RECORDING IN-PROGRESS

Week 1: Day 2

Tuesday 7/13/21

RECORDING IN-PROGRESS

Today’s agenda

1. Daily check-in
2. Recap from yesterday
3. Learning about conditionals
    a. Booleans
    b. If-statements
    c. Program control flow
4. Examples
5. Live coding

Daily check-in

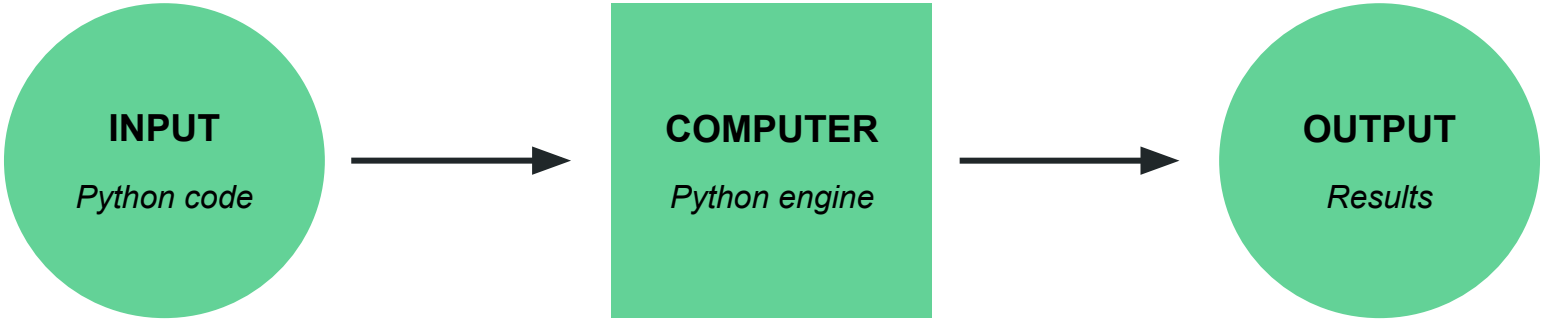
Check in

- Canvas announcements
  - Recordings
  - Calendar
  - (Shared links in Sococo)
- Lab 1
  - Questions first?
  - Go over solution (discuss due date for PM/AM)
  - Discuss how it went yesterday (breakout rooms, time to finish)
  - New Sococo rooms for working on labs
  - Submissions
- Future solutions - volunteers?
- Questions?

RECAP

Coding basics

- Programs we write get compiled and executed by the computer




```
>>> print(“Hello World!”)
```

```
>>> print(5+2)
```

```
>>> print("Hello World!")
Hello World!
```

```
>>> print(5+2)
7
```

Built-in Python functions we learned yesterday

```
print(...)
```

print(...) will automatically print whatever “...” is to the screen.
You can print a variable that you’ve already declared => print(x)
You can print an arbitrary value, like an int, float, or string => print(6.23), print(“hello!”)

```
input()
```

input() will wait for the user’s input on the Console or shell
You must enter some input for the computer program to continue.
You can also assign the user’s input to a variable => x = input()

Python types

- **Integer** (int, e.g. 5), **Floating-point** (float, e.g. 5.0), and **String** (str, e.g. “5.0”)
- We saw how to do math and arithmetic with ints and floats
  - Ints: (5 + 2), (100 / 4), (2 ^3)
  - Floats: (3.14 * 2.1), (2.4 / 0.2), (5.5 + 8.9)
- We learned how to manipulate strings to print text to the screen
  - print(“Welcome to Coding “ + “101!”)

Name	Data Type	Description
Integer	int	a number that can be written without fractions: 22 10 0 -300
Floating-point	float	a number that has a decimal component: 3.14 2.73 10.0
String	str	sequence of characters: "hello world" "hey88340" '2018'

Booleans

Python types: **booleans**

- Today we'll learn about Booleans, another Python type '**bool**'
- Booleans are logical values that indicate **True** or **False**

Name	Data Type	Description
Integer	int	a number that can be written without fractions: 22 10 0 -300
Boolean	bool	logical value that indicates **True** or **False**
Floating-point	float	a number that has a decimal component: 3.14 2.73 10.0
String	str	sequence of characters: "hello world" "hey88340" '2018'

