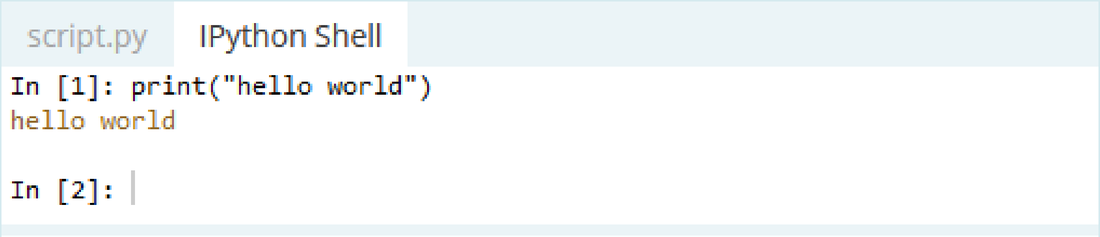
Python

- A simple language with straight-forward syntax
  - Interactive Commands



```
In [1]: print("hello world")
hello world

In [2]:
```

  - Program without Compiling

```
main.py:

    print("hello world")
```

```
shell:
$ python run.py
hello world
```

- Find tutorial online: www.learnpython.org

Python

- Plug-and-play packages for scientific computation
    - NumPy: numerical computation
    - SciPy: collections of numerical algorithm, including machine learning
    - Pandas: providing high-performance, easy-to-use data structures
    - Deep Learning Frameworks: DL with GPUs
    - ……

Installation

- We use Python 3 in the course

- You can find the official release of python (support all platforms)
  - www.python.org
  - For linux users: sudo apt-get install python3

- Or you can install Anaconda (recommended if you want to use GPUs)
  - mirror.tuna.tsinghua.edu.cn/help/anaconda/ (Tsinghua Mirror)

Installation

- Package Manager
  - Automatically install packages and dependencies
  - pip: the default package manager in the official Python
  - conda: a package manager for complex dependencies (like GPU support)

- Virtual Environment
  - Use different packages or python versions across projects
  - virtualenv: venv [name]
  - conda: conda create …

Installation

- IDE
  - Syntax check, highlight
  - Debugger
  - Interactive commands

  - VSCode: code.visualstudio.com
  - PyCharm: www.jetbrains.com/pycharm

NumPy

- High-Performance Numerical Computation
- Vector / Matrix Operations

- Installation
  - pip install numpy
  - or conda install numpy

NumPy

- Vector

```
>>> import numpy as np
>>> a = np.array([1, 2, 3])  #3-dim array
>>> a.shape
(3,)
>>> a + 2 #constant add
array([3, 4, 5])
>>> a * 2 #constant multiplication
array([2, 4, 6])

>>> b = np.array([4, 5, 6])
>>> a + b  #element-wise add
array([5, 7, 9])
>>> a * b  #element-wise multiplication
array([4, 10, 18])
```

- Matrix

```
>>> import numpy as np
>>> a = np.array([[1, 2, 3].
                  [4, 5, 6]])  #2*3 matrix
>>> a.shape
(2,3)
>>> a + 2 #constant add
array([[3, 4, 5], [6, 7, 8]])
>>> a * 2 #constant multiplication
array([[2, 4, 6], [6, 7, 8]])

>>> b = np.array([[4, 5, 6], [7, 8, 9]])
>>> a + b  #element-wise add
array([[5, 7, 9], [11, 13, 15]])
>>> a * b  #element-wise multiplication
array([[4, 10, 18], [28, 40, 54]])
```

NumPy

Advanced Indexing:
https://numpy.org/devdocs/user/quickstart.html#advanced-indexing-and-index-tricks

- Indexing

```
>>> import numpy as np
>>> a = np.array([1, 2, 3, 4])

>>> a[0] # element access
1
>>> a[-1] # backward access
4
>>> a[1:3]  # slicing
array([2, 3])
>>> a[:3]   # slicing from head
array([1, 2, 3])
>>> a[1:]   # slicing to tail
array([2, 3, 4])

>>> a[0] += 1  # in-place add
>>> a
array([2, 2, 3, 4])
```

```
>>> a = np.array([[1, 2, 3].
                  [4, 5, 6]])

>>> a[1, 2]
5
>>> a[1]   # select row
array([4, 5, 6])
>>> a[:, 1]   # select column
array([2, 5])
>>> a[1:2, 0:1]  # different from a[1, 0]
array([[4]])
>>> a[:-1, 1:]
array([[2, 3]])

>>> a[0] *= np.array([3, 2, 1])
        # in-place multiplication
>>> a
array([[3, 4, 3], [4, 5, 6]])
```

NumPy

- Initialization

```
>>> import numpy as np

>>> np.zeros((2,2))
array([[0., 0.], [0., 0.]])
>>> np.ones((2,2))
array([[1., 1.], [1., 1.]])
>>> np.eye(2)
array([[1, 0], [0, 1]])
>>> np.zeros((2,2,3)).shape #3-dim tensor
(2,2,3)
```

- Operation

```
>>> import numpy as np
>>> a = np.ones((3,2))

>>> np.sum(a)
6.0
>>> np.sum(a, axis=0) #sum across rows
array([3., 3.])
>>> np.sum(a, axis=1) #sum across columns
array([2., 2., 2.])

>>>b = np.eye(3)
>>> np.max(b)
1.0
>>> np.max(b, axis=0)
array([1., 1., 1.])
```

NumPy

- Matrix Multiplication

```
>>> import numpy as np

>>> a = np.array([[1, 2], [3, 4], [5, 6]])
>>> b = np.array([[2, 1], [4, 5]])
>>> a @ b # matrix multiplication
array([[10, 11],
       [22, 23],
       [34, 35]])
>>> a @ b.T
……
>>> c = np.array([3, 2])
>>> a @ c  # matrix-vector multiplication
array([ 7, 17, 27])
```

- Broadcasting

```
>>> import numpy as np

>>> a = np.array([[1, 2], [3, 4], [5, 6]])
>>> b = np.array([2, 1])
>>> a + b # broadcast add
array([[3, 3],
       [5, 5],
       [7, 7]])

(3, 2) + (2) ->  (3, 2)
```

broadcast rules:
https://numpy.org/devdocs/user/basics.broadcasting.html

NumPy

- Datatype

```
>>> import numpy as np

>>> x = np.array([1, 2])   # Let numpy choose the datatype
>>> x.dtype
dtype('int32')

>>> x = np.array([1.0, 2.0])   # Let numpy choose the datatype
>>> print(x.dtype)
dtype('float64')

>>> x = np.array([1, 2], dtype=np.int64)   # Force a particular datatype
>>> x.dtype
dtype('int64')
```

NumPy

- Reference in Python

```
>>> def func(x):
...    x += 1

>>> a = 1
>>> func(a)  # basic types are passed by reference
>>> a
1

>>> b = np.array([1])
>>> func(b)  # objects are passed by reference
>>> b
array([2])
```

NumPy

- Reference in Python

```
>>> a = 1
>>> t = a  # basic types are assigned by value
>>> t += 1
>>> a
1

>>> b = np.array([1])
>>> t = a  # objects are assigned by reference
>>> t += 1
>>> a
array([2])
```

More tutorials

- Online Tutorials
  - https://cs231n.github.io/python-numpy-tutorial/
  - https://numpy.org/devdocs/user/quickstart.html