LESSON 1: INTRO TO COMPUTER SCIENCE WITH PYTHON

Kalu Kalu

Careers With Python

Software Developer/Python Developer

A Python Web Developer is responsible for writing server-side web application logic. Python web developers usually develop back-end components, connect the application with the other (often third-party) web services, and support the front-end developers by integrating their work with the Python application.

Data Analyst

Data analysts translate numbers into plain English. Every business collects data, whether it's sales figures, market research, logistics, or transportation costs. A data analyst's job is to take that data and use it to help companies make better business decisions.

Data Scientist

A Data Scientist combines statistical and machine learning techniques with Python programming to analyze and interpret complex data. (datacamp.org)

Quantitative Analyst

In finance, quantitative analysts ensure portfolios are risk balanced, help find new trading opportunities, and evaluate asset prices using mathematical models.

Running Python Programs

Two Ways To Run Python Programs

There are two ways to run python programs. One way is by typing your python commands directly into the python interactive interpreter of your terminal. Another way is to first use a text editor to type your python into a python “script” or “program” and run that “script” (aka program). We will go through each way in turn.

Method 1: Using The Python Interactive Interpreter

As mentioned above the first way to type python code is by using the python interactive interpreter. The python interactive interpreter aka your “python shell” is a way to quickly test

out python code. You type and run commands, generally one line at a time. Please open up your Terminal and enter python3 by typing “python3” in your terminal.

Pic1

If you see a command that says python3 is not available, then simply try python, like pictured below.

Pic2

Once you are in your Python Interactive Interpreter you can try your first bit of python code. Please type the following:

```
>>> bill = 54.87
>>> tip_percent = 0.10
>>> tip = bill * tip_percent
>>> total = tip + bill
>>> print("Here is your total tip amount")
>>> print(tip)
>>> print("Here is your total bill")
>>> print(total)
```

Your output should look like this:

```
Here is your total tip amount
5.487
Here is your total bill
60.357
```

Pic3

Congratulations. You have just ran your first code using the Python Interactive Interpreter. To leave the interactive interpreter and get back to your bash shell please enter the command quit()

```
>>> quit()
```

Pic4

You should now be back in your Bash Shell.

Method 2: Saving Your Code In a Script And Running The Script

The second method of running python code is using a Text Editor, such as Sublime, to type your code in a script and then saving and running that script in your terminal.

Creating A Python Script In Sublime

To get started first open a new sublime document. Then click File > Save As

Pic5

We want to save the Python file somewhere where we will remember it when we wish to run it. And we want to make sure we end our filename with .py (pronounced dot pie) so that the computer knows it's a Python program. Why don't you name the program, calculate_bill.py . Make sure that you do not use a space when naming python programs. You can use an underscore, _ , instead of a space.

Remember, we want to pay special attention to **WHERE** we save it. For now just save it in the Desktop.

Pic6

With the file saved, we can move on to typing our code in our newly created calculate_bill.py file. Inside of your sublime calculate_bill.py please type the following:

calculate_bill.py

```
bill = 54.87
tip_percent = 0.10
tip = bill * tip_percent
total = tip + bill
print("Here is your total tip amount")
print(tip)
print("Here is your total bill")
print(total)
```

Pic7

Saving Changes In Sublime

And then save. You will know when your file is saved in Sublime, by look at the file tab. If there is a grey dot next to your filename, then you have not yet saved. If there is a grey X instead then you have saved all changes.

When it is unsaved it looks like this: Grey Dot

Pic8

When it is saved it looks like this: Grey X

Pic 9

Running Your Script In The Terminal

Now that we have our created our script it is time to run it. Please open your Terminal.

Pic10

First, we must “cd” into the directory (aka folder) where we saved our calculate_bill.py file. Since we saved it in our Desktop, which is located in our home folder, we need to cd into ~/Desktop .

```
Kalus-MacBook-Pro:~ kalukalu$ cd ~/Desktop/
```

Pic11

Once we are in the same folder as our script, we can run the file using the python keyword and the name of our file as an argument. Python calculate_bill.py

```
Kalus-MacBook-Pro:Desktop kalukalu$ python calculate_bill.py
```

We should get the following output printed on our Terminal screen.

```
Here is your total tip amount
5.487
Here is your total bill
60.357
```

See pic12

Do Not Confuse Your Bash Shell With Your Python Shell

Many students make the mistake of confusing their Python Interactive Interpreter (aka Python Shell) with the Bash Shell. Your Bash Shell is the default shell that your Terminal opens up in. Bash is a programming language. With this programming language you can direct the operation of the computer by entering commands as text for a command line interpreter to execute.

