Python

leangaurav

Tools

leangaurav

Tools and Softwares to use

- Anaconda
  - Jupyter
  - Sypder
- Editor + CMD/Terminal
  - Notepad++ (windows only)
  - Sublime
  - Atom etc.
- Git and Github
  - Create your **github** account
  - On windows install **gitbash**

leangaurav

Anaconda

- Anaconda is distribution of Python which comes with bunch of tools
- We will use Anaconda3 and tools which come along:
  - Jupyter (mostly)
  - Spyder
- Install Python 3.x version from here:
  - https://www.anaconda.com/distribution/

* For windows if asked to add to PATH, put tick in check box

leangaurav

Notepad++ and Terminal

- You can write python code/scripts in any text editor and run from terminal.
- In any text editor, write your python code and save file with extension *.py*
- Now open a terminal in same folder as your python script and run like this:

  python <filename>

replace file name with your script name and don't put <>

leangaurav

Git and Github

- Git is a tool for version control
- Github is a website that allows people to collectively work on a project
- On Mac/Linux ***git*** comes preinstalled or you can install if running **git** on terminal gives error
- On windows install ***gitbash*** to use git.

leangaurav

Other Tools and Stuff

- Try other **IDE** to write python code like **PyCharm**.

- Books:
  - Learning Python (Beginners)
  - Programming Python (Intermediate)
  - Python Cookbook, Fluent Python (Advanced)

- **pip**: This is important to understand packaging and dependency management

leangaurav

What is Python

leangaurav

Python: Language

- Python is a Programming Language
- Python is an interpreted language.
- But it uses a hybrid model of both compilation and interpretation to improve performance
- Dynamically typed and case sensitive

leangaurav

Compiled vs Interpreted

- Compilation:
  - Convert to binary and save
  - Run saved binary
- Interpretation:
  - Read one instruction at a time
  - Convert to binary and run
  - Repeat above steps till all code is executed
- **Compiled** languages are ***faster*** than interpreted.
- **Interpreted** one give ***platform independence***.

leangaurav

Which python are we using

- Anaconda: collection of some tools(Jupyter etc.) and libraries (pandas, numpy etc.) along with Cpython. That justifies the size difference
- Cpython: official Python implementation (available at python.org)
- Source code of Cpython is written in C programming language
- There are other implementations like Jython, PyPy etc.

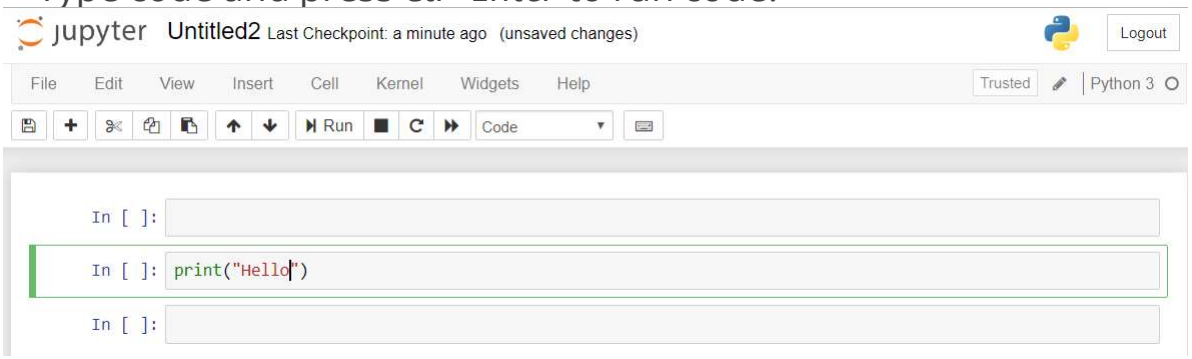
Running Python Code
>>> print(“Hello World”)

