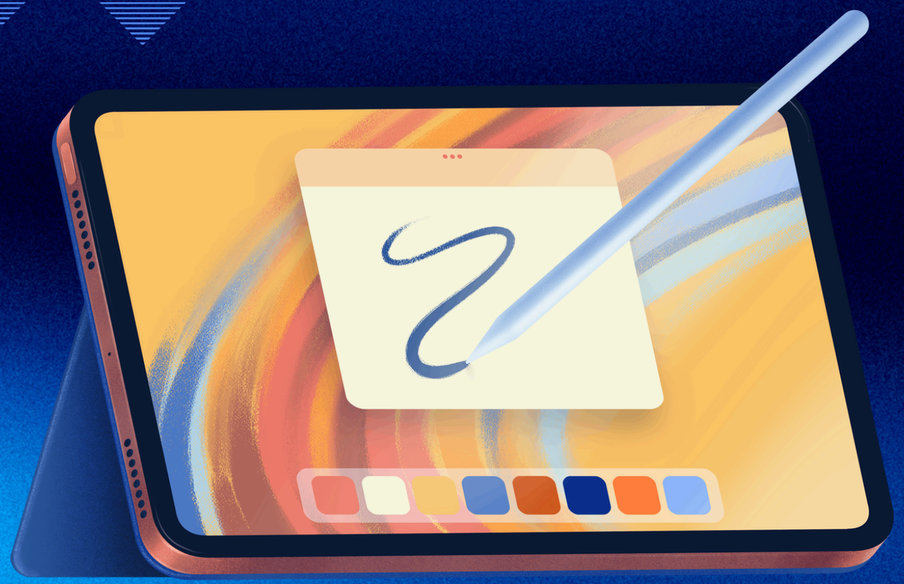
ARITHMETIC OPERATORS AND EXPRESSIONS

1. Addition is performed with the + operator.
2. The two numbers on either side of the + operator are called operands.
3. You can add an integer to a float.
4. When one type of quotes is used as the delimiter, the other type of quote can be used inside of the string.
5. Float added to an integer is a float.
6. PEP 8 recommends separating both operands from an operator with a space, 1 + 1, and not 1+1.
7. Substraction is performed with the - operator.
8. Substracting integers always results in an integer.
9. Whenever one of the operands is a float, the result is also a float.
10. You can subtract a negative number from another number.
11. PEP 8 recommends putting the second negative number in a parenthesis 1 - (-1).



ARITHMETIC OPERATORS AND EXPRESSIONS

1. To multiply two numbers, use the * operator
2. The type of number you get from multiplication follows the same rules as addition and subtraction. Multiplying two integers results in an int and multiplying a number with a float results in a float.
3. The / operator is used to divide two numbers.
4. Unlike addition, subtraction, and multiplication, division with the / operator always returns a float. If you want to make sure that you get an integer after dividing two numbers, you can use int() to convert the result.
5. Keep in mind that int() discards any fractional part of the number.
6. Python provides a second division operator, //, called the integer division operator.
7. // returns a floating-point number if one of the operands is a float.

