An Introduction to Python

Data Science 2 / Data & AI 3

KdG Karel de Grote Hogeschool

Agenda

1. Introduction
   - Who uses Python, what & Why
2. Variables and data types
3. Program flow, functions, objects and modules
4. Notebook time!

Cursus / Boek

canvas Oefeningen

Introduction

1

Introduction

Python is popular

Python is implicitly compiled

Explicitly compiled to machine code

Explicitly compiled to byte code

Implicitly compiled to byte code

Purely interpreted

C, C++, Fortran

Java, C#

Python

Shell, Perl

Introduction

What is Python used for?

Introduction

Why Python for Data Engineering?

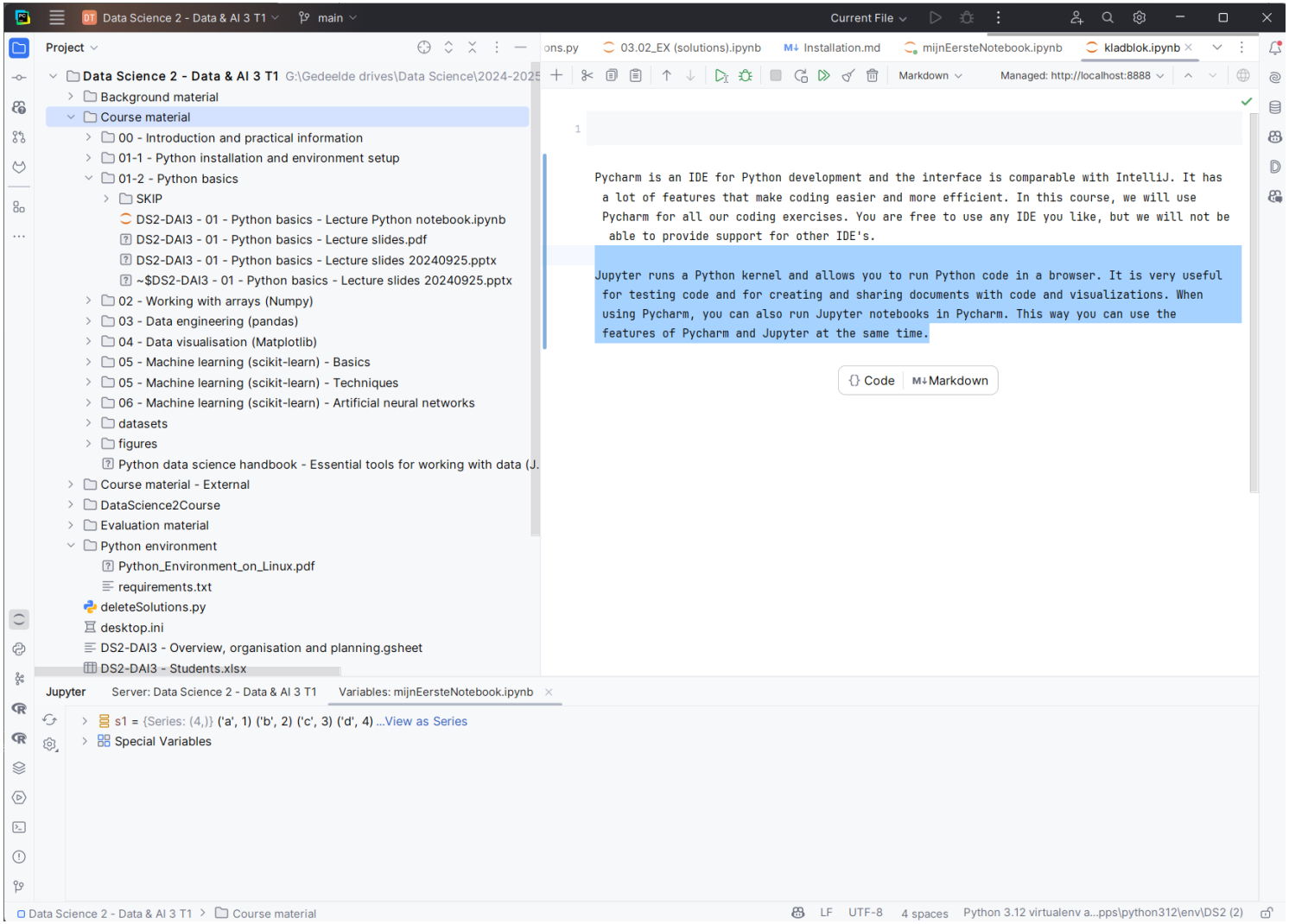
Extensive Data Engineering Libraries

- **NumPy** homogeneous array-based data manipulation
- **Pandas** heterogeneous and labeled data manipulation
- **SciPy** scientific computing tasks
- **Matplotlib** publication-quality visualizations
- **Jupyter** interactive notebooks and sharing of code
- **Scikit-Learn and Tensorflow**: machine learning