Booleans

- Recall that integers can be any whole number: -15513, 0, 1, 2, 3, ...
- Strings can be any text between quotes: “hello”, “hi123”, “blah blah blah”
- Floats can be any number with decimals: 3.14, 2.113, -0.2221

Name	Data Type	Description
Integer	int	a number that can be written without fractions: 22 10 0 -300
Boolean	bool	logical value that indicates **True** or **False**
Floating-point	float	a number that has a decimal component: 3.14 2.73 10.0
String	str	sequence of characters: "hello world" "hey88340" '2018'

- Booleans, on the other hand, can only be values that indicate **True** or **False**

Booleans

- Create two boolean values:

```
# Assign the value True if today is Tuesday
>>> todayIsTuesday = True
>>> today_is_tuesday = True

# Assign the value True if today is Monday
>>> todayIsMonday = False
```

- Print their values:

```
>>> print(todayIsTuesday)
True

>>> print(todayIsMonday)
False
```

Booleans

- Create two boolean values:

```
# Assign the value True if today is Tuesday
>>> todayIsTuesday = True

# Assign the value True if today is Monday
>>> todayIsMonday = False
```

- Print their types:

```
>>> type(todayIsTuesday)
<class ‘bool’>

>>> type(todayIsMonday)
<class ‘bool’>
```

```
# Check if todayIsTuesday is True
if (todayIsTuesday == True):
    # do something

# Another valid option
if (todayIsTuesday):
    # do something

# Both do the same thing!
```

Booleans

- Now that we have variables that can be **True** or **False**, what can we do with them?

Booleans

- Now that we have variables that can be **True** or **False**, what can we do with them?

Surprise, there are many things we can do! Today, we'll cover conditionals and comparison operators.

Conditionals

Depending on whether a variable is **True** or **False**, we can tell our Python program to execute certain things, versus other things, in the code.

We do this using *if-statements*.

```
if (True):
    print(“It’s true!”)

if (False):
    print(“It’s false!”)
```

Conditionals

The format for *if-statements* is **always the same:**

```
if (condition):
    statement
    optional statement
    optional statement
```

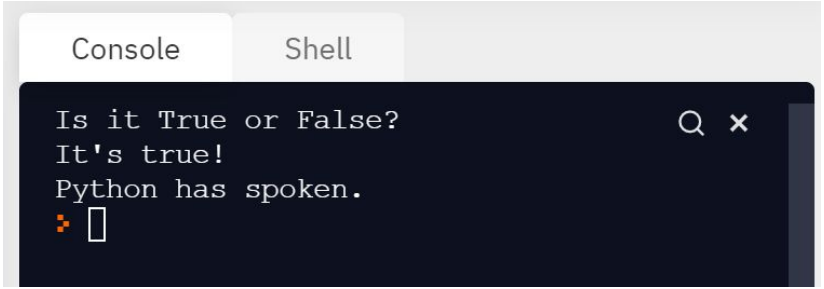
- There is always a '**:**' at the end of an if-statement
- The **condition** is any value that evaluates to **True** or **False**
- The **statements** are Python code that will be executed, *if* the conditional in the if-statement evaluates to True
- Statements must ALWAYS be indented

Conditionals

When we write *if-statements* in our code, Python will know “behind the scenes” to ONLY execute the code in the *if-statement* block if the value inside of the parentheses **evaluates to True**.

Notice how line 17 does not get printed on the right:

```
11   print("Is it True or False?")
12
13   if (True):
14       print("It's true!")
15
16   if (False):
17       print("It's false!")
18
19   print("Python has spoken.")
```



Conditionals

Here is the official Python documentation for *if*-statements:

4.1. `if` Statements

Perhaps the most well-known statement type is the `if` statement. For example:

```
>>> x = int(input("Please enter an integer: "))
Please enter an integer: 42
>>> if x < 0:
...     x = 0
...     print('Negative changed to zero')
... elif x == 0:
...     print('Zero')
... elif x == 1:
...     print('Single')
... else:
...     print('More')
...
More
```

