

Learn Python

StartOnAI Tutorial 1:

This is a tutorial that developments a firm foundation for understanding the Python coding language and its relevance and applications to Machine Learning (ML). Python itself is extremely useful for ML due to its simplicity, stability, and availability of many tools at a coder's disposal. This tutorial will cover basic concepts such as programming syntax, constructions such as loops, some basic algorithms, reading from a .csv file type, and understanding the Jupyter Notebook.

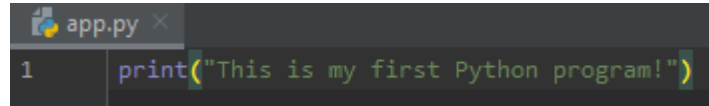
1. What is Python?

Python is a versatile, object-oriented, high-level coding language with dynamic semantics. It has many efficient and useful data structures built into its libraries that allow it to be attractive for application development. Python's extremely simple syntax and comprehensibility allow it to be one of the world's most popular coding language, and this is one of the reasons it adapts so well to ML. Machine learning. Due to Python being a non-compiling language, but rather a script language, it outputs comparatively faster than other languages, a definite benefit when it comes to complex ML tasks. Python's flexibility allows programmers to focus strictly on the inner-working of ML program, rather than focus on technicalities with Python's syntax and other potential nuances that may adversely effect progress. Python's immense collection of libraries in ML (collections of pre-written code by software engineers to solve common programming tasks) let it be extremely useful for a plethora of ML tasks such as data analysis, computer vision, natural language processing, OpenCV (Open Source Computer Vision Library) and many more.

2. Understanding Programming Syntax, Constructions, Algorithms, and Files

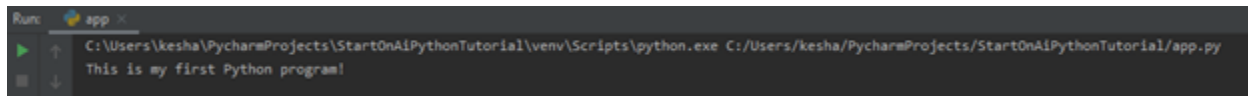
Introduction - Our journey of python begins with using a programming environment that allows us to enter our own code where the computer will read and produce an output. These programming environments known as IDE's (Integrated Development Environments) make coding much easier as it has helpful shortcuts. A preferred IDE for Python would be PyCharm (the one that I will be using for the specific tutorial). Another option for Python is using an online compiler such as Repl.it which is extremely useful if you are unable to download files.

Prints – We can now move on to creating basic print statements. Create a python file and name it app.py (any name will do as long as you label it as a .py file). Now in your compiler, enter the following code.



```
app.py x
1 print('This is my first Python program!')
```

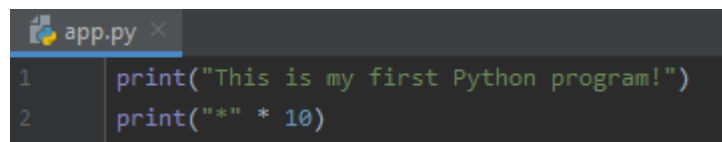
Now at the top of PyCharm you should see a “run” option. Click run and at the bottom of the screen you should see your first Python output!



```
Run: app x
C:\Users\kesha\PycharmProjects\StartOnAiPythonTutorial\venv\Scripts\python.exe C:/Users/kesha/PycharmProjects/StartOnAiPythonTutorial/app.py
This is my first Python program!
```

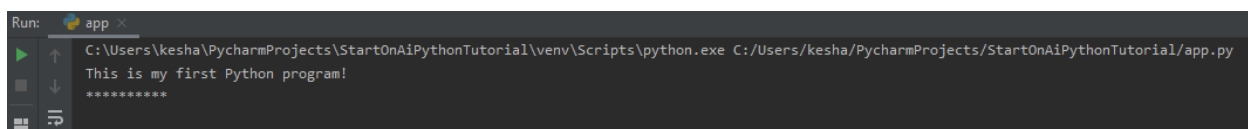
As simple as it seems, let us take a moment to digest what truly happened when we pressed run. The Python translator first identified the location on our computer where our file was located, and that is the first automated output that we see. Next the code that we entered was returned to us just the way we wanted it.