Beginner friendly programming language

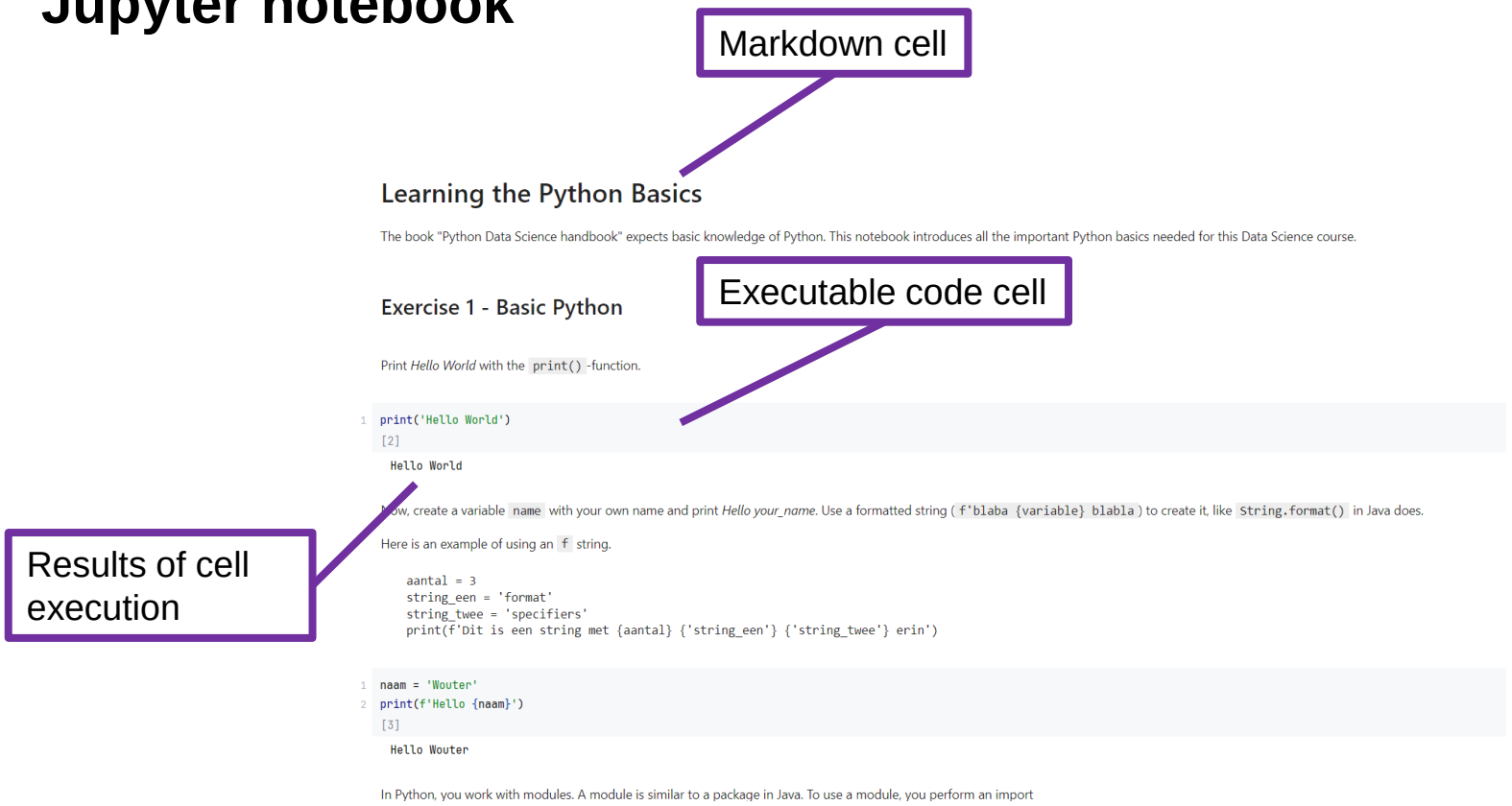
Introduction / Running Python

PyCharm



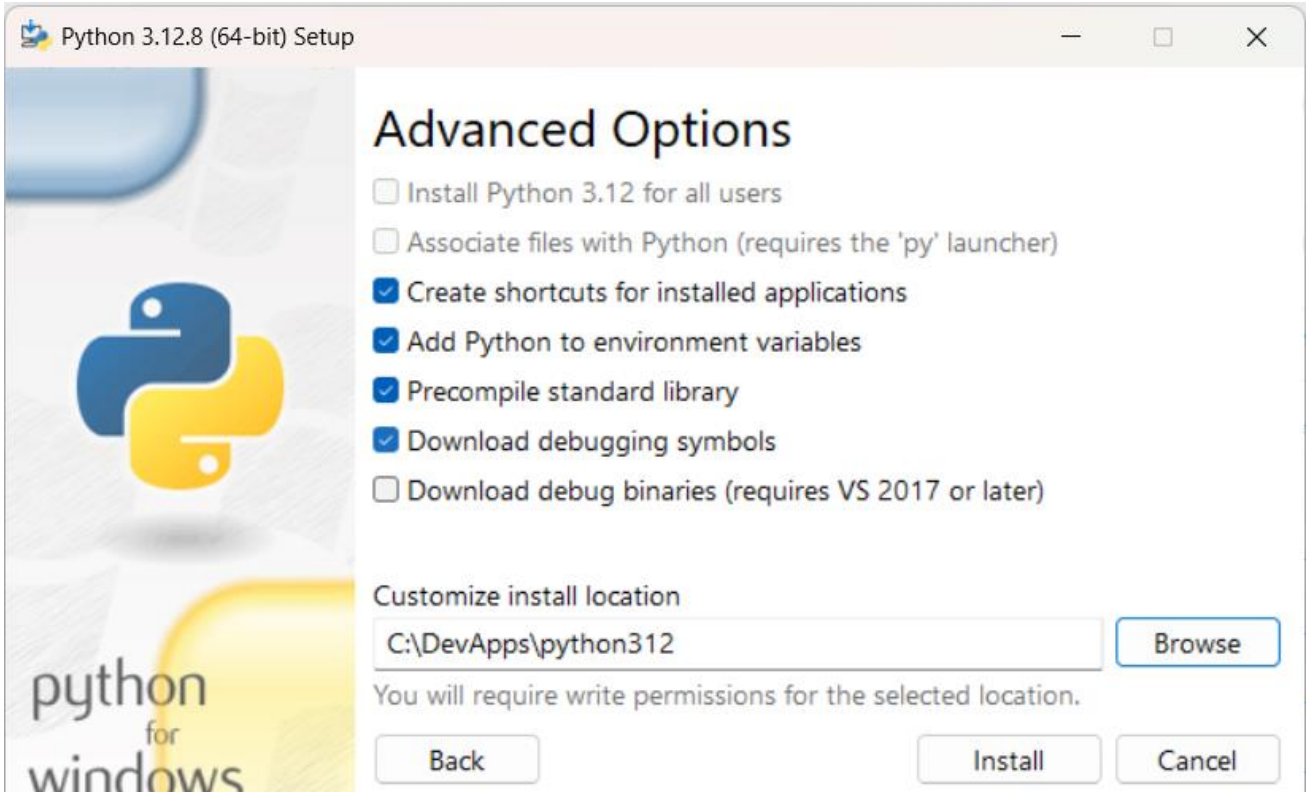