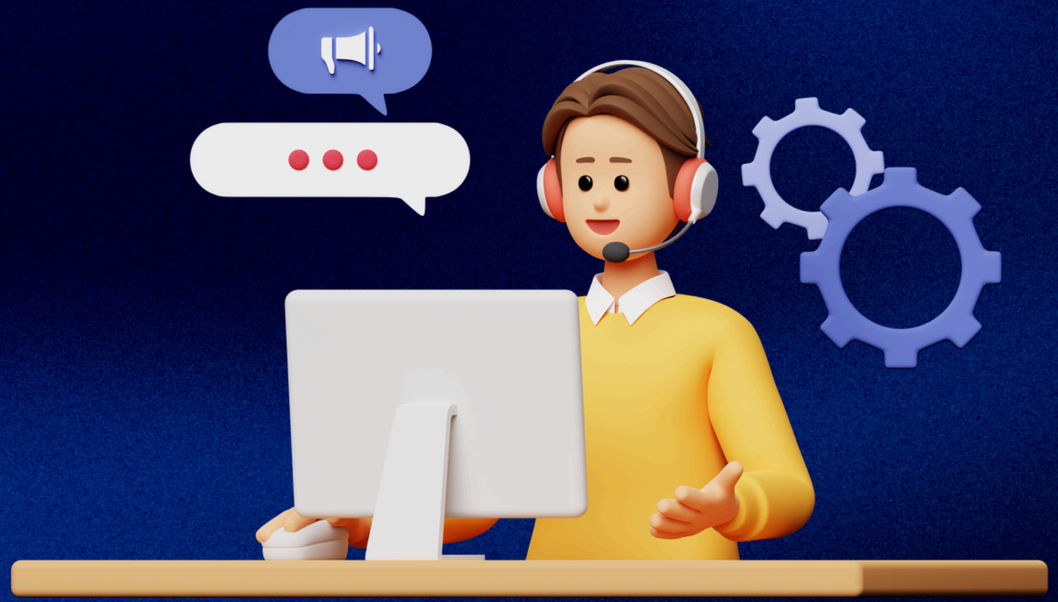


ARITHMETIC OPERATORS AND EXPRESSIONS

1. Raise a number to a power using the ** operator.
2. Exponents can also be floats.
3. For positive operands, the ** operator returns an integer if both operands are integers, and a float if any one of the operands is a floating-point number.
4. Exponents can also be negative.
5. The modulus % operator, returns the remainder of dividing the left operand by the right operand.
6. To calculate the remainder r of dividing a number x by a number y, Python uses the equation:
   r = x - (y * (x // y)).
7. Complex expressions: *, /, //, and % operators all have equal precedence, in an expression, and each of these has a higher precedence than the + and - operators.

BEST PRACTICES

- “If operators with different priorities are used, consider adding whitespace around the operators with the lowest priority(ies). Use your own judgment; however, never use more than one space, and always have the same amount of whitespace on both sides of a binary operator.



REVIEW QUESTIONS:

- Write a script called exponent.py that receives two numbers from the user and displays the first number raised to the power of the second number.

```
Enter a base: 1.2
Enter an exponent: 3
1.2 to the power of 3 = 1.7279999999999998
```

FLOATING-POINT REPRESENTATION ERROR

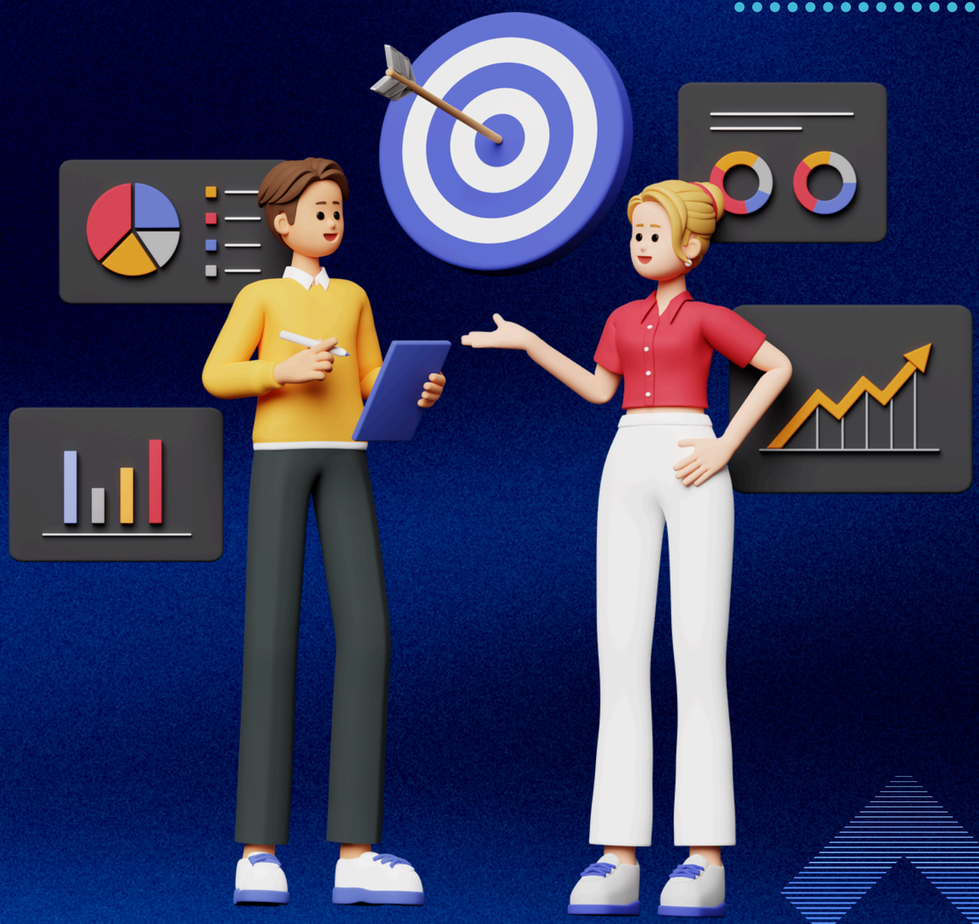
It's related to a way floating point numbers are stored in a computer's memory

