Introduction to Python

Outline

- Introduction to Python
- Jupyter notebook
- Simple value and expressions
- List
- Define a Functions
- List operation: filter map
- List comprehensions
- Multiple-list operations: map, zip
- Strings/List operation
- Data structure: sets, dictionaries
- If/else
- Break/Continue
- For/While Loops

Introduction to Python

Python:one of the most widely used programming languages

- Open Source
- Extensive Support Libraries
- Learning Ease
- User-friendly Data Structures
- Productivity and Speed

c++

```cpp
// 'Hello World!' program

#include <iostream>

int main()
{
  std::cout << "Hello World!" << std::endl;
  return 0;
}
```

java

```java
public class Main {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

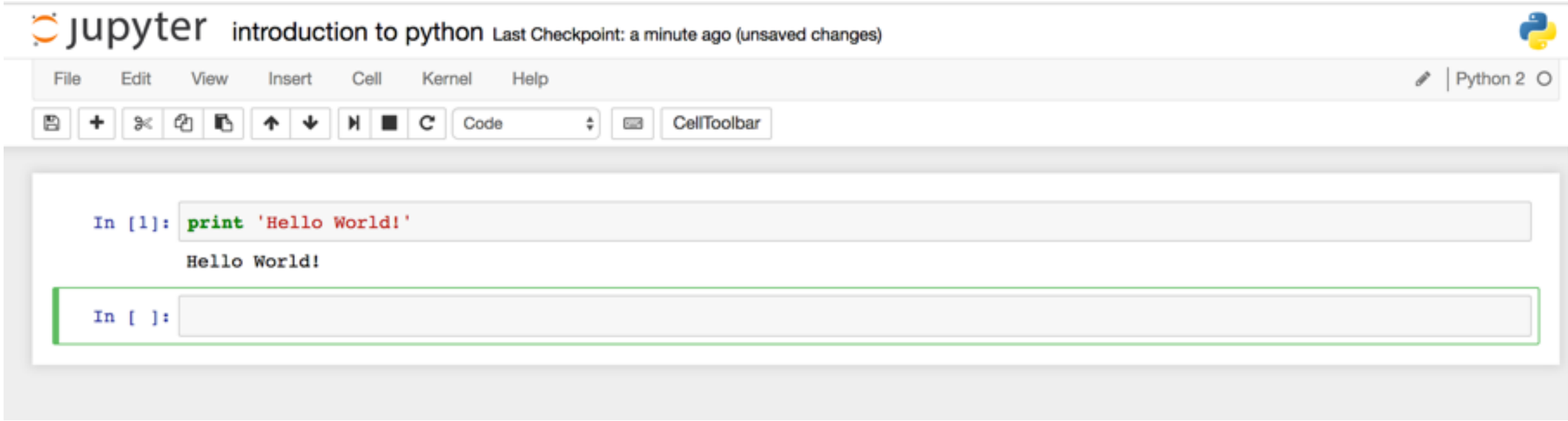
python

```
print “Hello, World!”
```

Jupyter Notebook

An interactive computational environment:

- A powerful interactive shell
- Support for interactive data visualization
- Flexible, embeddable interpreters to load into your own projects
- Easy to use, high performance tools for parallel computing



View all keyboard shortcuts：
Esc+H

Simple value and expressions

Comment: #
Variable assignment: =
Naming conventions: lettter, letter+num, letter_letter/num, can not start with number

Calculator: +(addition),-(subtraction),*(multiplication), /(division),** (power),
// (quotient), %(reminder)

Built-functions and module: import

Defining functions: define a function/lambda

```
def f(x):
    return expression
```

```
f=lambda x: expression
```

Basic data type:

- integer/float
- string
- boolean: T/F

List operations, Conditionals

Lists: ordered collections of num/str

Single list operations and functions:

- create a list: []
- a sequence of integers: range()
- index: starting from 0, ending with -1
- select single element or consecutive elements
- select non consecutive elements
- single list selection/computation: filter/map

list comprehensions

- an elegant way to define and create lists based on existing lists.
- more compact and faster than normal functions and loops for creating list

Multiple list operations:
lists computation/combination:map/zip

Strings/List operations

String built-in functions (immutable):

- str has index, starting with 0
- strip: delete whitespace from the beginning and end
- split: split a long string to a list of short strings
- join: concatenate a list of short strings to a long string, reverse of split
- replace: change a letter
- find: first index/ not found return -1
- lower: change uppercase to lowercase
- upper: change lowercase to uppercase

Mutable functions to a list:

- append: add a value to the end of a list
- remove/del: delete a value
- insert: add a value to a certain position
- sort: sort the list

Data structure

- Lists: ordered collections of num/str
- Tuples: a tuple is an immutable sequence of Python objects, create with ()
- The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.
- Sets:unordered collections without duplicates, create with set()
- The major difference is that sets, unlike lists or tuples, cannot have multiple occurrences of the same element and store unordered values.
- Dictionaries:maps from one value (key) to another value (value)

  Dictionaries:

  - key: string, keys()
  - value: string/num, values()
  - item: a set of key and value, items()

Loops

Conditionals:
if/else loop
if conditions:
  statement1
else:
  statement2

break:The break statement terminates the loop containing it. Control of the program flows to the statement immediately after the body of the loop.

continue: The continue statement is used to skip the rest of the code inside a loop for the current iteration only. Loop does not terminate but continues on with the next iteration.

For loop:
for name in collection:
    statements

While loop:
Initial statement
While condition:
    statements