INTRODUCTION TO PYTHON

OUTLINE

- Relevance of Python
- Coding environment
- Basics

PYTHON

PYTHON

- Powerful general purpose programming language.
- While it is used for a wide range of applications including web development, automation, and software development, we will focus on its use in data analysis.
- In recent years, Python has become one of the most popular tools for solving data science problems.
- Known for its simplicity and readability, Python offers a wide range of libraries and frameworks that make it particularly well-suited for data analysis, machine learning, and predictive modeling.

WHY PYTHON FOR DATA SCIENCE?

- Open-source high-level programming language
- Comprehensive libraries to conduct explore, manipulate, analyze and visualize data
- Popular language for Data Science
- Integrates seamlessly with other tools

REASONS NOT TO USE PYTHON

- Not point and click
- Functionality depends on user demand and contributions
- No customer support
- Interpreted language will run slower than code written in a compiled language like Java or C++
- Python can be a challenging language for building highly concurrent, multithreaded applications

FEATURES OF PYTHON

- Syntax is simple and readable (compared to low-level languages)
- Object-oriented programming language
- Interpreted language. It can be run from the command line.

1991

Created by Guido van Rossum

2000

Python 2.0

2008

Python 3.0

2010s

Adoption for Data Science with libraries like Pandas and Scikit-learn

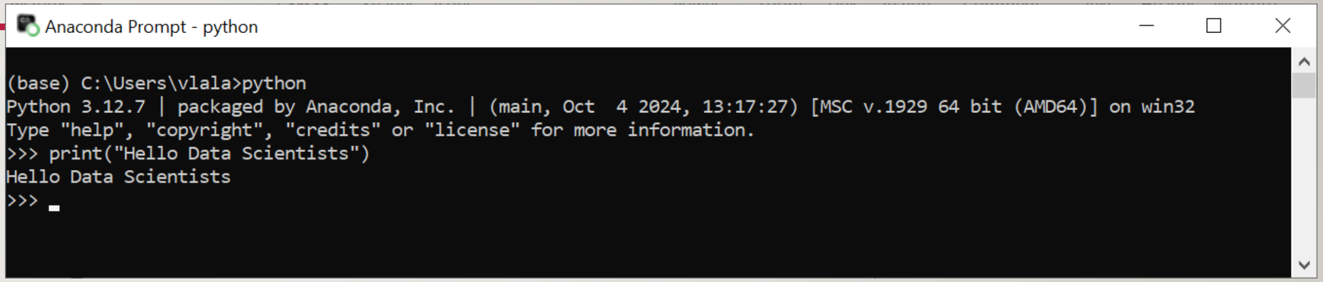
2020s

Most popular Data Science Language

CODING ENVIRONMENT

USING PYTHON

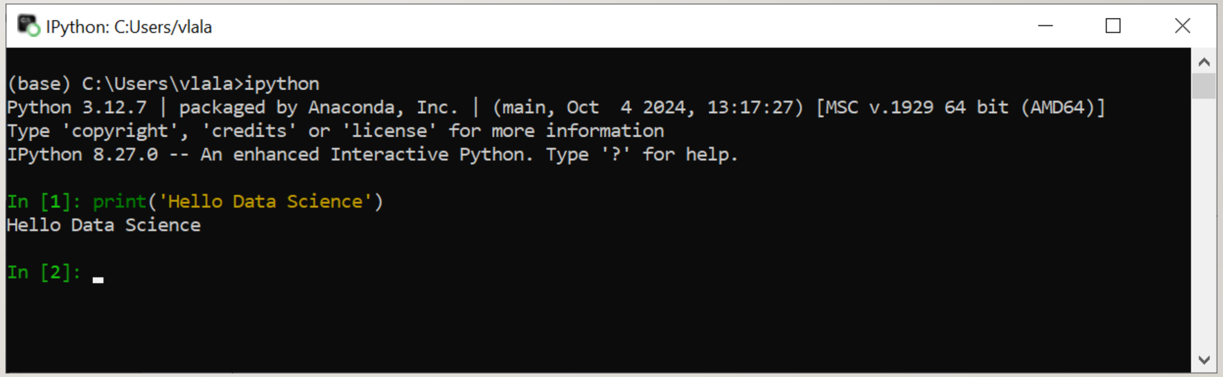
- Command Line



Anaconda Prompt - python

```
(base) C:\Users\vlala>python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello Data Scientists")
Hello Data Scientists
>>>
```

- iPython



IPython: C:Users/vlala

```
(base) C:\Users\vlala>ipython
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)]
Type 'copyright', 'credits' or 'license' for more information
IPython 8.27.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print('Hello Data Science')
Hello Data Science

In [2]:
```

