Introduction to Data Science

**Data Science** is the science which uses computer science, statistics and machine learning, visualization and human-computer interactions to collect, clean, integrate, analyze, visualize, interact with data to create data products.

Business Digital Data contains useful information for organizational growth. It helps managers to analyse opportunities and problems, develop strategies and be proactive to customer needs.

Use cases:

* Healthcare – Predict the outbreak of diseases, right treatment at right stage, AI devices
* Insurance – recommend right plan for right customer
* Banks – cross-sell, upsell, detect fraud
* Education – Enrollment to right course
* Manufacturing – product innovation, JIT
* Retail & eCommerce – recommend right products, increase scale of operations
* Telecom – balance traffic, maximise profit, churn
* Marketing & Retail - understand customer behavior
* Human Resources – allocate right person for right job, performance appraisal
* Government Subsidies & Programs – maximise reach, minimize leakages
* Defense & Space & Missile technologies – increase longevity of satellites, detect resources
* Automobile – driverless cars, pollution reduction, safety
* Transportation Services – manage traffic flow
* Law enforcement – face recognition
* Cinema – animation, movie translation to multi-languages
* Home security – smart homes
* Internet of Things – performance and accuracy

stages of BA

why BA and what is BA

* To manage business uncertainities and create wealth
* Business enterprises relying on processes than personality
* Business analytics provides insights from electronic data
* Business analytics allows your business to make predictive analysis rather than reacting to changes in data. By carefully monitoring data and sharing information openly across your organization, you can take a data-driven approach to decision making

**Tools and process of BA,**

* excel, R programming, Python…
* CRISP-DM

**skills**

* software programming skills
* statistics and machine learning
* business domain skills and experience

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

What can Python do?

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

Why Python?

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

* The most recent major version of Python is Python 3. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* Python can be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Popular versions: Anaconda, Pycharms, Atom, Google Colab, Intel Python, Eclipse, Netbeans, Jupyter

**Introduction to anaconda**

opensource works on Win/Mac/Linux has more than 1.9M users

Quickly download 1,500+ Python/R data science packages

Manage libraries, dependencies, and environments with [Conda](https://conda.io/docs/)

Develop and train machine learning and deep learning models with [scikit-learn](https://scikit-learn.org/stable/), [TensorFlow](https://www.tensorflow.org/), and [Theano](https://pypi.org/project/Theano/)

Analyze data with scalability and performance with [Dask](https://dask.org/), [NumPy](http://www.numpy.org/), [pandas](https://pandas.pydata.org/), and [Numba](http://numba.pydata.org/)

Visualize results with [Matplotlib](https://matplotlib.org/), [Bokeh](https://bokeh.pydata.org/en/latest/), [Datashader](http://datashader.org/), and [Holoviews](http://holoviews.org/)

**Keyword and identifiers**

* keywords (reserved words in Python) and identifiers (names given to variables, functions, etc.).
* We cannot use a keyword as a [variable name](https://www.programiz.com/python-programming/variables-datatypes), [function](https://www.programiz.com/python-programming/function) name or any other identifier. They are used to define the syntax and structure of the Python language.
* In Python, keywords are case sensitive. There are 33 keywords in Python 3.7.
* All the keywords except True, False and None are in lowercase and they must be written as it is. The list of all the keywords is given below.
  + False class finally is return
  + None continue for lambda try
  + True def from nonlocal while
  + and del global not with
  + as elif if or yield
  + assert else import pass
  + break except in raise
* An identifier is a name given to entities like class, functions, variables, etc. It helps to differentiate one entity from another.
* Identifiers can be a combination of letters in lowercase **(a to z)** or uppercase **(A to Z)** or digits **(0 to 9)** or an underscore \_. Names like myClass, var\_1 and print\_this\_to\_screen, all are valid example.
* An identifier cannot start with a digit. 1variable is invalid, but variable1 is perfectly fine.
* Keywords cannot be used as identifiers.

