Basic Python Tutorial

Juxihong Julaiti

Download & Install Python

- Google Anaconda
- Download for python 3.6 version
- Install & Reboot your system
- Open Notebook:
  - Mac: Open a terminal
  - Win: Open anaconda prompt
  - Input: jupyter notebook

Python

- Scripting Language, easy to learn and use
  - Clean, clear syntax and high-level data types

  Python makes difficult things easy: so programmers can focus on overriding algorithms and structures rather than nitty-gritty low level details.
- Object-Oriented
  - Everything is an object (functions, classes)
  - Variables do not need to be defined before using
  - Data type is a property of the object

  You can do things like x=134 and x='I'm a string' without error

  Python will detect the type of variable automatically

  cost: slower than lower level language
- Portable:
  - high end servers and workstations, down to windows CE

Major Uses of Python

web applications

automation

scientific modelling (Operations Research, cplex, gurobi, etc.)

big data applications

……

It's also often used as "glue" code to get other languages and components to play nice.

Python & Scientific Computing

- Large community of users, easy to find help and documentation.
- Extensive ecosystem of scientific libraries and environments
  - – numpy: http://numpy.scipy.org (Numerical Python)
  - – scipy: http://www.scipy.org (Scientific Python)
  - – matplotlib: http://www.matplotlib.org (graphics library)
- Great performance due to close integration with time
  - tested and highly optimized codes written in C and Fortran
- Good support for
  - – Parallel processing with processes and threads
  - – Interprocess communication (MPI)
  - – GPU computing (OpenCL and CUDA)
- Readily available and suitable for use on high-performance computing clusters.
- No license costs, no unnecessary use of research budget.

Outline

- “Hello World”
- Basic Data Types
- Operators
- Basic Data Flow Mechanisms (control flow)
- Compound Data Types
- Conditional Expressions

Hello World!

print('hello world!')

and this is a comment

```
print("Hello world!!")
## and this is a comment
```

```
Hello world!!
```

ipt=input("tell me your name:")

print(ipt)

```
In [1]: ipt=input("tell me your name:")
tell me your name:Jushkhun

In [2]: print(ipt)
Jushkhun
```

Basic Data Types

Naming rules:

Case sensitive: var1 and Var1 are different

Can't start with a number

Name can contain letters, numbers, and underscores

Do not use reserved names (int, float, str, bool, list, dict…..... Find more online)

Var = 'hello world!' # string
Var2 = 99 # integer
var_3 = 0.314 # float
var4=True # boolean

```
Var = 'hello world!'    # string
Var2 = 99               # int
var_3 = 0.314           # float

print(Var,";",Var2,";",var_3)
```

```
hello world! ; 99 ; 0.314
```

Assignment: you don't need to define a variable like JAVA or C

Everything is an object, you can assign the value directly

```
a=b=c=9
print(a,";",b,";",c)
```

```
9 ; 9 ; 9
```

Basic Data Types

Strings (powerful----web crawler, natural language processing, web app)

c = "I love python"

d = 'I love python‘

c and d are same

e = ''' if you need to use " and ' in your string '''

f = ' or like this \ ' '

Object-Oriented

List can be indexed
c[0], c[-1], c[:], c[1:], c[:5],c[1:3]
c.lower()
c.upper()
c.count('o')
c.split()

Basic Data Types

- Check the type of variables: type(x)

```
In [8]: Var = 'hello world!'
        Var2 = 99
        var_3 = 0.314

In [14]: print('the type of Var is :',type(Var))
         print('the type of Var2 is :',type(Var2))
         print('the type of var_3 is :',type(var_3))

         the type of Var is : <class 'str'>
         the type of Var2 is : <class 'int'>
         the type of var_3 is : <class 'float'>
```