Introduction / Running Python

Jupyter notebook



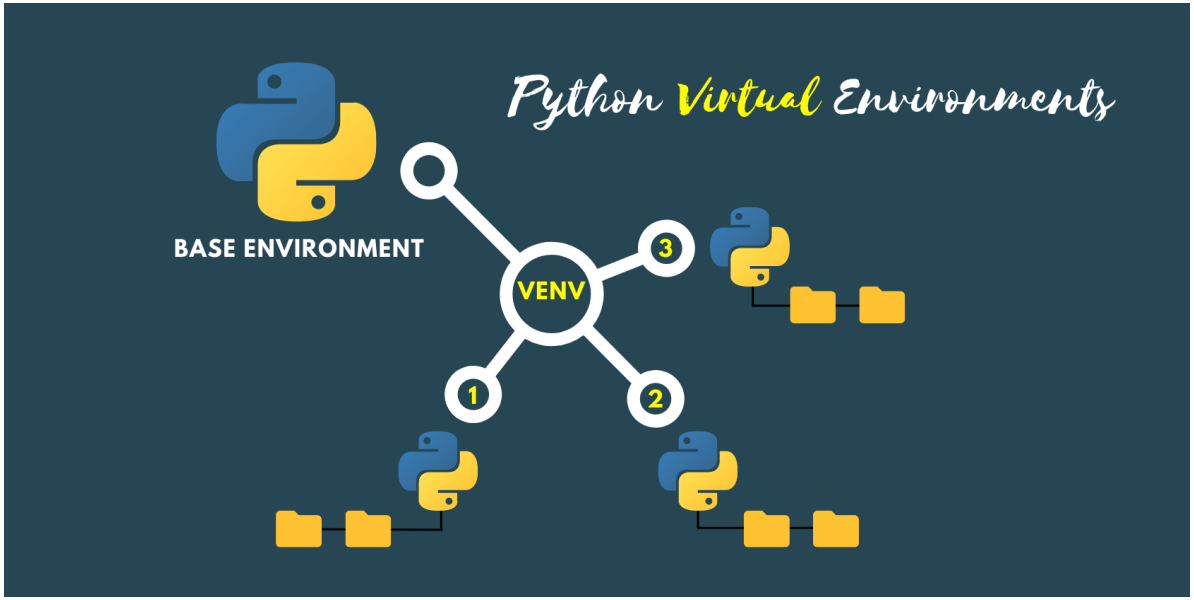
Introduction / Installation

Step 1: Install Python