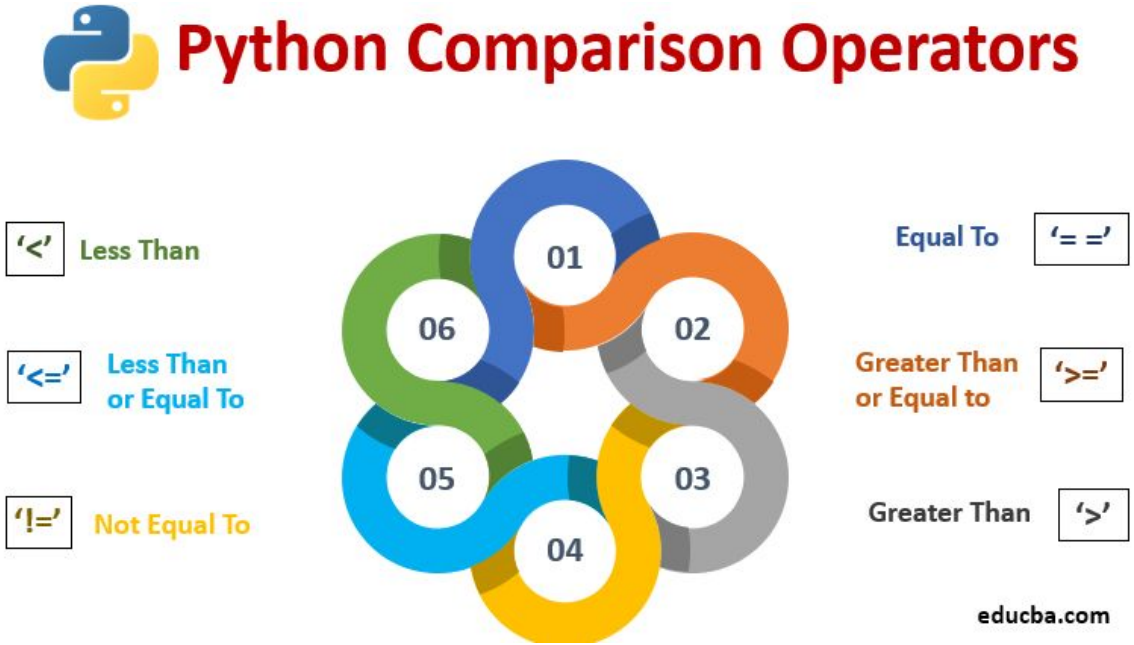
There can be zero or more `elif` parts, and the `else` part is optional. The keyword '`elif`' is short for 'else if', and is useful to avoid excessive indentation. An `if` ... `elif` ... `elif` ... sequence is a substitute for the `switch` or `case` statements found in other languages.

Conditionals

- The content of the **conditional** (aka, what's inside of the parentheses in an if-statement) *does not need to be explicitly* **True** or **False**
- Notice I've been saying "as long as the condition *evaluates* to True or False"
- One way we evaluate a conditional is by evaluating comparison operators

Comparison operators

There are 6 different Python comparison operators:



Comparison operators

Each comparison operator has a specific description that explains how the operator is evaluated in Python:

Operator	Description
==	If the values of two operands are equal, then the condition becomes true.
!=	If values of two operands are not equal, then condition becomes true.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.

Comparison operators

Operators	Meaning	Example	Result
<	Less than	5<2	False
>	Greater than	5>2	True
<=	Less than or equal to	5<=2	False
>=	Greater than or equal to	5>=2	True
==	Equal to	5==2	False
!=	Not equal to	5!=2	True

- **X < Y** is True when X is less than Y
- **X > Y** is True when X is greater than Y
- **X <= Y** is True when X is less than or equal to Y
- **X >= Y** is True when X is greater than or equal to Y
- **X == Y** is True when X is exactly the same as Y
- **X != Y** is True when X is not exactly the same as Y

What are the answers to the following?