**Comments, indentations and statements**

* Comments
* Python has commenting capability for the purpose of in-code documentation.
* Comments start with a #, and Python will render the rest of the line as a comment:
* Comments can be placed at the end of a line, and Python will ignore the rest of the line:

#This is a comment.  
print("Thanks Persistence, your training is useful!")

* Multi Line Comments
* Python does not really have a syntax for multi line comments.
* To add a multiline comment you could insert a # for each line or use ‘’’:
* """  
  This is a comment  
  written in  
  more than just one line  
  """  
  print("Hello, World!")
* Python Indentation
* Indentation refers to the spaces at the beginning of a code line.
* Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important.
* Python uses indentation to indicate a block of code.

A=1234

B=4321

if A > B:  
 print("A is greater than B!")   
else:  
     print("A is lesser than B!")

if Training == “Good”:

print(“Training is really Good & useful”)

for i in range(1,11):

print(i)

if i == 5:

break

* Instructions that a Python interpreter can execute are called statements. For example, a = 1 is an assignment statement. if statement, for statement, while statement etc. are other kinds of statements
* Multi-line statement: In Python, end of a statement is marked by a newline character. But we can make a statement extend over multiple lines with the line continuation character (\). For example:
* marks = 1 + 2 + 3 + \
* 4 + 5 + 6 + \
* 7 + 8 + 9
* This is explicit line continuation. In Python, line continuation is implied inside parentheses ( ), brackets [ ] and braces { }. For instance, we can implement the above multi-line statement as
* Marks2 = (1 + 2 + 3 +
* 4 + 5 + 6 +
* 7 + 8 + 9)
* print statement is often used to output variables. To combine both text and a variable, Python uses the + character:
* x = “today”; print(“Training is ”+x)
* For numbers, the + character works as a mathematical operator: Profit = 100; Cost=80;Profit+Cost

**Variables and data types in python**

x = 5  
y = " Training is really Good!"

* Variables do not need to be declared with any particular type and can even change type after they have been set.
* String variables can be declared either by using single or double quotes:
* A variable can have a short name (like x and y) or a more descriptive name (age, carname, total\_volume). Rules for Python variables:
* A variable name must start with a letter or the underscore character
* A variable name cannot start with a number
* A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and \_ )
* Variable names are case-sensitive (age, Age and AGE are three different variables)
* Python allows you to assign values to multiple variables in one line:
* x, y, z = "Orange", "Banana", "Cherry"  
  print(x)  
  print(y)  
  print(z)
* x = y = z = "Orange"  
  print(x)  
  print(y)  
  print(z)
* Global Variables: Variables that are created outside of a function (as in all of the examples above) are known as global variables.
* Global variables can be used by everyone, both inside of functions and outside
  + x = "awesome"
  + def training():
  + print("training is " + x)
  + training()
* If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value
  + x = "awesome"
  + def myfunc():
  + x = "fantastic"
  + print("Python training is " + x)
  + myfunc()
  + print("Python training is " + x)
* To change the value of a global variable inside a function, refer to the variable by using the global keyword:
* x = "awesome"  
    
  def myfunc():  
    global x  
    x = "fantastic"  
    
  myfunc()  
    
  print("Python training is " + x)

**Standard input and output**

* Output using print() function Ex: print(‘Training is awesome’)
* print(\*objects, sep=' ', end='\n', file=sys.stdout, flush=False) Ex:print(1,2,3,4,sep=’+’)
* using placeholders{} Ex: x = 5; y = 10; print('The value of x is {} and y is {}'.format(x,y))
* input using prompt. Ex: input([prompt]) num = input('Enter a number: ');print(num)
* A module is a file containing Python definitions and statements. [Python modules](https://www.programiz.com/python-programming/modules) have a filename and end with the extension .py.
* Definitions inside a module can be imported to another module or the interactive interpreter in Python. We use the import keyword to do this.
* For example, we can import the math module by typing in import math