Jupyter Notebook

- Jupyter Notebook opens in browser
- On top right there is new button: with Python3 and Folder options

leangaurav

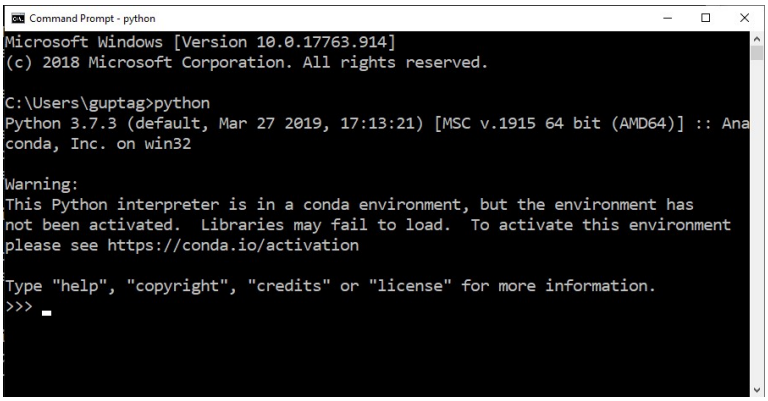
- *New → Python3 :* Creates new notebook (with extension .ipynb)
- Cells: a block where you write code
- Type code and press *Ctr+Enter* to run code.



leangaurav

Python Interpreter

- Open a terminal or CMD and type python (in small)
- Notice **>>>**
- Check Version 3.x.x 3.7.7 here
- Should not be 2.x.x



leangaurav

Notepad++ and Terminal

- You can write python code in a text editor and run in a terminal.
- In a text editor, write your code and save file with extension ***.py***
- Now open terminal in folder where python script is saved and run like this:

  python <filename>

  * replace file name with script name and don't put <>

  Example: if file is saved as test.py

  python test.py

leangaurav

Building Blocks of Code

leangaurav

Tokens

- Four kind of Tokens:
  - Keywords
  - Identifiers
  - Literals
  - Operators
- All code elements fall into one of these category

leangaurav

- Keywords: Special reserved words predefined by language

- List of keywords (Python 3.8)

False	*await*	*else*	*import*	*pass*
None	*break*	*except*	*in*	*raise*
True	*class*	*finally*	*is*	*return*
and	*continue*	*for*	*lambda*	*try*
as	*def*	*from*	*nonlocal*	*while*
assert	*del*	*global*	*not*	*with*
async	*elif*	*if*	*or*	*yield*

leangaurav

- **Identifiers**: These are names.
- Rules:
  - Use letters in lowercase (a to z) or uppercase (A to Z)
  - Digits (0 to 9) allowed but not in beginning
  - Underscore (_) allowed anywhere
  - Should not be name of keyword

- Variable name, Class name, Function name and Module name are all identifiers

- Python has special identifiers (having two underscores):

  _ _*_ _ : Special Reserved system defined names

  _ _* : Used to define private class members

leangaurav

DIY 1

- Read about operator precedence

  https://docs.python.org/3/reference/expressions.html#operator-precedence

leangaurav

- Literals: Constant of fixed values

- Type of literals:

int	:	1,-1,0....
float	:	-1.0, 0.0, 3.14
string	:	‘’, ‘ ’, ‘a’, ‘abcd’
bool	:	True, False
None	:	Empty

leangaurav

- Operators: Python has Unary and Binary operators
  - **
  - +x, -x, ~x
  - *, @, /, //, %
  - +,-
  - in, not in, is, is not, <, <=, >, >=, !=, ==
  - not, and, or
  - >>, <<

leangaurav

Comments

- Comments are pieces of text present in between code.
- Comments start with a # symbol.
- Any text written on right side of # is ignored by python interpreter
- They are used to add some information or description inside code.

```
x = 1 + 2  # this is a comment and will be ignored
```

leangaurav

DIY 2

- Read python naming convention

  https://www.python.org/dev/peps/pep-0008/#naming-conventions

- Write Python code to import and print list of all keywords

- What is **PYTHONPATH**

leangaurav

DIY 2

- Print List of Keywords

```
C:\Users\guptag>python
Python 3.7.3 (default, Mar 27 2019, 17:13:21) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on
win32

Warning:
This Python interpreter is in a conda environment, but the environment has
not been activated.  Libraries may fail to load.  To activate this environment
please see https://conda.io/activation

Type "help", "copyright", "credits" or "license" for more information.
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue'
, 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', '
in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with'
, 'yield']
>>> _
```