Decimal representations are base 10 but computers store float in base 2.

```
: 0.1+0.2

: 0.30000000000000004
```

MATH FUNCTIONS

- round(), for rounding numbers to some number of decimal places
- abs(), for getting the absolute value of a number
- pow(), for raising a number to some power
- Rounding Numbers:
  - A tie is any number whose last digit is a five. 2.5 and 3.1415 are ties, but 1.37 is not.
  - When you round ties to even, you first look at the digit one decimal place to the left of the last digit in the tie. If that digit is even, you round down. If the digit is odd, you round up. That's why 2.5 rounds down to 2 and 3.5 round up to 4.
- is_integer() method that returns True if the number is integral, meaning it has no fractional part and returns False otherwise.



REVIEW QUESTIONS:

- Write a script that asks the user to input a number and then displays that number rounded to two decimal places.

```
Enter a number: 5.432
5.432 rounded to 2 decimal places is 5.43
```

- Write a script that asks the user to input a number and then displays the absolute value of that number.

```
Enter a number: -10
The absolute value of -10 is 10.0
```

REVIEW QUESTIONS:

- Write a script that asks the user to input two numbers by using the input() function twice, then display whether or not the difference between those two number is an integer.

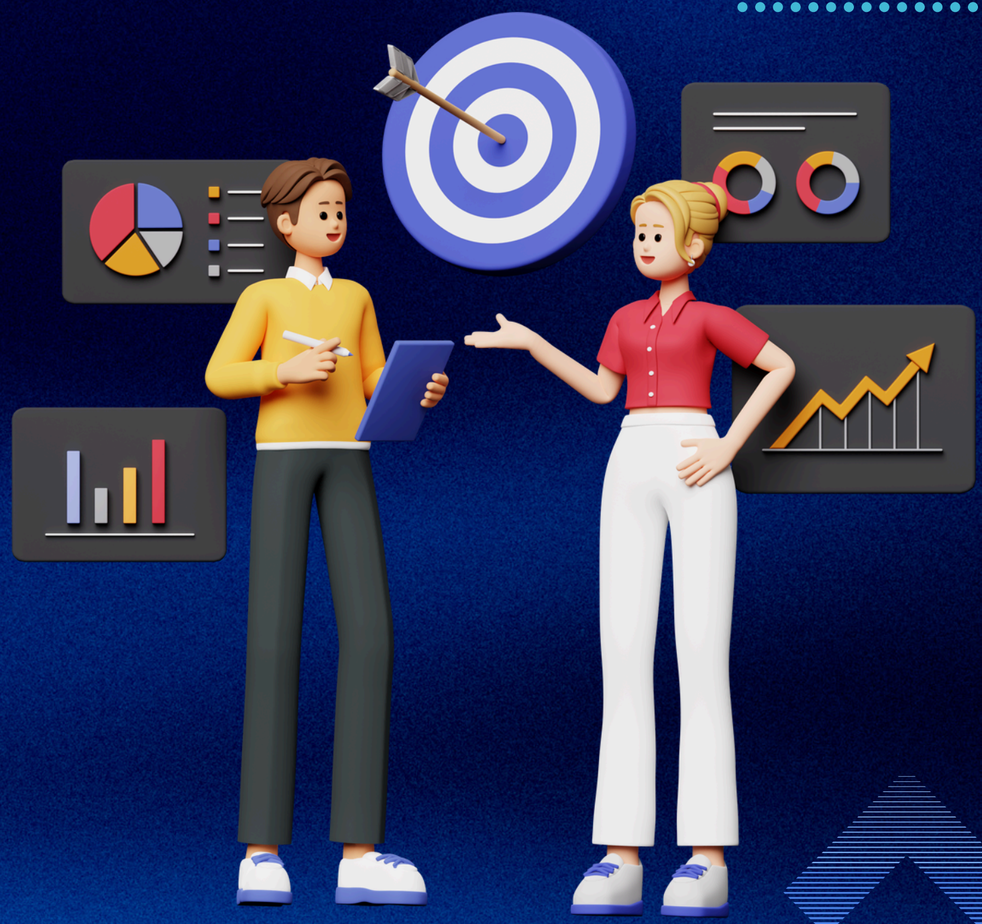
```
Enter a number: 1.5
Enter another number: .5
The difference between 1.5 and .5 is an integer? True!
```

```
Enter a number: 1.5
Enter another number: 1.0
The difference between 1.5 and 1.0 is an integer? False!
```