Now for something cool, let us take our understanding of prints and create an expression. An expression is simply code that produces a value.



```
app.py x
1 print('This is my first Python program!')
2 print('*' * 10)
```

The interesting thing about our second line of code is that we have 2 different types of elements. The first is a string, which consists of character(s), then we have an asterisk, which is called an operator. An operator is an element that performs a certain task, in this case multiplication, and finally we have an integer which is simply a number.



```
Run: app x
C:\Users\kesha\PycharmProjects\StartOnAiPythonTutorial\venv\Scripts\python.exe C:/Users/kesha/PycharmProjects/StartOnAiPythonTutorial/app.py
This is my first Python program!
*****
```

As expected, our expression returned to us an intuitive 10 asterisks.

Variables – Next, we move on to variables, useful tools that exist in virtually every programming language you can think of! Variables are fundamental tools that allow users to store important values in pockets of memory in the computer to be retrieved later (similar to putting mail in a mailbox to be retrieved later on!)

Let us define a variable and print it. Copy the following code for the designated output.

```
app.py x
1 price = 10
2 print(price)
```

Now we can see that we have labeled a variable we called “price” and states that we want to store the number 10 inside of it. We said `print(price)` instead of `print(“price”)`, but why? This is because using double quotes automatically classifies that instance as a String, even if a variable has the same name as the string. For this reason, we don’t use quotes so Python knows we want to print the variable price.

```
Run: app x
C:\Users\kesha\PycharmProjects\StartOnAiPythonTutorial\venv\Scripts\python.exe C:/Users/kesha/PycharmProjects/StartOnAiPythonTutorial/app.py
10
```

As we wanted, we have successfully returned the value of our variable. There are many other variable types too, let us look at floats and Boolean types variables in addition to integers and Strings, and their respective outputs. (from this point I will not include the library of the output to conserve space).

```
app.py x
1 cost = 20
2 excellence = 9.8
3 name = "Roger"
4 is_correct = True
5 print(cost)
6 print(excellence)
7 print(name)
8 print(is_correct)
```

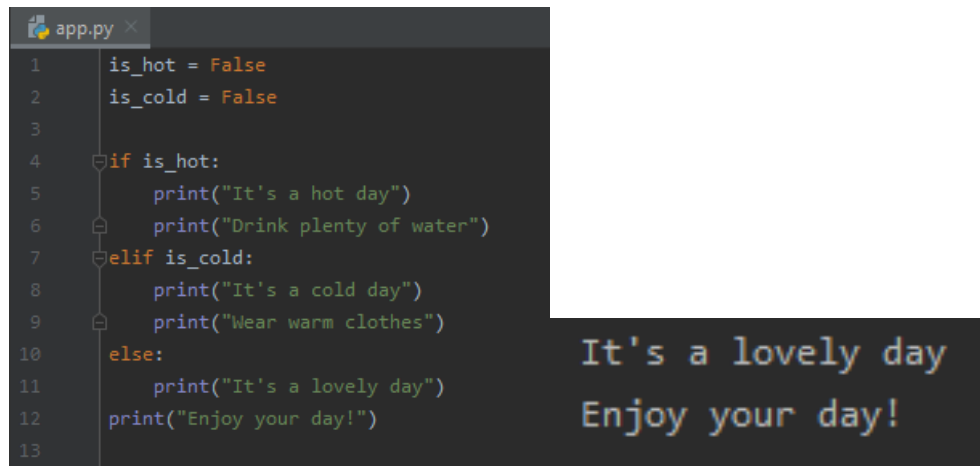
20
9.8
Roger
True

As we see here floats are essentially integers that have decimal points, and booleans are true and false variables, they can only hold these 2 values. Interestingly, we set `is_correct` to `True` with a capital T as Python is case-sensitive, so it differentiates between capital and lowercase letters. For this reason, it is standardized to keep all variables lowercase. `True` is capitalized because it is a key word in Python that it built into the libraries, for example if we set `is_correct = true`, Python will not interpret it as a boolean so this is a careful distinction we must take care of when coding in Python.