leangaurav

Data types, variables and some Built-in Functions

leangaurav

Data Types

- Data types primarily tell two things:
  - Behaviours or operations that can be done on the data
  - How much memory is used to store data
- In python we look only at behaviours most of the times.
- Python has some simple built-in types like other langauges:
  int, float, bool, str

leangaurav

Variables

- Variables are a way of accessing some data stored in memory.
- Python variables are not associated to memory.
- Instead, data takes up memory and variables just refer to these data objects.
- And python is a dynamically typed language.

* Python is also a strictly typed language apart from being dynamically typed

leangaurav

Built-in function: type

- Returns data type of variable/data

```
>>> x  = 10
>>> print(type(x))

<class 'int'>
```

leangaurav

Built-in function: print

- Prints 0-n values on screen

  >>> print(10); print(10, “abcd”); print()

- By default it prints spaces between multiple values and a newline at the end. But these can be customized

  >>> print(1,2,3, sep= ‘#’, end=‘---’); print(“Next”)

  1#2#4---Next

leangaurav

- Print Syntax:

  print(<var/const>, …, sep= ‘<separator>’, end = ‘<delimiter>’)

- *Sep and end need to be string types.*

leangaurav

Built-in function: dir

- Prints names available in current workspace

```
>>> dir()
```

- If an argument is given, it prints attributes/options/functions available inside that argument.

```
>>> print(dir(int)); print(dir(10))
```

leangaurav

Built-in function: input()

- Used to take input from console.
- It returns the entered data always as a string.

```
>>> r = input()
```

- An optional message can be given

```
>>> r = input("Enter something")
```

leangaurav

Type conversion

```
>>> x =  10
>>> y = "10"
>>> print(x, y)
>>> x + y          # exception
```

- Often we need to convert one type to another, like in above example

```
>>> print( x + int(y)) # this works now
>>> float("10.5"); float(10)
>>> int("10"); int(10.5)
>>> bool(''); bool(10); bool("abcd")
```

* Try all combinations you can think of.

leangaurav

DIY

- Explore these functions and try out some examples:
  - **bin() :** returns the binary representation of an integer
  - **hex() :** gives hexadecimal representation of an integer
  - Also explore all the options available for **int()**

leangaurav

Back to operators

- **
- //
- in, not in
- not
- and
- or

leangaurav

Strings

leangaurav

What is a string

- **Sequence** of symbols
- **Immutable**
- Written in quotes

leangaurav

Indexing and Slicing

- Indexing:
  <string>[<integer index>]
- Slicing:
  <string>[start : end]
  <string>[start : end : step]
- Start and end decide the end and start point in string

* Indexes start from 0 and end at (length – 1)

leangaurav

Simple Functions

- len # common function, can be used with other sequences
- upper, lower, strip, lstrip, rstrip
- isupper, islower
- isdigit, isalpha, isalnum, isspace, …
- count, find, replace

leangaurav

Escape sequences

- These are special characters starting with a slash '\'
  - \n: new line
  - \t: tab
  - \b: backspace
  - \r: carriage return
  - \\: slash symbol '\'
  - \": escape a quote in double quoted string
  - \': escape a single quote in single quoted string

There are many more escape sequences. Each counts as one character

leangaurav

Ways of writing a string in python

- “Double Quote”
- ‘Single Quotes’
- “”””Triple using double”””
- ’’’Triple using single’’’

Triple quoted strings are also used a multi-line comments.

leangaurav

Ord and Chr

- All characters are represented by a numeric value.
  - A –> 65, B-> 66 ..
  - a –> 97
- An example of such number encoding is ASCII
  - **ord()** function returns the ascii value of a character
  - **chr()** converts numeric to Character

```
>>> ord(‘A’)
>>> chr(97)
```

leangaurav