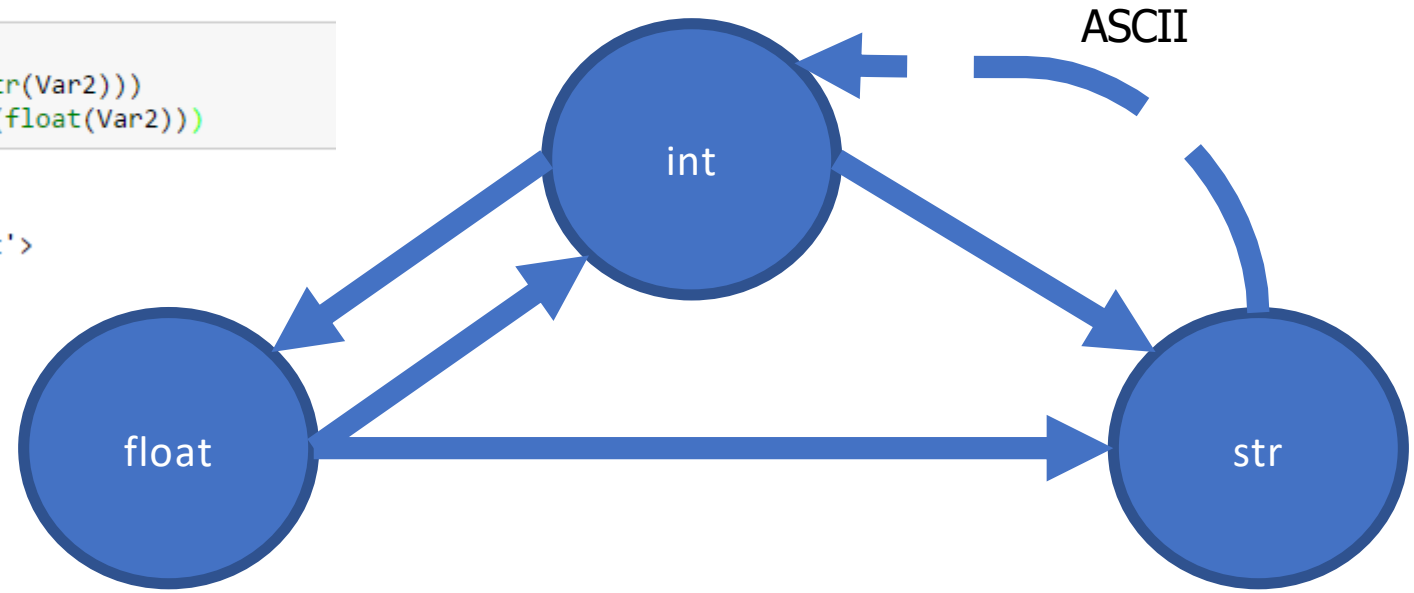
Basic Data Types

- Convert the type of variables
- int(x) str(x) float(x)

```
In [19]: Var2 = 99

In [21]: print('the type of Var2 is :',type(Var2))
         print('the type of str(Var2) is :',type(str(Var2)))
         print('the type of float(Var2) is :',type(float(Var2)))

         the type of Var2 is : <class 'int'>
         the type of str(Var2) is : <class 'str'>
         the type of float(Var2) is : <class 'float'>
```



Operators
Math Operators

- + - * /
- Power: **

```
5**2
```

```
25
```

- Integer division: //

```
5//2
```

```
2
```

+ and * work on strings as well!!

```
"hello"+"world"
```

```
'helloworld'
```

```
"hello"*3
```

```
'hellohellohello'
```

Exercise

- Create a program that asks the user to enter their name and their age.
- Print out a message addressed to them that tells them the year that they will turn 60 years old!

Solution

```
name = input("What is your name: ")
age = int(input("How old are you: "))
year = str((2017 - age)+60)
print(name + " will be 60 years old in the year " + year)
```

1)Add on to the previous program by asking the user for another number and printing out that many copies of the previous message.

2)Print out that many copies of the previous message on separate lines. (*Hint: the string "\n is the same as pressing the ENTER button*)

3)Let user confirm(yes/no) the number of copies if it is greater than 10

Do 2) twice, with/without for statement

Basic data flow mechanisms (control flow)

IF statement

if statement1:
    Action1
elif statement2:
    Action2
elif statement 3:
    Action3
    …….
else:
    Action 3

Remember tab after "**:**"

```
In [49]: pn = input('what\'s your phone number ')
         if pn[:3]==str(814):
             print('you have a Pennsylvania phone number')
         elif pn[:3]==str(201):
             print('you have a New Jersey phone number')
         else:
             print('unknow area code!')

         what's your phone number 8141234567
         you have a Pennsylvania phone number
```

Basic data flow mechanisms (control flow)

FOR Statement

```
for i in range(0,10):
        print(i)
```

```
In [58]: for i in range(0,10):
             print(i)

0
1
2
3
4
5
6
7
8
9
```

```
for i in pn:print(i)
```

```
for i in pn:print(i)

8
1
4
1
2
3
4
5
6
7
```