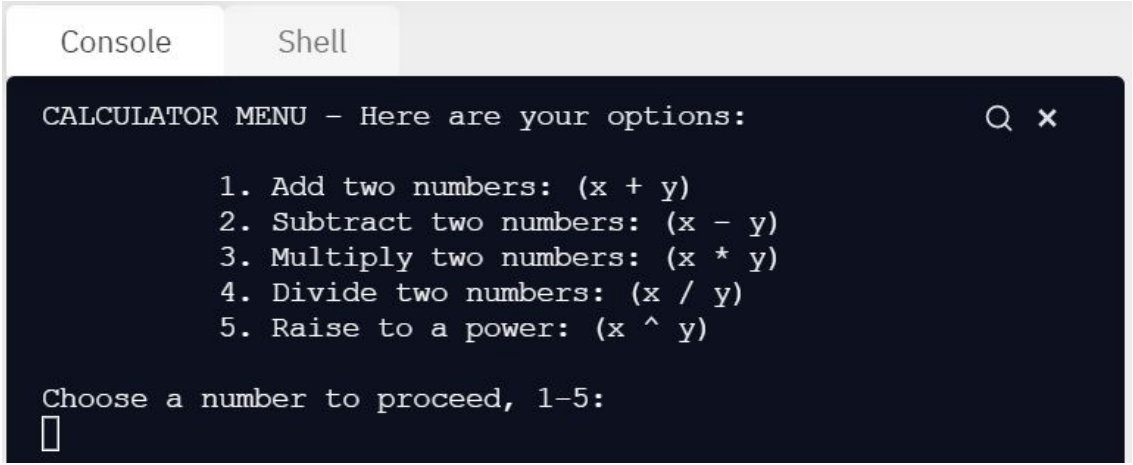
```
>>> 5 > 4
True or False?

>>> 5 == 5
True or False?

>>> 5 == “5”
True or False?

>>> 5 != 5
True or False?

>>> 5 >= 5
True or False?

>>> (3-1) * 2 > 3
True or False?
```

Answers

```
>>> 5 > 4
True
>>> 5 == 5
True
>>> 5 == "5"
False
>>> 5 != 5
False
>>> 5 >= 5
True
>>> (3-1) * 2 > 3
True
```

Making decisions in programs

- Sometimes we'll want to execute certain parts of the code we write *only when a specific condition is met*
- For example, the calculator project from yesterday's lab
- Recall our program had 5 menu choices:



Making decisions in programs

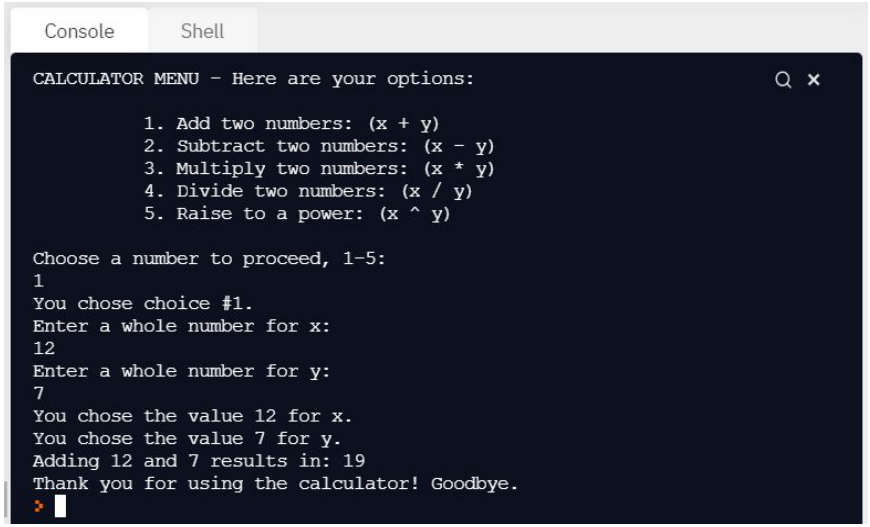
- We record what the user's choice is using input() in line 64:

```
62    print("Choose a number to proceed, 1-5:")
63    # Assign user's menu choice to a variable
64    userChoice = input()
```

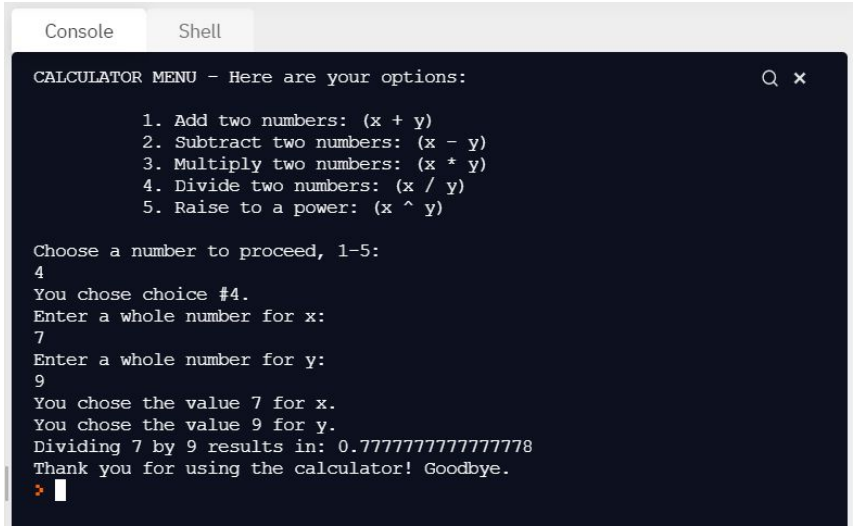
- Then, depending on the choice the user has made, we execute a specific equation from our calculator menu

