### **Printing numerical values**

There are multiple ways to print out numerical values. There are a few examples below to help you out.

print(5 + 10) # default, no string values connected.

print('The value of x is: ', (5+10)) # example 1

print('The value of x is: ' + str(5.5)) # example 2

### **Variables and printing**

Variables are used to temporarily store information, and can be manipulated in multiple ways. Variables should be named so that it is easy to tell what the value held by the variable is easily identifiable.

Variables can be of different types such as:

* integers
* floats
* characters
* strings
* Boolean values
* and more!

integerOne = 15

integerTwo = 22

print(str(integerOne - integerTwo))

greeting = 'Hello'

print(greeting)

When printing out variables, there are a few options for how to print this out. Similarly to the numerical values information above, there are a few examples below.

integerOne = 15

integerTwo = 22

print(integerOne) # default, no string values connected.

print('Our first integer value is', integerOne, 'units.') # example 1 comma separated

print(f'Our second integer value is {integerTwo} units.') # example 2 using f-strings

print('Our first integer value is ' + str(integerOne) + ' units.') # example 3 using the str() function

### **Mathematical Operations**

You can use the standard mathematical operators +, -, \*, / as well as some other ones that you might not be used to using such as:

* // - This is used to do integer division (see below)
* % - This is used to find the remainder after a division (see below)
* \*\* - This is used for a power (see below)

integerDivisionOfFloats = 5.5 // 2

print(integerDivisionOfFloats) # Should be 2

moduloOfIntegers = 5 % 2

print(moduloOfIntegers) # Should be 1

powerOfIntegers = 2 \*\* 3

print(powerOfIntegers) # Should be 8

### **Converting between types**

You can convert between different data types by using functions such as:

* int() - Will convert to an integer
* float() - Will convert to a float value
* str() - Will convert to a string
* and more!

integerConversion = int(5.5)

print(integerConversion)

floatConversion = float(2)

print(floatConversion)

### **Receiving input from the user**

Sometimes you will want to ask someone to enter information for you so that you can process that information and print it back out. This is useful if you are, for example, receiving information to create a database, or to calculate the hypotenuse of a right-angle triangle, etc.

To do so, we use the input() function. The input() function takes one argument (information in the parentheses) which is typically the question that we would ask the user to help guide them, as a string. For example:

name = input('Enter your name: ')

print(name)

The above program prompts the user with Enter your name: and waits for the user to enter information to continue. Afterwards, it will then just print out the name entered. For more examples, check the basics.py file above.

# **Notes (Conditional Statements (if))**

If statements allow a branching off of the program based on whether a condition is true or false.

To check whether a condition is false, it is found within the brackets of the if statement. The condition will return a boolean value (i.e. true or false), and that will determine whether the program enters that if statement or not.

x = 1

if x == 1:

print("It's ONE!")

Note the indenting of the print() command in the code above. The indenting is required for the program to understand that the print() command is part of the if statement. Anytime you have an if statement, you will need to indent any code that you want run under those conditions. Typically you use the TAB key to indent, or use 4 spaces as a Python convention.

For conditional statements, there are different ways for the boolean value to be checked. Some examples include using:

* == to check for equality
* >= to check greater than or equal to
* > to check greater than
* <= to check less than or equal to
* < to check less than
* != to check not equal to

There will be more options shown later in the course, but these are the basics boolean checks that a conditional statement can check.

If statements can combine to check if something is true, else do something different. See a code block below.

x = 1

if x == 2:

print("It's TWO!")

else:

print("It's NOT TWO!")

This will check to see if the value of x is the same as 2 and if it isn't, the program will jump down to the else statement and run the code found within that block.

You can also chain a variety of if statements together; however, every if beyond the first is written as elif which stands for "else if". See the code block below.

x = int(input('Enter a number between 1 and 100'))

if x > 10:

print('The number is at least two digits long')

elif x < 10:

print('The number is less than 10')

else:

print('The number IS 10!')

Finally, we can chain different conditions together to check to see if multiple things are true. Chaining different conditional statements together requires one of the following:

* and which makes sure the conditions on both sides of the and are true
* or which checks if either of the conditions on both sides of the or are true

For example:

x = 1

y = 2

if x == 1 and y == 2:

print('Yay!')

else:

print('Nay!')

or

x = int(input('Enter a number: '))

y = int(input('Enter another number: '))

if x == 1 or y == 1:

print('Yay! One of the numbers was a one!')

else:

print('Nay! Neither number was the number one...')

Lastly, you can chain the chains together; however, you will want to ensure that you use parentheses to help you, as and has priority over or, just like how \* has a higher precedent in the order of operations over + for mathematics.

x = 1

y = 2

z = 3

if (x == 1 and y == 1) or (z == 1):

print('Yay!')

elif (x == 1 and y == 2) and (y == 3 or z == 3):

print('WOOHOO!')

else:

print('Nay!')

# **Notes (Looping Structures (while and for))**

You will want to think of a loop as completing either the same, or similar, task over and over until a particular condition has been reached. In this course we'll focus on two different types of looping structures: while and for.