- Jupyter Notebook
- Google Colab, Anaconda Cloud

JUPYTER NOTEBOOK ADVANTAGES

- Readable markdown document including formatted narrative, code, and result
- Tab-completion
- Introspection: Using question mark (?) after a variable or function
- Documentation on Jupyter Notebook

- The Official Python Tutorial

BASICS OF PYTHON

ARITHMETIC

- Python can perform simple arithmetic operations
  - 123 * 456 + 789
  - 123 * (456 + 789)

OBJECT OR VARIABLE ASSIGNMENT

Values are assigned to an object using the "=" operator. It is worth noting that assigning a value to an object creates a reference to the object. It does not actually copy the object. A few more noteworthy aspects of variables in Python are

- The variable or object is always on the left of the assignment operator.
- Python evaluates the information to the right of the assignment operator first.
- If the Python shell is reset, all object values are lost.

OBJECT NAMING CONVENTION

- Object names must start with a letter, and can only contain letters, numbers, and underscore.
- Cannot name objects using reserved words (e.g., else, if)
- Naming style is a personal choice, but whatever your style, be consistent. We will use lower snake case.

SCALAR TYPES

- type()
  - Number:
    - int: 1234
    - float: 1234.567
  - String
    - str: ‘Python’
  - Boolean
    - Bool: True, False
  - Other:
    - None, bytes

TYPECASTING

- Process of converting one type to another.
- Explicit
  - Use type functions: str(), int(), float(), bool(), bytes()
- Implicit
  - isinstance(True + True)

BINARY OPERATORS

- Python has a set of binary operators like in most other languages and use familiar math notation.
- These work as expected with numbers, however they can also work with string types yielding handy results.

- 12 + 5
- 12 – 5
- 12 * 5
- 12 / 5
- "I am " + " smart“
- "I am smart." * 5
- 12 // 5 # floor division
- 12 % 5 # modulo operator for remainder
- 5 == 10 # test for equality

COMMON BINARY OPERATORS

- a = 1
- b = 2
- a + b # Addition
- a - b # Subtraction
- a * b # Multiplication
- a/b   # Division
- a**b # Power function

COMMON BINARY OPERATORS

- a>1 & b>1 # True if both conditions are met
- a>1 | b>1 # True if either condition is met
- a>1 ^ b>1 # True only if either condition is met but not both; like XOR
- a == b # Returns True if condition is met
- a != b # Returns True if condition is met
- a < b # Returns True if condition is met
- a > b # Returns True if condition is met
- a is b # True if a and b reference same object
- a is not b # True if a and b reference different objects

LANGUAGE SEMANTICS

- Python language is distinguished by its emphasis on readability, simplicity, and explicitness.
- Indentation and White Space
  - Python uses white space and indentation to structure code
- Comments
  - Python uses the # mark to indicate a comment. Code following # is ignored by Python.

OBJECT

- Objects have
    - Methods: A function applied to an object
        - my_object.useful_function()
    - Attributes: Characteristic of the object
        - my_object.useful_information

FUNCTION

- A function takes a set of arguments and returns a value
- A function contains a set of instructions or operations
- Functions help reuse code
- Functions are called using parentheses and passing zero or more arguments.

function_name(parameter)

- We “call” or “invoke” a function.

FUNCTION

- Python and its libraries have built-in functions that perform almost all operations desired by a data scientist.
- There are two reasons to create a function: -
  - There is no function to perform the desired task
  - A set of tasks are performed repeatedly (say more than three times).

```
def my_function(parameter_name = argument_value):
        result = do something with argument_value
        return(result)
```

OBJECTS

- Methods
- Attributes

METHOD

- A method is like a function that belongs to an object.
- Like a function, a method can take one or more arguments

```
my_object.my_method(parameter1 = arg1, ..)
```

ATTRIBUTE

- While methods are functions associated with an object, attributes are characteristics of the object
- Attributes are Python objects stored inside the object
- While attributes are called using the dot notation like methods, they do not have parentheses.
  - object.my_method() vs. object.my_attribute

MODULE

- A Python module is simply a file containing Python code (with a .py extension).
- A module can contain functions, classes, variables, or even runnable code.
- You can import the module to use its functions.

MUTABLE AND IMMUTABLE

- Objects that can be modified are mutable.
- Many Python objects such as list, dictionaries, NumPy arrays and most user-defined types (classes) are mutable.
- Others, like strings and tuples, are immutable.

CONTROL STRUCTURES

- Python has several built-in keywords for control structures that resemble those in other programming languages
    - if.. elif… else,
    - for.. in..
    - while..

CONCLUSION

- In this section, we examined
  - Relevance of Python
  - Python coding environments
  - Basics of the Python language