**Operators**

* Operators are special symbols in Python that carry out arithmetic or logical computation
* Add + , Sub -, Multiply \*, Divide /, Modulus %%, Floor //, Exponent \*\*
* Comparison > < >= !=, ==
* Logical and, or, not
* Bitwise &, !, ~, ^, >>, <<
* Assignment
  + Operator Example Equivatent to
  + = x = 5 x = 5
  + += x += 5 x = x + 5
  + -= x -= 5 x = x - 5
  + \*= x \*= 5 x = x \* 5
  + /= x /= 5 x = x / 5
  + %= x %= 5 x = x % 5
  + //= x //= 5 x = x // 5
  + \*\*= x \*\*= 5 x = x \*\* 5
  + &= x &= 5 x = x & 5
  + |= x |= 5 x = x | 5
  + ^= x ^= 5 x = x ^ 5
  + >>= x >>= 5 x = x >> 5
  + <<= x <<= 5 x = x << 5

Identity operators

|  |  |  |
| --- | --- | --- |
| Operator | Meaning | Example |
| is | True if the operands are identical (refer to the same object) | x is True |
| is not | True if the operands are not identical (do not refer to the same object) | x is not True |

Membership operators

|  |  |  |
| --- | --- | --- |
| Operator | Meaning | Example |
| in | True if value/variable is found in the sequence | 5 in x |
| not in | True if value/variable is not found in the sequence | 5 not in x |

Control flows: conditional statements

If else

if test expression:

Body of if

else:

Body of else

if test expression:

Body of if

elif test expression:

Body of elif

else:

Body of else

Nested if

num = float(input("Enter a number: "))

if num >= 0:

if num == 0:

print("Zero")

else:

print("Positive number")

else:

print("Negative number")

Control flow: loops

* for loop in Python is used to iterate over a sequence ([list](https://www.programiz.com/python-programming/list), [tuple](https://www.programiz.com/python-programming/tuple), [string](https://www.programiz.com/python-programming/string)) or other iterable objects. Iterating over a sequence is called traversal

for val in sequence:

Body of for

for i in range(100):

print(i)

fruits = ["apple", "banana", "cherry"]  
for x in fruits:  
  print(x)

* break statement we can stop the loop before it has looped through all the items:
* fruits = ["apple", "banana", "cherry"]  
  for x in fruits:  
    print(x)  
    if x == "banana":  
      break
* continue statement we can stop the current iteration of the loop, and continue with the next
* fruits = ["apple", "banana", "cherry"]  
  for x in fruits:  
    if x == "banana":  
      continue  
    print(x)
* while loop we can execute a set of statements as long as a condition is true
* i = 1  
  while i < 6:  
    print(i)  
    i += 1
* remember to increment I else loop continues for ever

Data structures:

|  |  |  |  |
| --- | --- | --- | --- |
| Text Type: | str | Training = ‘awesome’  Training = ‘’’Python  Training  Is awesome’’’ | Training[1:5]  Training[-5:-2]  Training.upper()  Training.lower()  Training.replace(‘T’,’R’)  Concat = A + B |
| Numeric Types: | int, float, complex | x = 1    # int y = 2.8  # float z = 1j   # complex | x+y |
| Sequence Types: | list, tuple, range | langlist = ["java", "c++", "python"] print(langlist)  newlang = [‘jira’,’aws’]  thistuple = ("apple", "banana", "cherry") print(thistuple)  longlist[1:2] | Longlist+newlang  Longlist[1]=‘c’  Longlist=[‘c’]  x = ("apple", "banana", "cherry") y = list(x) y[1] = "kiwi" x = tuple(y)  print(x) |
| Mapping Type: | dict | automobile = {   "brand": "Ford",   "model": "Mustang",   "year": 1964 } print(automobile) |  |
| Set Types: | set, frozenset |  |  |
| Boolean Type: | bool |  |  |
| Binary Types: | bytes, bytearray, memoryview |  |  |