PRINT NUMBERS IN STYLE

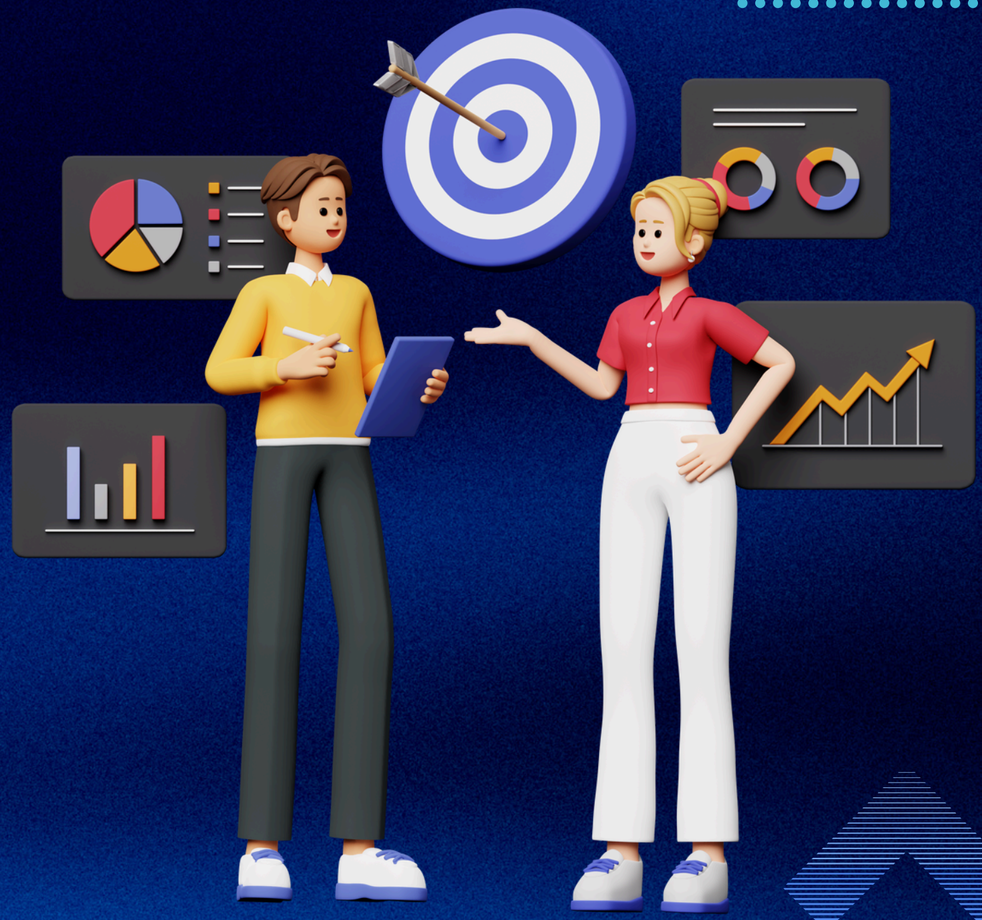

```
>>> n = 7.126
>>> f"The value of n is {n:.2f}"
'The value of n is 7.13'
```

```
>>> n = 7.126
>>> f"The value of n is {n:.1f}"
'The value of n is 7.1'
```