If Statements – If statements are essential for Python because they allow the computer to make decisions based off of certain conditions that are set. This is useful for ML because the entire purpose of a ML program is for the computer to compare its “knowledge” with external

attributes that it will be tested against to see if it can accurately peruse through a given situation.

To understand this concept, we will create a simple program that tells the user to what actin to do depending on the type of day. For instance, if it a hot day, the user will be told to drink water, and if it is a cold day, the user will be told to wear warm clothes etc. Here is our code and the output.

The image shows a code editor window titled 'app.py' with a dark background. The code is written in Python and uses syntax highlighting. It defines two variables, 'is_hot' and 'is_cold', both set to False. It then uses an 'if' statement to check 'is_hot', an 'elif' statement to check 'is_cold', and an 'else' statement for the default case. The output of the program is shown in a separate window to the right, displaying the text 'It's a lovely day' and 'Enjoy your day!' on two separate lines.

```
1 is_hot = False
2 is_cold = False
3
4 if is_hot:
5     print("It's a hot day")
6     print("Drink plenty of water")
7 elif is_cold:
8     print("It's a cold day")
9     print("Wear warm clothes")
10 else:
11     print("It's a lovely day")
12     print("Enjoy your day!")
13
```

It's a lovely day
Enjoy your day!

We first identify 2 variables that tell us whether it is a hot day or a cold day, and then we begin our conditions. We first say if it's a hot day, tell the user to drink water. This statement occurs if `is_hot` is true. Next, we have “elif” representing else if, which means if `is_hot` is false, and if `is_cold` is true, tell the user to wear warm clothes. Now since both `is_hot` and `is_cold` is false, we go to our final else condition and tell the user that it is a lovely day, and we tell them to enjoy their day. Since the final print condition is not indented, this tells us that is it not a part of the condition, meaning it will print regardless of the value of `is_hot` and `is_cold`. The catch to this problem is that if both `is_hot` and `is_cold` were true, the program would only think it's a hot day since that is the first condition, however since it cannot be a hot and cold day at the same time, our program works under realistic conditions.

While/For Loops – The concept of looping is significant as it allows us to repeatedly do a task without having to hard-code it for the total number of times that we want to complete the task. While loops make us set a condition that must be satisfied for the program to continually run, and for loops let us set the total number of times that we want to run a task. A common error on these programs are infinite loops (situations where the condition is never met) and this causes the program to run on forever and eventually crash. This is something that novice coders should definitely avoid when developing coding habits. Let's look at a sample while loop with output.

```
app.py x 1
1 i = 1 2
2 while i <= 5: 3
3     print(i) 4
4     i = i + 1 5
5 print("Done") 6
6 Done
```

Let us digest this program. We first define *i* as simply an index that is set to 1 and state that we will continuously run our program until our *i* is greater than 5 (namely 6). Once Python peruses this code it prints 1-5 as we are increasing *i* by 1 every time we go through the while condition and print it. Notice that we importantly increased the value of *i* by one every time we enter the loop, if we did not do this, the loop would never end and we would have an infinite loop! Below let us look at some other common comparison operators which can be used in while loop and if statement conditions.

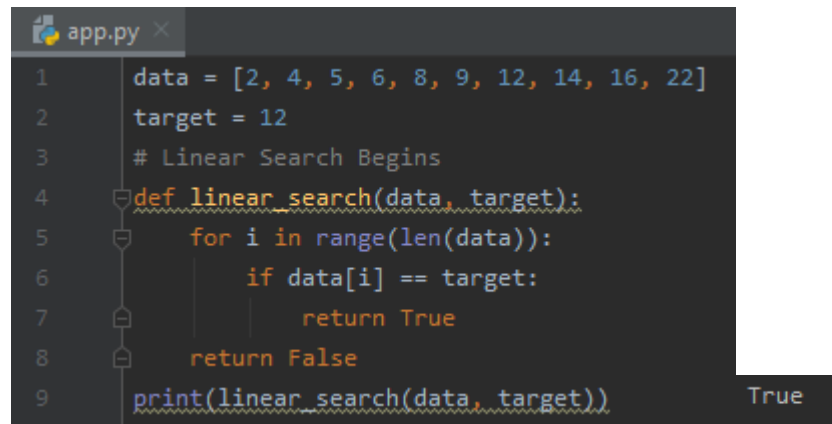
```
a > b
a >= b (greater than or equal to)
a < b
a <= b
a == b (equals)
a != b (not equals)
```