Python is a completely different programming language with different grammar and syntax. You cannot enter python code directly inside of a Bash Shell. You first must use some Bash Command to enter inside of your Python Shell. Once in your python shell you can comfortably enter Python Code directly into the shell. To quit your Python shell and return back to the default Bash shell, you simply type the command quit().

Python Shell or Python Script?

Great job on running your first python code. From here on out we will mostly type our python code in a script file, save that script file and then run that script file from our Bash shell. But, whenever we want to quickly test out code we can use the python shell instead.

Statements & Operations

Statements

Statements are a very important part of programming. Just like in Math, a statement is evaluated to it's simplest form.

From here on out we're going to run our python code by saving it in a script file and running it in our Bash shell.

First, open your Terminal and in your Desktop create a folder called python_book. Inside that folder we're going to create a file called ex1.py and use Sublime to write our code inside of it.

```
Kalus-MacBook-Pro:~ kalukalu$ cd Documents/
Kalus-MacBook-Pro:Documents kalukalu$ mkdir python_book
Kalus-MacBook-Pro:Documents kalukalu$ cd python_book/
Kalus-MacBook-Pro:python_book kalukalu$ touch ex1.py
```

Pic16

Click on the spotlight search icon on the top right corner of your screen and search and open Sublime Text.

Pic 17

Once Sublime Text is open Click on File > Open (if you are on Ubuntu click on Open Folder).

Pic18

The find and open your python_book folder.

Pic19

Lastly click and open ex1.py.

20

A Simple Expression

Inside of ex1.py please type the following code.

A note on Comments:

You do not have to type any lines of code that start with a #. Any code followed by the # sign is a comment and will be ignored by the complier at runtime. It Is merely explanatory to help explain a code segment.

ex1.py

```
# You go to the mall and you purchase
# one item for two dollars and forty cents.
#  and another for five dollars and six cents.
# What is your total bill?
```

```
total = 2.40 + 5.06
print("Here is the total bill")
print(total)
```

pic21

Don't forget to save the file. Then open your Terminal and use your python command to run your ex1.py program.

```
Kalus-MacBook-Pro:python_book kalukalu$ python ex1.py
```

Your output should look like this.

```
Here is the total bill
7.46
```

Pic22

Expression Order of Operations: PEMDAS

What happened? First the expression on the right side of the equals sign (2.40 + 5.06) is evaluated. Generally, just like in math, the order of operations in which expressions are evaluated follow the PEMDAS rule, standing for, Parenthesis, Exponents, Multiplication, Division, Addition, Subtraction. After the expression is evaluated it is then assigned to the variable to the left of the equal sign.

A More Complex Expression

Let's try a more complex express. 2 + 2 * 3 . Before we type the code, can you guess what this Math expression will evaluate to following the order of operations rules you learn in grade school? Think about it for a second and then create another file called ex2.py . Type the following code inside of your new ex2.py file.

ex2.py

```
# Pens And Pencils both cost $2.
# You buy three pens and one pencil.
# What is your total?
total = 2 + 2 * 3
print(total)
```

Pic23

After you have typed and saved it, open your terminal. In case you closed your past terminal cd back into your python_book folder so that we can run the newly created ex2.py script that is in there. Then run your script.

```
Kalus-MacBook-Pro:python_book kalukalu$ cd ~/Documents/python_book/
Kalus-MacBook-Pro:python_book kalukalu$ python ex2.py
8
```

Your response should be six, the same answer that you should have received when doing it by hand. According to order of operations, when we have the expression 2 + 2 * 3 , we first evaluate the multiplication because multiplication (M) comes before addition (A) in order of operations (PEMDAS). Therefore we first must evaluate the statement 3 * 2. The expression 3 * 2 evaluates to 5. Now we are left with the expression 2 + 6 , which equals 8. Therefore the value 8 is assigned to the variable name total.