## **While Loops**

This looping structure will continue doing something while a condition is True, like the structure below.

count = 1

while count < 10:

print(str(count))

count = count + 1

The structure of a while loop is very similar to an if statement as shown above. You can use anything that you can with an if to check whether the condition is True or False.

Another option that open up with a while loop is the else option, just like an if statement. The else is what happens the moment the while loop becomes False

For more examples, check out looping\_while.py file above.

The while loop is used for when you are not sure exactly how many times you need to run through a loop, whereas the for loop is only run when you know how many times to loop.

## **For Loops**

This looping structure will count from one value to another by a particular increment structured as below.

for iterator in range(start, stop, step):

pass

The above has some new concepts such as:

* iterator - Is a variable that is created on the spot and can be used inside the for loop. This value changes every time the loop starts over.
* range() - This creates a list of values for the iterator to be while inside the for loop.
* start - Is an integer value to start the range.
* stop - Is an integer value to end the range. Python will not include that particular value itself (see examples).
* step - Is the value that the iterator changes by each time the loop restarts.

# Count up

for count in range(0, 5, 1):

print(str(count))

# Output

# 0

# 1

# 2

# 3

# 4

# Count down

for count in range(5, 0, -1):

print(str(count))

# Output

# 5

# 4

# 3

# 2

# 1

There are advanced uses of for loops. These will be pointed out when we get to those concepts in the future.

#### **else**

Another option that open up with a while loop is the else option, just like an if statement. The else is what happens the moment the while loop becomes False

For more examples, check out looping\_for.py file above.

## **Loops in general**

A few of the commands and options that open up with a while or for loop is:

* break - To break out of a loop early
* continue - To jump back to the beginning of the loop early. This should be rarely used.
* pass - This is something to be used either (a) temporarily, or (b) to ensure your program runs properly

# Printing the addition of two integers

print(str(1 + 2))

# Printing of the subtraction of two floats

print(str(4.3 - 2.1))

# Boolean values

right = True

wrong = False

print(str(right))

print(str(wrong))

# Two integers to be used for the rest of the program.

integerOne = 1

integerTwo = 2

# Two float values to be used for the rest of the program.

floatOne = 3.14

floatTwo = 5.3

# Multiplying two float values.

productOfFloats = floatOne \* floatTwo

print(str(productOfFloats))

# Division of two float values.

quotientOfFloats = floatTwo / floatOne

print(str(quotientOfFloats))

# Using integer division on two floats

integerDivisionOfFloats = floatTwo // floatOne

print(str(integerDivisionOfFloats))

# Using the modulo operator

moduloOfIntegers = 5 % 2

print(str(moduloOfIntegers))

# Converting between floats and integers

print(str(int(floatOne)))

print(str(float(integerOne)))

# Equation

print(str(4 \* 2 / 8))

# Getting input from the user using input('Question goes here')

# Then formatting and printing back out the user input.

name = input('Enter your name: ') # Note: This asks the user to 'Enter your name: '. Do NOT change this to your name directly

age = input('Enter your age: ')

print('Hello ' + str(name) + ', you are ' + str(age) + ' years old.')

# For loop to count up

for count in range(0, 5, 1):

print(str(count))

# For loop to count down

for count in range(5, 0, -1):

print(str(count))

# Using an "else" statement with a while loop

for count in range(0, 5, 1):

print("Else: " + str(count))

else:

print("This happens after the for loop ends")

# Breaking out of a loop early

for count in range(0, 5, 1):

if count == 2:

break

# Using Continue to skip certain values

for count in range(0, 5, 1):

if count % 2 == 0: # skip EVEN numbers (and ZERO)

continue # immediately jump back to "while count < 10"

print(str(count))

# Using pass

for count in range(0, 5, 1):

if count % 2 == 0: # Plan to do something for EVEN numbers (and ZERO)

pass

elif count == 3: # Plan something else for when count is 3

pass

print(str(count))

# Using a while loop to count up

count = 1

while count < 10:

print(str(x + count))

count = count + 1 # this can also be written as count += 1

# Using a while loop to count down

count = 275

while count > 250:

count = count - 1 # this can also be written as z -= 1

if count % 2 == 0:

print(str(z) + ": This number is even")

# Creating an infinite loop. This loop won't stop.

count = 1

while count == 1:

print("Count is equal to the number 1")

# Using an "else" statement with a while loop

count = 1

while count < 10:

print(str(2 \* count))

count += 1

else:

print("Done")

# Breaking out of a loop early

count = 1

while count < 10:

if count == 5:

break

count += 1

# Using Continue to skip certain values

count = 1

while count < 10:

count += 1

if count % 2 == 0: # skip EVEN numbers (and ZERO)

continue # immediately jump back to "while count < 10"

print(str(count))

# Using pass

count = 1

while count < 10:

count += 1

if count % 2 == 0: # Plan to do something for EVEN numbers (and ZERO)

pass

elif count == 5: # Plan something else for when count is 5

pass

print(str(count))