Now let us move onto for loops, which have the same output but run on a different concept. Here is the code and output for a for loop that completed the same task as our while loop.

```
app.py x 1
1 for item in range(1, 6): 2
2     print(item) 3
4 5
```

Amazingly, just 2 lines of code give us the same output! To understand this, not that we begin with the key word *for*, signifying a for loop. Next, we define a variable that the loop will iterate or pass through as Python needs a placeholder to store the values that are being read. In this case it is “*item*”. Next, we say “*in*” which signifies that this loop is passing through the “*range*” of numbers 1-6. However, the “*range*” function in python reads number from (first, last – 1) meaning that we truly are printing the number 1-5 as can be seen by our end output. Essentially, for loops are much faster to use when there is no condition to monitor, and we can even see the significance of loops for ML later on in this tutorial.

Linear Search Simple Algorithm – Let us know take our knowledge and create a simple algorithm called a linear search. An algorithm is basically a structure that can look at many types of data whether if it is a list, a tree and is basically a set of instructions to perform a certain task using those data types. Here we have created a simple linear search which allows us to peruse data and tell us if a certain number exists within a set of data. *Don't get daunted by the long explanation below, its not as hard as it looks, trust us! Take your time when coding it for better comprehension of what you are actually telling the computer to do in each line :)



```
app.py x
1 data = [2, 4, 5, 6, 8, 9, 12, 14, 16, 22]
2 target = 12
3 # Linear Search Begins
4 def linear_search(data, target):
5     for i in range(len(data)):
6         if data[i] == target:
7             return True
8     return False
9 print(linear_search(data, target)) True
```

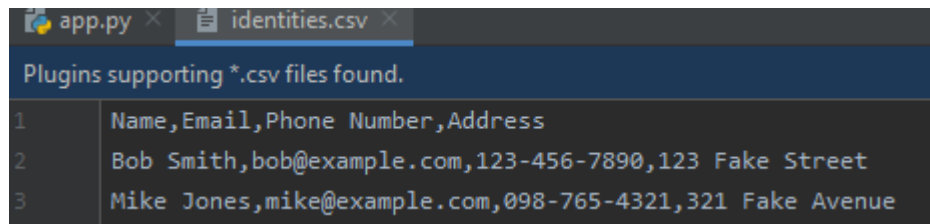
Let us know take apart this algorithm step by step to understand what is occurring in this seemingly daunting program. First, we simplify define the data in a variable called “data” which has numbers inside of square brackets []. This is the instantiation (defining) of an array, which is simplify a collection of multiple smaller data points. For example, 2 itself would be an integer if our statement said “data = 2” however we have many integers, so we can all put them in one array instead of separating them into each of their own individual integers, which would be both inefficient and a waste of memory. Next, we set our target value as 12 as we want to see if our data contains the number 12. Now we define our function as linear_search using “def”. The “def” keyword tells the computer that we are writing a piece of code that can be applied to any situation no matter what data we fill the parameters with (as long as the same type of variable is put into the parameters because we cant put a integer where a String is supposed to go). Now what parameters are we referring to? The parameters (values to be tested) here are “data and target” in line 4 of the code. These are the values that the user will substitute with data when they call the function to perform a task. Defining and calling a function are different things. Defining a function is giving it a specific sent on instructions to do each time like teaching a person HOW to make a pencil. Calling the function would be like us telling the person HOW MANY pencils to make using the instructions the person was given when we defined the function. In our scenario, we will call the function with the data and the target as the parameters because we want to test those specific values against out function to see whether if it is true that the number 12 is inside our data set.