Expressions & Variables

Expressions work the exact same way with variables. In ex3 we're going to recreate what we did in ex2 except with variable names instead. Please create a new file called ex3.py.

A Note On Exercise Names:

You will notice a pattern. Every new exercise will have a new file with a new filename that you will have to create. The filenames will be named by adding one to the last exercise number, for example, ex4.py , ex5.py ex6.py and so on.

Please make sure that you take note of which exercise we're in.

ex3.py

```
# Pens And Pencils both cost $2.
# You buy three pens and one pencil.
# What is your total?
pen = 2
pencil = 2
total = pen + pencil * 3
print(total)
```

As you can probably already guess, when we run this in our terminal we will get the same output as with ex2.py.

```
Kalus-MacBook-Pro:python_book kalukalu$ python ex2.py
8
Kalus-MacBook-Pro:python_book kalukalu$ python ex3.py
8
```

Order of operations work the same whether you are using the actual number, or a variable name as a placeholder.

Expressions & Parenthesis

Just like in math, you can use parenthesis for order of operations reasons, or just to make your code more clear.

For instance, try this exercise:

ex4.py

```
# Pens And Pencils both cost $2.
# You buy three pens and one pencil.
# What is your total?
pen = 2
pencil = 2
total = pen + pencil * 3
correct_order = pen + (pencil * 3)
incorrect_order = (pen + pencil) * 3
print(total)
print("This is the correct order of operations")
print(correct_order)
print("This is an  order of operations of the original question")
print(incorrect_order)
```

When you run the code, this should be your output:

```
Kalus-MacBook-Pro:python_book kalukalu$ python ex4.py
This is the default order of operations
8
This is the correct order of operations
8
This is an  order of operations of the original question
12
```

Operators

You have already been exposed to many operators so far in this book. For instance, the addition and multiplication operator. When we did 2 + 3 * 2 .

You have also been introduced to probably the most vital operator, the assignment operator, = . When we wrote total = 2 + 3 * 2 , the assignment operator acted to assign the result of the

expression 2 + 3 * 2 to the variable name total. As we stated earlier, the expression to the right of the assignment operator is first evaluated, and then the result is assigned to the variable name to the right of the operator.

Pic24

Math Operators

Here is a list of basic math operators:

1. Addition
2. Multiplication.
3. Subtraction
4. Division
5. Modulus
6. Power

Comparison/Equality Operators

There are your basic Math operators. But you also have other types of operators such as comparison operators. Here the list of comparison operators

1. Greater than >
2. Less than <
3. Greater than or >=
4. Less than or equal <=
5. Equal ==

Do Not Confuse the Equality (==) Operator With the Assignment Operator (=)

Before we test out the comparison operators, it is worth noting, do not get the double equals equality comparison operator, == , confused with the single equals assignment operator mentioned earlier, = . They are not they same and confusing is the cause of errors for beginner programmers and even experienced absent minded programmers (myself included).

Comparison Operators Return True or False Boolean Values

As we will discuss more below, comparison operators (for example, ==) checks to see if the value on the left of the == is equal to the value to the right of it. Equality comparisons always equate to a True or False Boolean expression. For example 2 + 3 == 6 -1 would evaluate to the expression True.

Comparison/Equality Operators

Operators Practice

The main thing to know for order of evaluations when it comes to operators is that they are pretty intuitive. Arithmetic operators are evaluated first, then the others.

Before we get started let's discuss the modulus operator, %. The % operator gives yoo the remainder when dividing by a number. For example, 3 % 2 will return 1 because 3 / 2 leaves a remainder of one. Ok, now let's get started. Before you type and run the code try to guess what the output will be. After you guess go ahead and type and run the following code.

ex5.py

```
remainder = 5 % 2
boolean1 = remainder == 1
boolean2 = remainder >= 2
boolean3 = boolean2 and boolean1
boolean4 = boolean2 or boolean1
boolean5 = True and False or True and True or False and True or False or True
num1 = 2 + 3 ** 2 - (3 + 3)
num2 = num1  + 19 % 17 * 4 - 1
num3 = num1 / 2 + num2

print(remainder)
print(boolean1)
print(boolean2)
print(boolean3)
print(boolean4)
print(num1)
print(num2)
print(num3)
```

Did you get the expected you guess? Can you create your own fun brain teasers?

