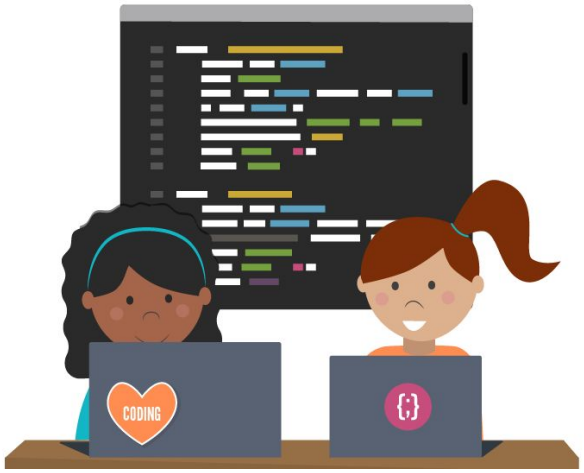
Programming in Python

Input

Getting User Input

- So far your programs haven't had any user input.
- That makes for a boring program because the output will always be the same every time you run it.
- We want to have the user enter some values and then have the program operate on those values.



Getting User Input

- To get input from the user, use the input() function.
  - When we call input(), we put a message to the user as a string.

```
var = input("This is a message to the user")
```

- The function will give us a **string** with their response from the console window.
  - We need to have a variable waiting to capture the information.

```
name = input("Enter your name: ")
print("Hi, " + name)
```

```
Enter your name: Rashi
Hi, Rashi
```

Text in green is user input

Examples

```
weather = input("Enter the weather: ")
day = input("Enter the day: ")
print(day, "is", weather)
```

```
Enter the weather: sunny
Enter the day: Friday
Friday is sunny
```

- The input() function always returns a string (str).

Reading in Numbers

- What will happen?

```
num1 = input("Enter a number: ")
num2 = input("Enter another number: ")
print(num1 + num2)
```

```
Enter a number: 8
Enter another number: 4
84
```

Reading in Numbers

- What happened?
  - The **`input()`** function always returns a string.
  - The **+** combines (or concatenates) two strings together.
- Solution:
  - When you want numbers, convert them to an **`int`** or **`float`**.

Function	Description	Example	Returns
`float(x)`	Returns a floating-point value by converting x	`float("10.0")`	`10.0`
`int(x)`	Returns an integer value by converting x	`int("10")`	`10`
`str(x)`	Returns a string value by converting x	`str(10)`	`"10"`

Reading in Numbers

- Updated code:

```
num1 = int(input("Enter a number: "))
num2 = int(input("Enter another number: "))
print(num1 + num2)
```

```
Enter a number: 8
Enter another number: 4
12
```

Example

- Write a program that asks the user to enter their name and age.
- Determine the year they were born by subtracting the age from the current year.

- Here is an example with user input.

```
What is your name? Quinn
How old are you? 17
Quinn was born in 2006.
```

Code

```
name = input("What is your name? ")
age = int(input("How old are you? "))
year = 2023
born = year - age
print(name + " was born in " + str(born) + ".")
```

```
What is your name? Quinn
How old are you? 17
Quinn was born in 2006.
```

Code with Escape Characters

```
name = input("What is your name? ")
age = int(input("How old are you? "))
year = 2023
born = year - age
print("\"" + name + "\" was born in \"" + str(born) +
"\".")
```

```
What is your name? Quinn
How old are you? 17
"Quinn" was born in "2006".
```

Programming in Python

Expressions and Operators

Variables

- Syntax (pseudocode)

```
variable = value
```

```
variable = expression
```

- Code

```
age = 20
```

```
total = 5 + 3
```

- The **=** (equals sign) means assignment.
  - Take the number **20** and store it in a variable named **age**.
- The variable has to be on the left side of the **=**.
  - The computer will evaluate the expression on the right side.
  - Then the variable will be able access that information.

Primitive Variable Types

- Python has various types of variables
  - Primitive types (we will learn today)
  - Non-primitive types (we will learn in the future)
- The primitive data types in Python are:

Data Type	Description	Examples
`str`	a string of characters	`greet = "Hello World!"`

`int`	an integer or whole number that can be positive or negative	`num = 42`
		`zero = 0`
		`negative = -1`

`float`	a floating point or decimal number	`pi = 3.1415` `neg = -12.5`
`bool`	a boolean has a value of True or False	`isLearning = True` `isSnowing = False`

Example

- Let's do an example of using a variable.
- Write a program that solves the equation:
  x = 3 * 4 + 7

```
x = 3 * 4 + 7
print(x)
```

```
result = 3 * 4 + 7
print(result)
```

- If you give the print() function a variable, it will print the value of the variable!

```
19
```

Arithmetic Operators

- To store numbers, Python has two types of variables: **`int`** and **`float`**.
- We can do mathematical operations with each.
- Arithmetic operators that are the similar to a calculator:

+	−	×	÷

+	-	*	/

- Modulo (remainder from division):

%

- Exponent:

**

2 Types of Division

- In Python, there are two types of division: True and Integer.
- True division:
  - Is what we usually think of as division
  - Will always return a float


```
4 / 3
10 / 2
10 / 2.5
99 / 100
```

```
1.3333333333333333
5.0
4.0
0.99
```

2 Types of Division

- Integer division:
  - › Only results in the integer part
  - › Truncates (or removes) the decimal part of the result

```
//
```

```
4 // 3
10 // 2
10 // 2.5
99 // 100
```

```
1
5
4.0
0
```

- Note: no rounding!

Modulo Operator

- The modulo operator gives you the remainder from division.

%

```
4 % 3
10 % 2
10 % 2.5
99 % 100
9 % 2
```

```
1
0
0.0
99
1
```

- Uses:
  - Determining if an integer is odd or even
  - Determining if one integer is evenly divisible by another integer

Math with Integers

Operator	Description	Example	Evaluates To
`+`	Addition	`7 + 3`	`10`
`-`	Subtraction	`7 - 3`	`4`
`*`	Multiplication	`7 * 3`	`21`
`/`	Division (True)	`7 / 3`	`2.333333`
`//`	Division (Integer)	`7 // 3`	`2`
`%`	Modulus	`7 % 3`	`1`
`**`	Exponent	`7 ** 3`	`343`

Math with Floats

Operator	Description	Example	Evaluates To
`+`	Addition	`7.0 + 3.0`	`10.0`
`-`	Subtraction	`7.0 - 3.0`	`4.0`
`*`	Multiplication	`7.0 * 3.0`	`21.0`
`/`	Division (True)	`7.0 / 3.0`	`2.333333`
`//`	Division (Integer)	`7.0 // 3.0`	`2.0`
`%`	Modulus	`7.0 % 3.0`	`1.0`
`**`	Exponent	`7.0 ** 3.0`	`343.0`

Question

- Will the two equations give us the same result?

  x = 3 * 4 + 7

  y = 7 + 3 * 4

```
x = 3 * 4 + 7
print(x)
```

```
y = 7 + 3 * 4
print(y)
```

- Yes! Why?

Order of Operations

- PEMDAS
  - Step 1: parenthesis
  - Step 2: exponent
  - Step 3: multiplication, division, and modulo
  - Step 4: addition and subtraction
- Prioritize with parenthesis:

```
(cost + tax) * discount
cost + (tax * discount)
```

- Without parentheses, expression are evaluated according to the order of operations.