Line 5 of the code is a simple for loop that peruses all the data as it goes through the length (len) of data as this is the defined range. *len is the function that tells the length of the data.

Line 6 tells us that if the data at position *i* in the array is equal to the target, it is true that the target exists so we will return true to the inputter. The form `data[i]` represents data at position 0, data at position 1 etc. For example in the first iteration the computer will look at `data[0]` which is equal to 2 (we start numbering from 0 in arrays so data at position 1 is really the second number in the array so 4 for that instance) and since it is not equal to 12, we do not return true and we now look at the next value in the array and keep going. Eventually the data will be perused, or we will have found our answer!

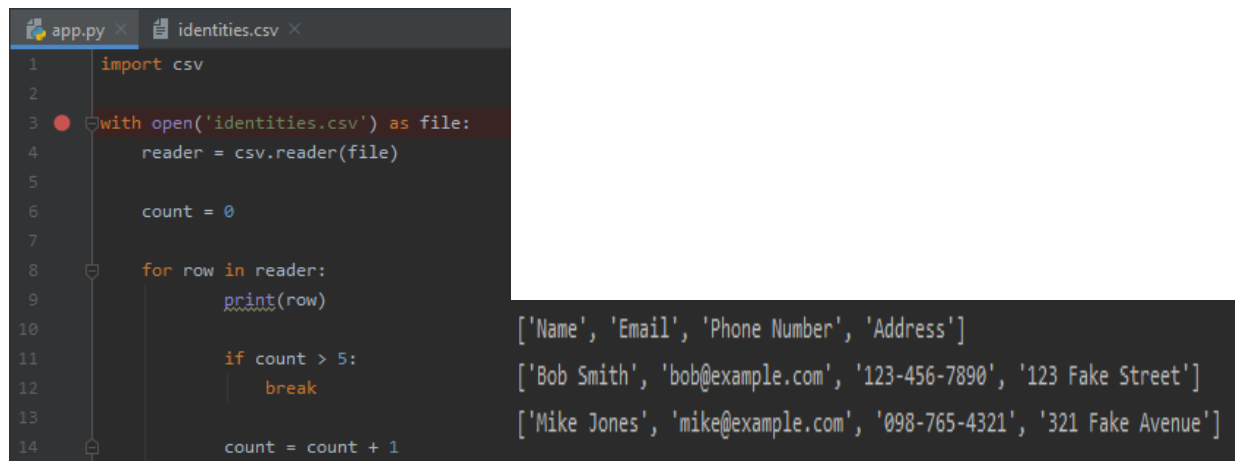
Reading from a .csv Type File

Reading external files is an incredibly useful tool in Python because at times there is mountains of data and there is no way we can input it into the code by hand as it is inefficient, or just impractical. A .csv file stands for “comma spaced values” file which means the file has a lot of data separated by just commas. This comma separation lets us do interesting things when we split the data into pieces and organize it. Let’s take a look at a csv file that I have created that my program will read and display data from.



```
app.py x identities.csv x
Plugins supporting *.csv files found.
1 Name,Email,Phone Number,Address
2 Bob Smith,bob@example.com,123-456-7890,123 Fake Street
3 Mike Jones,mike@example.com,098-765-4321,321 Fake Avenue
```

Here we see that we have lists of data separated by commas, fitting the definition of a csv file.



```
app.py x identities.csv x
1 import csv
2
3 with open('identities.csv') as file:
4     reader = csv.reader(file)
5
6     count = 0
7
8     for row in reader:
9         print(row)
10
11         if count > 5:
12             break
13
14     count = count + 1
```

```
['Name', 'Email', 'Phone Number', 'Address']
['Bob Smith', 'bob@example.com', '123-456-7890', '123 Fake Street']
['Mike Jones', 'mike@example.com', '098-765-4321', '321 Fake Avenue']
```