Introduction / Installation

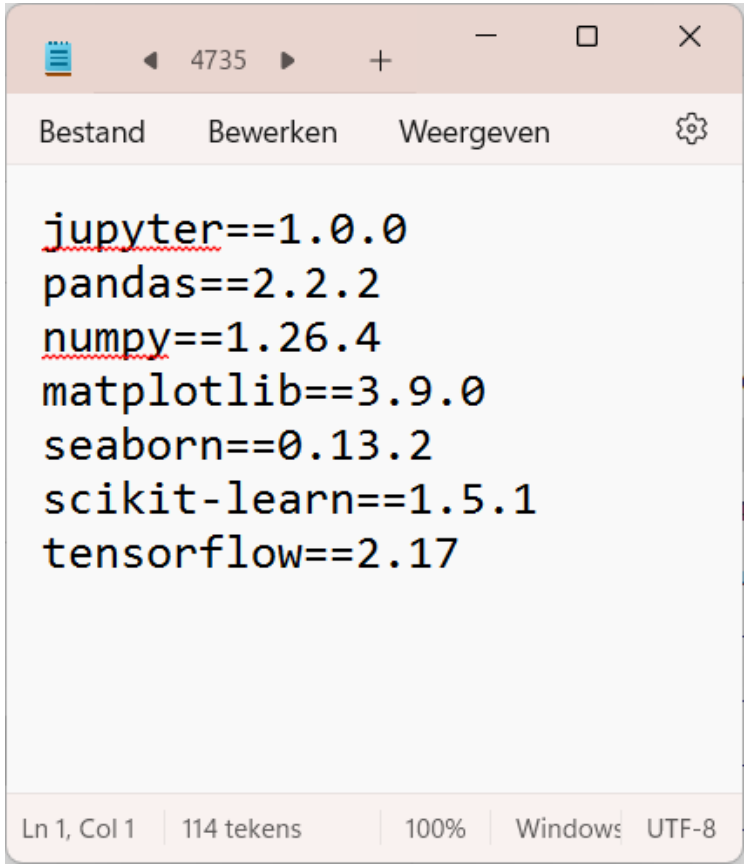
Step 2: Create a virtual environment



Source: Geekpyhton.in