* You can get the data type of any object by using the type() function
* Strings are arrays. Square brackets can be used to access elements of the string. You can return a range of characters by using the slice syntax. You can use negative indexes to start the slice from the end of the string

**List, Tuples,Dictionary,Strings**

* **List** is a collection which is ordered and changeable. Allows duplicate members.
* **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
* **Set** is a collection which is unordered and unindexed. No duplicate members.
* **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.
* **List methods**

|  |  |
| --- | --- |
| Method | Description |
| [append()](https://www.w3schools.com/python/ref_list_append.asp) | Adds an element at the end of the list |
| [clear()](https://www.w3schools.com/python/ref_list_clear.asp) | Removes all the elements from the list |
| [copy()](https://www.w3schools.com/python/ref_list_copy.asp) | Returns a copy of the list |
| [count()](https://www.w3schools.com/python/ref_list_count.asp) | Returns the number of elements with the specified value |
| [extend()](https://www.w3schools.com/python/ref_list_extend.asp) | Add the elements of a list (or any iterable), to the end of the current list |
| [index()](https://www.w3schools.com/python/ref_list_index.asp) | Returns the index of the first element with the specified value |
| [insert()](https://www.w3schools.com/python/ref_list_insert.asp) | Adds an element at the specified position |
| [pop()](https://www.w3schools.com/python/ref_list_pop.asp) | Removes the element at the specified position |
| [remove()](https://www.w3schools.com/python/ref_list_remove.asp) | Removes the item with the specified value |
| [reverse()](https://www.w3schools.com/python/ref_list_reverse.asp) | Reverses the order of the list |
| [sort()](https://www.w3schools.com/python/ref_list_sort.asp) | Sorts the list |

* A tuple is a collection which is ordered and **unchangeable**. In Python tuples are written with round brackets.you can only Convert the tuple into a list to be able to change it
* set is a collection which is unordered and unindexed. In Python sets are written with curly brackets

Dictionary

* A dictionary is a collection which is unordered, changeable and indexed. In Python dictionaries are written with curly brackets, and they have keys and values.

automobile = {  
  "brand": "Ford",  
  "model": "Mustang",  
  "year": 1964  
}  
print(automobile)

* items of a dictionary can be accessed by referring to its key name, inside square bracket x = automobile[“brand”]
* change the value of a specific item by referring to its key name automobile[“brand”] = “tesla”
* The pop() method removes the item with the specified key name.The del keyword removes the item with the specified key name

**functions:,Introduction ,Types of functions ,recursive functions,Lamda functions**

* function is a group of related statements that perform a specific task
* Once we have defined a function, we can call it from another function, program or even the Python prompt. To call a function we simply type the function name with appropriate parameters
* Types of functions: built-in such as max(), min(), sum() and user defined

Syntax

def function\_name(parameters):

"""docstring"""

statement(s)

def profit(sales,cost):

profit = sales-cost

print(profit)

return

profit(2500,2000)

def amazon(products):

for i in products:

print(i+" mobile")

brands = ['Apple','Samsung','Nokia','Oppo']

amazon(brands)

* Python also accepts function recursion, which means a defined function can call itself
* A lambda function is a small anonymous function.
* A lambda function can take any number of arguments, but can only have one expression
* lambda functions are used when an anonymous function is required for a short period of time

**modules ,Packages,exception handling ,Debugging python**

* Modules refer to a file containing Python statements and definitions
* We use modules to break down large programs into small manageable and organized files. Furthermore, modules provide reusability of code
* A file containing Python code, for e.g.: example.py, is called a module and its module name would be example.

# Python Module example.Save code as example.py

def add(a, b):

"""This program adds two

numbers and return the result"""

result = a + b

return result

* import example. Give some values example.add(4,5.5)
* import math as m,
* we can import specific names from modules without importing the whole. Example from math import pi
* packages can contain one or more modules. Packages can have sub-packages and modules