We see that as a result the data is neatly formatted and separated to our liking. Let us see learn this is done. First, we import the csv library which means we want to call the pre-programmed functions that Python developers had written for us that allow us to use certain methods that apply to .csv files specifically. We then open our identities.csv file and store it as

a file variable essentially just transferring all the data so we don't change the csv files contents. Next, we store the file as an object of the csv class, basically allowing us to use the methods of the csv library to edit the data that was inside of the identities file. Then we loop through the contents of the identities file which is now in the reader object and print out each row of data in an organized fashion, which is done automatically for us and can be seen in the output. The if statement with count provided below is not significant in this case, but in case we had 1000 rows of data and wanted to only print the first 5 rows, it would be useful at that point. If say we wanted to print just the emails, we should change `print(rows)` to `print (rows [1])` since that is the data that is being displayed in the second position of each array.

3. Concepts of Python Useful to Machine Learning

Concept 1 – Variables and ML – Just like the way that we have individualized variables for Python that are built in, there are variables that are specific for NumPy. Because ML is so much more involved than Python which involves standard coding without storing data and results, there are many more NumPy variables in comparison to Python variables due to this notion. Each grid that is created using NumPy stores data of a certain type and it is these different data types that are accessible from the extended NumPy library that are incorporated inside the grids to create an extensive network for ML.

Concept 2 – Loops and ML - Integrating ideas from concepts of Python to our main goal of understanding machine learning as a whole is extremely important and intuitive. Similar to the way that we can loop through a set of data using while loops or even for loops if needed, we can complete a traverse of a NumPy array or a tuple in a ML program. A NumPy array is a special array that lets us grid many values with indices that are non-negative integers which works especially well for ML. Tuples on the other hand relate data and group it together as assigning tuples occurs simultaneously rather than in the sequence that it is in, which makes it very useful for swapping values. For these reasons these concepts of Python go hand in hand with those that make ML so efficient and useful.

Concept 3 – Algorithms and ML – The highest level of computer science is completely dominated by the effective use of algorithms that can perform as efficiently as possible. Just like we developed an algorithm for Python, there are many algorithms that are commonly used in ML. Machine learning algorithms are generally based off of 3 concepts, either supervised, unsupervised or reinforced learning. Supervised learning involves checking predictors against a set of target variables and examples of this include KNN, Regression and Decisions Trees. Unsupervised learning does not involve a target and is just a cluster of data and examples include K-means and Apriori. Reinforced learning trains the algorithm to make specific decisions using trial and error and examples include the Markov Decision Process.

Concept 4 - What is Jupyter? – Jupyter Notebook itself is an IDE that allows for an even faster and more efficient coding experience for a programmer. It is truly a combination of an IDE and a tool that lets you present your data in a very structured manner using data graphs and tables. This becomes exponentially necessary in ML as many times a programmer needs to use data entries and find the results of a program in a visual manner as there is too much input that happens at one time to try and comprehend one at a time. Installing Jupyter Notebook is definitely more complicated than Python itself and would be complicated to understand with just words, so here are a videos that goes through the steps fairly well for Windows Users: <https://www.youtube.com/watch?v=4dCqoHqYxAg> and here is a video for Mac Users: https://www.youtube.com/watch?v=5Ce_V99M20s

4. References

- A. Hoffman, C. (2018, April 16). What Is a CSV File, and How Do I Open It? Retrieved from <https://www.howtogeek.com/348960/what-is-a-csv-file-and-how-do-i-open-it/>
- B. What is Python? Executive Summary. (n.d.). Retrieved from <https://www.python.org/doc/essays/blurb/>
- C. Why Use Python for AI and Machine Learning? (2020, February 6). Retrieved from <https://steelkiwi.com/blog/python-for-ai-and-machine-learning/>
- D. 9. Tuples¶. (n.d.). Retrieved from <http://openbookproject.net/thinkcs/python/english3e/tuples.html>
- E. Ray, S., & Business Analytics. (2020, March 27). Essentials of Machine Learning Algorithms (with Python and R Codes). Retrieved from <https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/>