Python Data Types

1) Strings

How to Create A String

Strings are a fundamental data type that we have already been working with. A string is created by merely wrapping quotes, single or double, around any text. Let's create several example strings.

ex6.py

```
string1 = 'How are you doing'

string2 = "You can create a string this way as well."
string3 = "Strings can also have numbers, like 23 414, and some symbols, like #$*&#"
intro1 = "Strings are used to display text that is meant for human eyes."
print("\n\nHere is the first intro 1")
print(intro1)
intro2 = " You can add one string to another string. "

# You can use intro1 in the assignment even though you are overriding it
intro1 = intro1 + intro2
# IT's very comment to see the same variable name on both sides
# of the equals sign like above. It's okay
# as long as that variable has already been created before

print("\n\nHere is string 1")
print(string1)
print("\n\n Here is string 2")
print(string2)
print("\n\n Here is string 3")
print(string3)
print("\n\n Here is the second intro 1")
print(intro1)
print("\n\n Here is intro2")
print(intro2)
```

Did you notice that strings had a funny \n ini them. Some characters inside of a string definition have special meaning, we call these characters escape characters. If you do \n the will tell python to add a new line when displaying this string for humans to see.

Substring Selection

How do you select one single character from a string? How do you select a substring of characters from a string? That is the topic of substring selection, also known as string slicing.

If you have a string, quote = “We have a dream.” , how do you select a piece of that string? The answer is string select with “bracket-notation”. Bracket refers to [ ] . To select a sub-string you just add [ ] to the string and write the start and stop points you want. name[0:3] would give me the first to characters 0, 1, and 2, or “I h”

Please run the following code.

ex7.py

```
quote = "We have a dream"
substring1 = quote[1]
substring2 = quote[-1]
substring3 = quote[0:3]
substring4 = quote[1:4]
substring5 = quote[:]
substring6 = quote[:7]
substring7 = quote[6:-1]
substring8 = quote[-5:]

print(substring1)
print(substring2)
print(substring3)
print(substring4)
print(substring5)
print(substring6)
print(substring7)
print(substring8)
```

Can you guess what the output should be? Try to guess first before looking. The answer is below:

```
e
m
We h
e h
We have a dream
We have
e a drea
dream
```

Position Numbers Start With 0

As you may have just noticed, when discussing position numbers in programming we usually start with zero. For instance, the first character in the string “We have a dream” is the “W”. The number we used use to describe this position is position 0, not position 1. The second character, “e” , would position 1, and so on.

When Doing String Selection With Bracket Notation The Second Argument Is Exclusive

From Position X Up Until Position Y

Earlier we had a piece of code that said substring3 = quote[0:4]. Surprising this printed our “We h” instead of “We ” . That is because the second number is EXCLUSIVE. That means we do not include the character at that position, we include everything UP UNTIL. On the other hand, the first number is always include. So you should think of [0:4] as saying give me all the characters **from** position 0 **up until** position 4.

String Quiz

ex8.py

```
quote = "Practice makes perfect"

substring1 = quote[0:8]
substring2 = quote[-7:]
substring3 = quote[9:14]
substring4 = substring2 + " " + substring1 + " " + substring3
substring5 = substring3 + " " + substring2 + " " + substring1
substring6 = substring1 + " " + substring3 + " " + substring2
substring7 = substring3 + " " + substring1 + " " + substring2

story = "I once heard that "
story += substring4 + " "
story += " the saying " + substring6 + " true. "
story += "So we must " + substring1 + " " + substring2
story += " to be " + substring2
story += "."

print(substring1)
print(substring2)
print(substring3)
print(substring4)
```

```
print(substring5)
print(substring6)
print(substring7)
print(story)
```

QUESTION 1:

What will be the output of print(substring1)?

QUESTION 2:

What will be the output of print(substring2)?

QUESTION 3:

What will be the output of print(substring3)?

QUESTION 4:

What will be the output of print(substring4)?

QUESTION 5:

What will be the output of print(substring5)?

QUESTION 6:

What will be the output of print(substring6)?

QUESTION 7:

What will be the output of print(substring7)?

QUESTION 8:

What will be the output of print(story)?