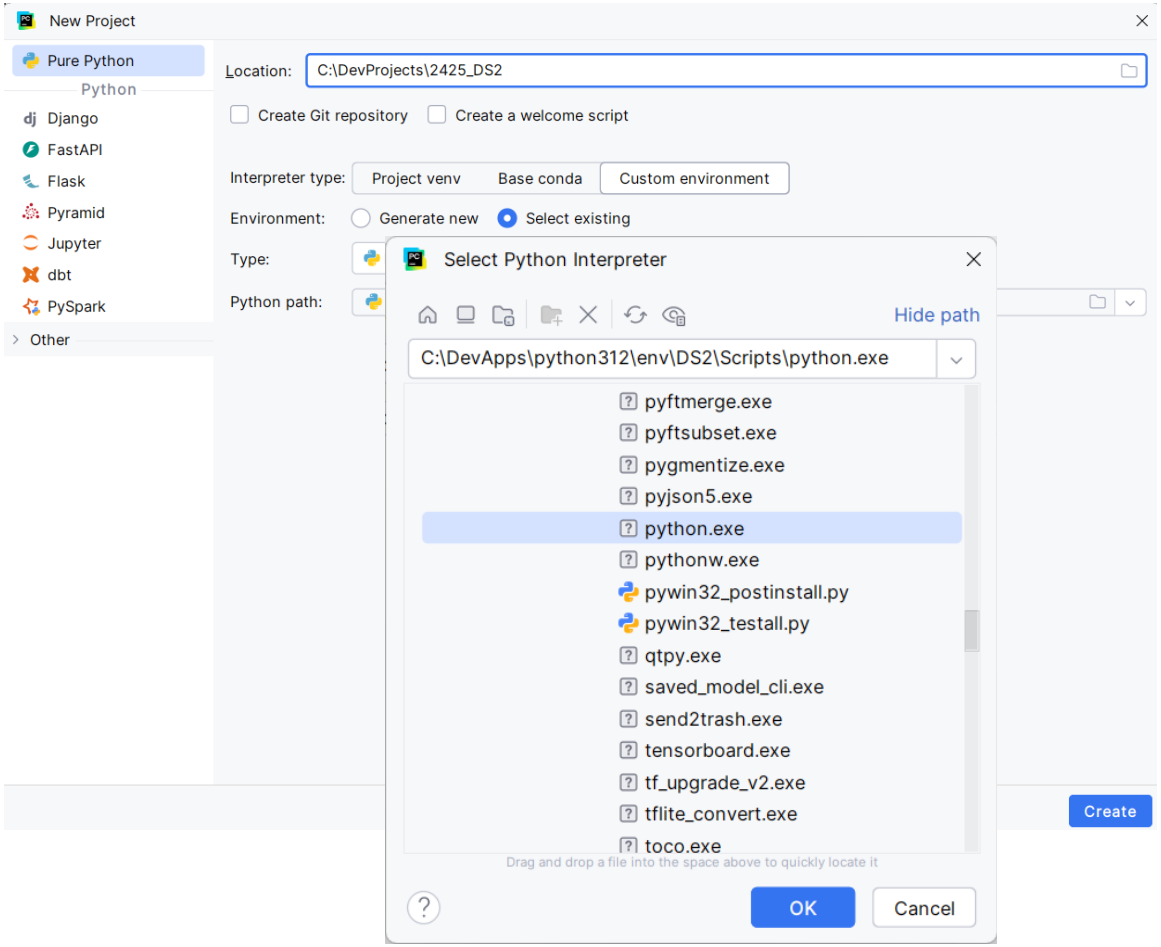
Introduction / Installation

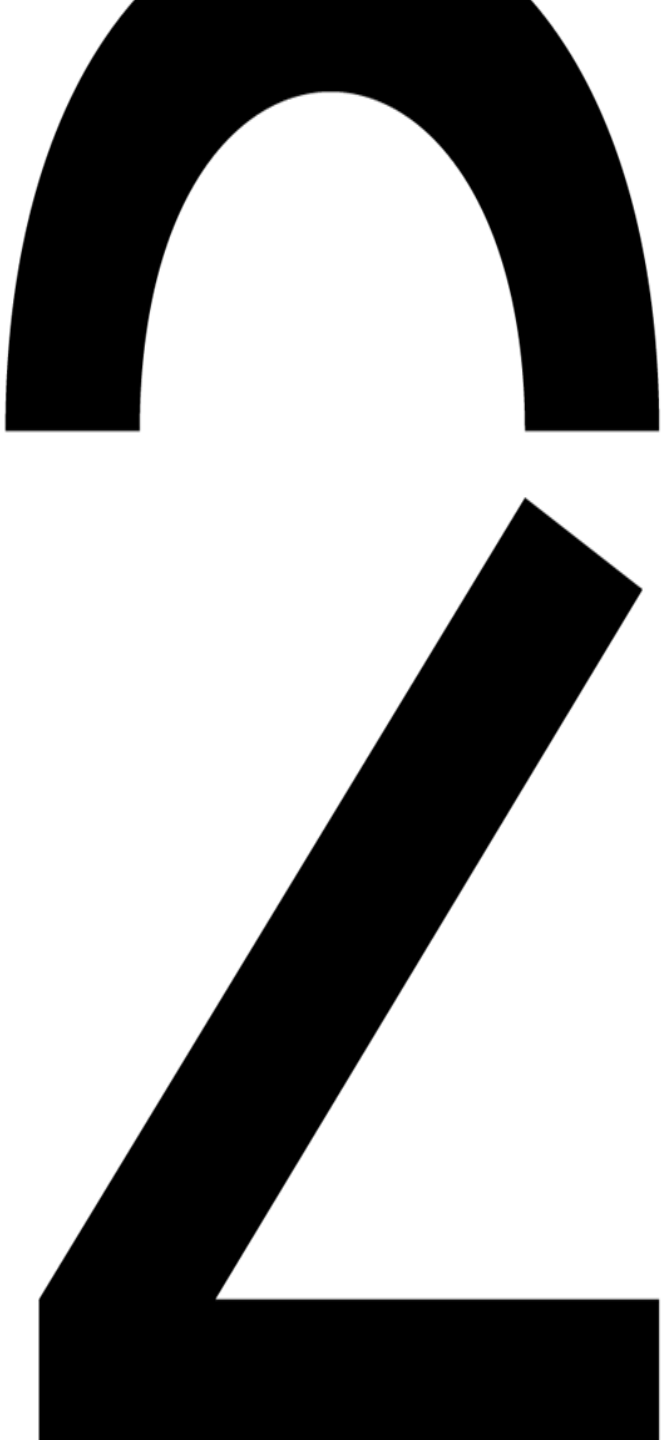
Step 3: Install packages with requirements.txt file