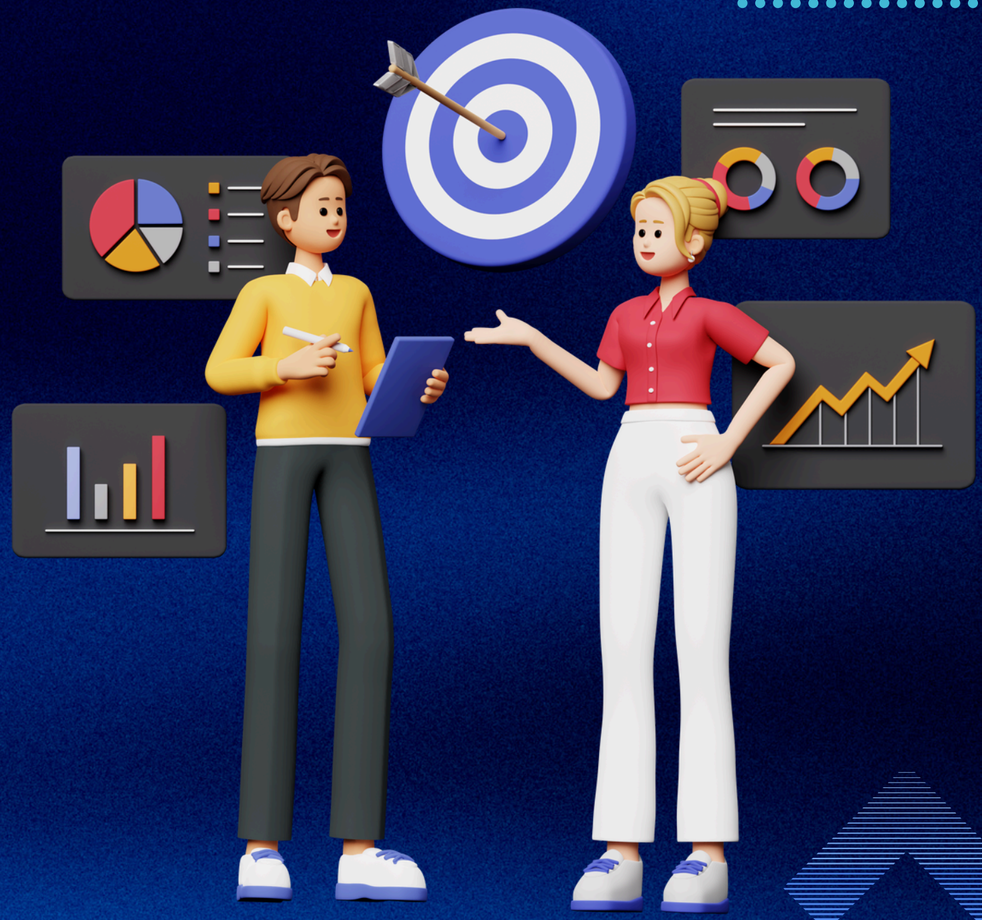
PRINT NUMBERS IN STYLE


```
>>> balance = 2000.0
>>> spent = 256.35
>>> remaining = balance - spent

>>> f"After spending ${spent:.2f}, I was left with ${remaining:,.2f}"
'After spending $256.35, I was left with $1,743.65'
```



PRINT NUMBERS IN STYLE


```
>>> ratio = 0.9
>>> f"Over {ratio:.1%} of Pythonistas say 'Real Python rocks!'"
"Over 90.0% of Pythonistas say 'Real Python rocks!'"

>>> # Display percentage with 2 decimal places
>>> f"Over {ratio:.2%} of Pythonistas say 'Real Python rocks!'"
"Over 90.00% of Pythonistas say 'Real Python rocks!'"
```



REVIEW QUESTIONS:

1. Print the result of the calculation 3 ** .125 as a fixed-point number with three decimal places.
2. Print the number 150000 as currency, with the thousands grouped with commas. Currency should be displayed with two decimal places.
3. Print the result of 2 / 10 as a percentage with no decimal places. The output should look like 20%.

THANK YOU