Making decisions in programs

- Our program automatically performed the correct equation:



Choosing #1 - adds two numbers



Choosing #4 - divides two numbers

Making decisions in programs

- We had multiple *if-statements* in our code comparing the variable userChoice to the numbers 1-5, by using the comparison operator '=='
- Think of this as normal English:
  - If the user's choice is equal to 1, then do THIS
  - If the user's choice is equal to 4, then do THIS

```
96   ##############################
97   # CHOICE 1: Add two numbers (x + y)
98   # Fill out the code where you see TO DO's
99   ##############################
100  if (userChoice == "1"):
101      # TO DO: Assign the appropriate value f
102      # Comment out the next line and fill it
103      # z = ...
104      # TO DO: Fix this print statement by re
         values of x, y, and z. For example:
105      # "Adding 10 and 2 results in: 20"
106      print(f"Adding x and y results in: z")
```

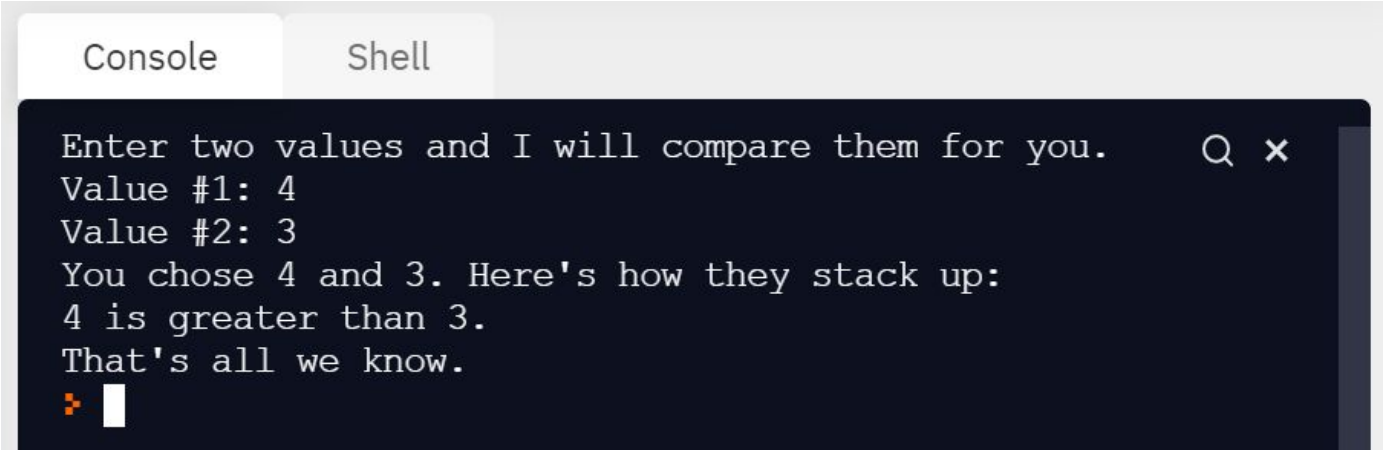
Choosing #1 add two numbers

```
132  ##############################
133  # CHOICE 4: Divide two numbers (x / y)
134  # Fill out the code.
135  ##############################
136  if (userChoice == "4"):
137      # TO DO: Assign the appropriate value fo
138      # Comment out the next line and fill it
139      # z = ...
140      # TO DO: Fix this print statement by rep
         values of x, y, and z. For example:
141      # "Dividing 10 by 2 results in: 5"
142      print(f"Dividing x by y results in: z")
```

Choosing #4 divides two numbers

Making decisions in programs

Using logic (like comparison operators, conditionals) we can control the flow of our program by ensuring only certain parts of it gets executed by Python. For example, we could ask a user to input two numbers and we could automatically compare them for the user, and print out statements depending on how they compare:



Making decisions in programs

We'll move over to replit now to keep coding together (Lecture-02).

Any questions?