Introduction / Installation

Step 4: Project setup PyCharm



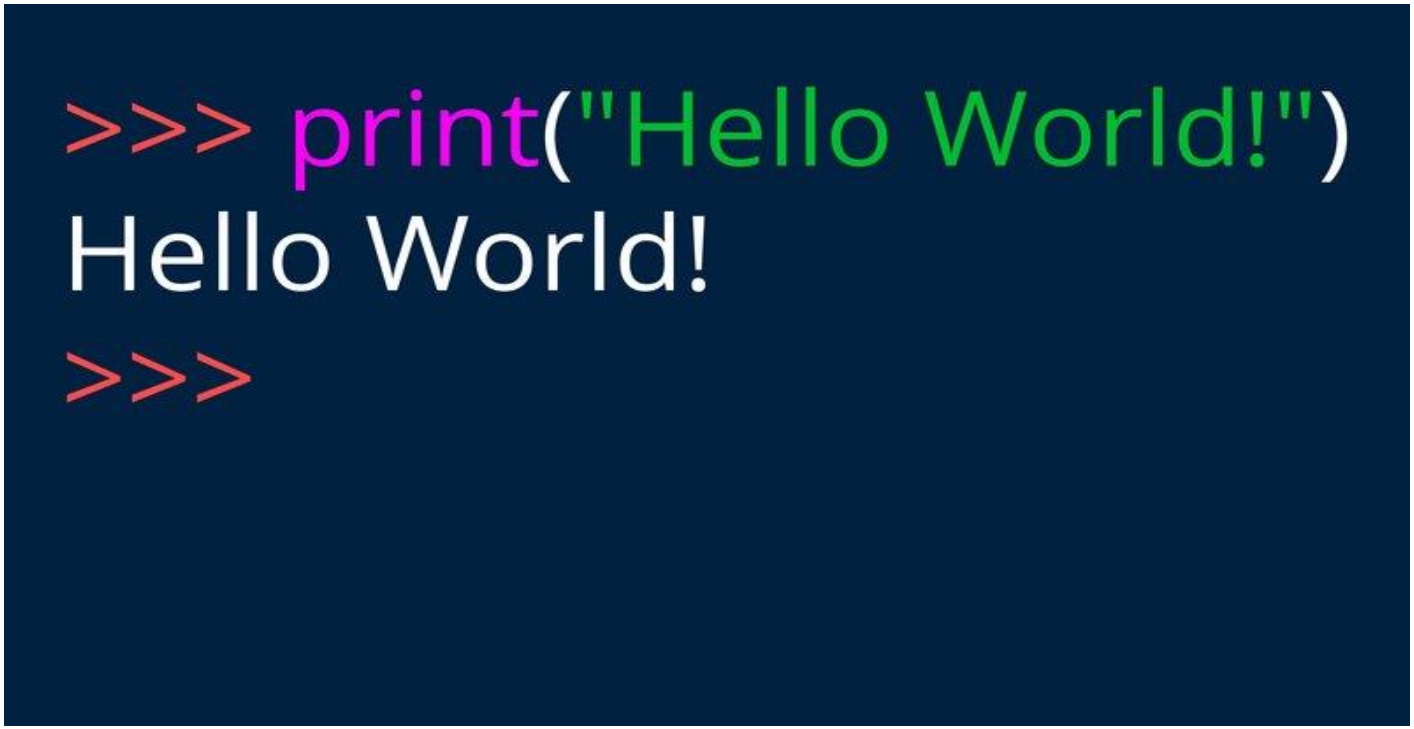


Learning the Python Basics

Introduction / Syntax

- This notebook introduces all the important Python basics
  - Data Science course.

Python commands!!



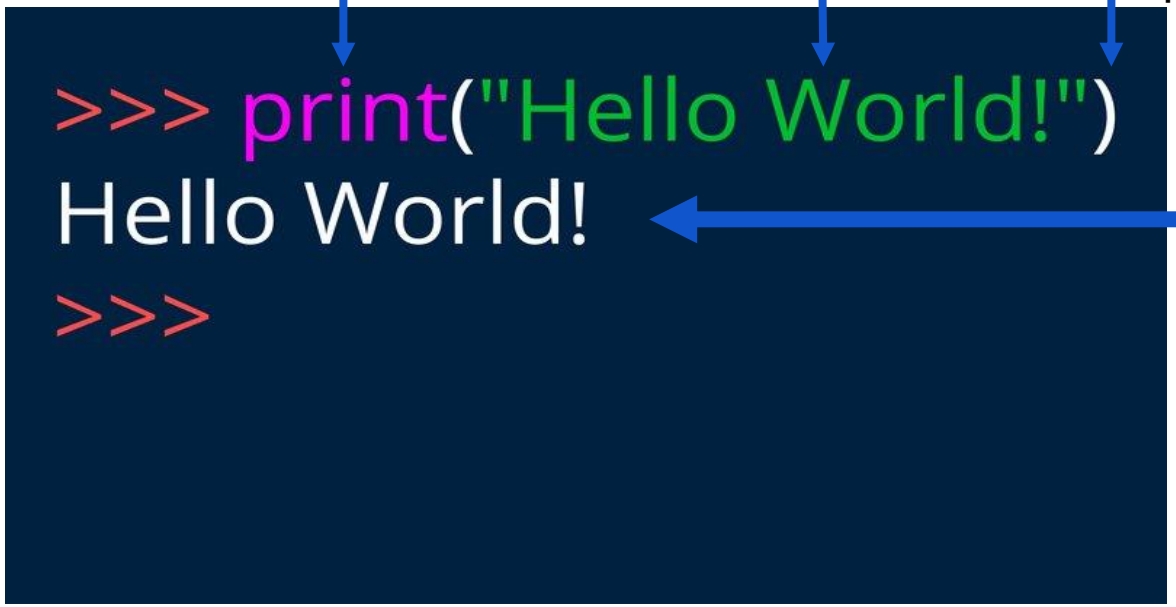
Introduction / Syntax

Python commands!!

Function's "argument"

Python function

Round brackets — "parentheses"



Output

print ≠ **PRINT**

Case sensitive

Variables and data types

2

Variables and data types

1. List elements: A list in Python is an ordered, mutable collection of elements.
    - Lists can contain elements of different types (e.g., integers, strings, ETC).
        - # List of integers
            - numbers = [1, 2, 3, 4, 5]
        - # List of strings
            - fruits = ['apple', 'banana', 'cherry']
        - # Mixed-type list
            - mixed_list = [1, 'apple', 3.5, [2, 3]]
        - # Nested list (list within a list)
            - nested_list = [[1, 2], [3, 4], [5, 6]]
    - Lists are created by placing elements inside square brackets [], separated by **commas**.
2. **Slicing** refers to extracting a subset of elements from a list. In Python, you can slice a list using the syntax:
    - list_name[start:stop:step]

Variables and data types

1. Sets:
   - Unordered: The elements are not stored in any particular order, and their position can change.
   - Unique: A set cannot contain duplicate elements. If you try to add a duplicate, it will be ignored.
   - Mutable: You can add or remove elements from a set.
   - Defined using curly braces {} or the set() function.

1. Dictionary:
   - Key-value pairs: Each key is mapped to a value (similar to a real-world dictionary where each word is associated with a definition).
   - Keys are unique: A dictionary cannot have two identical keys.
   - Keys must be immutable: Keys can be strings, numbers, or tuples, but not lists or other dictionaries.
   - Values can be of any type: Values can be of any data type, including lists, tuples, or even other dictionaries.
   - Defined using curly braces {} or the dict() function.

Variables and data types

```
# int
my_variable = 1

# str
my_variable = "Hello world"

# bool
my_variable = True

# float
my_variable = 18.275

# print formatted string
print(f'The temperature is {my_variable:.1f} degrees')

# type function
type(1.2)
type(3>2)
```

Variables and data types

```
# list
list1 = [0, 1.0, "two"]
list1[1] = 2.0

# tuple, is unchangeable
tuple1 = (0, 1.0, "two")
tuple1[1] = 2.0  # error

# set
set1 = {5, "six"}

# dict
dict1 = {"f":1.0, "s":"six"}
dict1["s"]

# 2-dimensional list
a = [[1, 2, 3],
     [4, 5, 6],
     [7, 8, 9]]
a[0][1] = 10 # 2 replaced by 10, first index = row, second = column
print(a[1])    # [4, 5, 6]
```

Operators

- Assignment

  ```
  =, +=
  ```

- Numerical

  ```
  +, -, *, /, % (modulo) , ** (power of)
  ```

- Comparison

  ```
  <, >, ==, <=, >=, !=
  ```

- Boolean

  ```
  not, and, or
  ```

- Conversion

  ```
  int(3.2), float(2), str(3)
  ```

- String

  ```
  .count('x'), .find('x'), .lower(), .upper(),
  .replace('a', 'b'), .strip()
  ```

- Lists

  ```
  .append(item), .pop(index), .insert(index, item),
  .sort()
  ```

  ```
  len(list1), item in list1
  ```

List comprehension

```
[i for i in range(5)]
# [0, 1, 2, 3, 4]

[x**2 for x in range(1,3)]
# [1, 4]

[ [x,x**2] for x in range(0,3,2)]
# [[0, 0], [2, 4]] (step 2)

[ w[0] for w in ["the", "world"] ]
# ['t', 'w']

[x+y for x in [10,30,50] for y in [2,4]]
#[12, 14, 32, 34, 52, 54]
```

Program flow, functions, objects and modules

Program flow

```
while True:
   print("Looping")      # indentation


for i in range(4,12,2): # 4 -> 12 (not including), step 2
   print(i**2)


if i <0:
   print("negative")
elif i == 0:
   print("zero")
else:
   print("positive")
```

Functions

```
def fac(n):
    if n <= 1:
        return 1
    else:
        return n * fac(n-1)


def power(base, exponent=2):          # optional parameters last
    return base ** exponent


power(2)
power(exponent=1, base=5)
```

Classes and Objects

```
class Person():
    def __init__(self, name):                    # constructor
        self.name = name                         # self: reference to instance
class Student(Person):                           # inheritance
    def __init__(self, name, studentid):
        super().__init__(name)
        self.studentid = studentid
    def welcome(self):                           # method
        print(f"Welcome {self.name}, your id is {self.studentid}")

pers1 = Person("Elise")
pers1.name                          # Elise
stud1 = Student("Nick", 202401)
stud1.name                          # Nick
stud1.welcome()                     # Welcome Nick, your id is 202401
```

Modules

```
# helloworld.py
def hello():
        print("Hello World")


import helloworld
helloworld.hello()


from helloworld import hello
hello()


from helloworld import *
hello()
```

Python code guidelines

4

Python code guidelines

- Indentation:

  4 spaces
- Methods/Functions:

  lowercase_with_undercores
- Variables:

  lowercase_with_undercores
- Constants:

  UPPERCASE
- Private attributes or methods:

  _single_leading_underscore or __double_leading_underscore
- Class:

  CapWords
- Modules:

  lowercase

Notebook time!