Basic data flow mechanisms (control flow)

FOR Statement

Searching item ‘o’ in ‘I love python’ and return the index

```
idx=0
for i in 'I love python'.lower():
        if i =='o':
                print('the index is:',idx)
        idx+=1
```

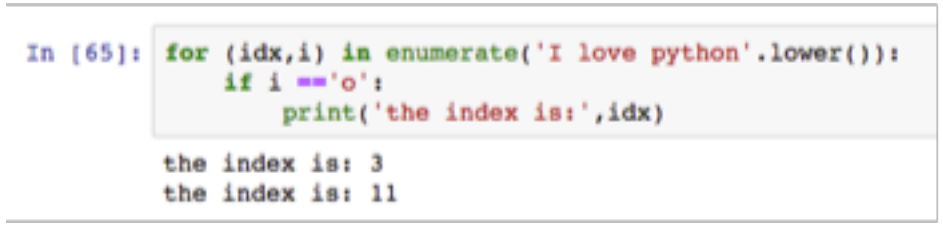
```
idx=0
for i in 'I love python'.lower():
    if i =='o':
        print('the index is:',idx)
    idx+=1
```

```
the index is: 3
the index is: 11
```

Basic data flow mechanisms (control flow)

FOR Statement

```
for (idx,i) in enumerate('I love python'.lower()):
        if i =='o':
                print('the index is:',idx)
```



Exercise

Create a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.

Extras:

1)Add on to the previous program by asking the user for another number and printing out that many copies of the previous message.

2)Print out that many copies of the previous message on separate lines.
(*Hint: the string "\n is the same as pressing the ENTER button*)

3)Let user confirm(yes/no) the number of copies if it is greater than 10

Do 2) twice, with/without for statement

Solution

```
name = input("What is your name: ")
age = int(input("How old are you: "))
num = int(input("How many times should I repeat? "))
if num>10:
        cf = input("Are you sure(yes/no)?")
        if cf =='no':
                num = int(input("How many times should I repeat? "))
year = str((2014 - age)+60)
op=name + " will be 60 years old in the year " +year
print(num*(op))
#print(num*('\n' + op))
for i in range(0,num):print(op)
```

Compound Data Types
Lists

```
a = [2,41,54,4]
print(a)
print(type(a))
```

```
[2, 41, 54, 4]
<class 'list'>
```

- Lists can be heterogeneous

```
b = ['List',6,['sublist',3],"python",'IE']
print(b)
print(type(b))
```

```
['List', 6, ['sublist', 3], 'python', 'IE']
<class 'list'>
```

- List can be indexed

```
b = ['List',6,['sublist',3],"python",'IE']
print(b[0])  # the index start from 0 to the lenght of the list-1
print(b[2])
print(b[-2]) # - mean indexing the list from backward
```

```
List
['sublist', 3]
python
```

Compound Data Types Lists

- Lists can be manipulated

```
print(b)
b[1] = b[1] + 23
print(b)
b[0:2] = [1,12]
print(b)
b[0:2] = []
print(b)
len(b)
```

```
['List', 6, ['sublist', 3], 'python', 'IE']
['List', 29, ['sublist', 3], 'python', 'IE']
[1, 12, ['sublist', 3], 'python', 'IE']
[['sublist', 3], 'python', 'IE']
```

- Add new values to a list
- Sorting list

```
lt = [] # an empty list
print(lt)
```

```
[]
```

```
lt.append(5)
lt.append("hello")
lt.append([1,2,3])
print(lt)
```

```
[5, 'hello', [1, 2, 3]]
```

Compound Data Types
Lists

- a = b # refer
- a = b[:] # copy

- Refer

```
b = ['List',6,"data","python",'IE']
print(b)
b1=b # b1 and b refer each other
print(b1)
b[2]="refered"
print(b1)
b1[0]='haha'
print(b)
```

```
['List', 6, 'data', 'python', 'IE']
['List', 6, 'data', 'python', 'IE']
['List', 6, 'refered', 'python', 'IE']
['haha', 6, 'refered', 'python', 'IE']
```

- Copy

```
b = ['List',6,"data","python",'IE']
print(b)
b1=b[:] # b and b1 will become independent
print(b1)
b[2]="refered"
print("b:",b)
print("b1:",b1)
```

```
['List', 6, 'data', 'python', 'IE']
['List', 6, 'data', 'python', 'IE']
b: ['List', 6, 'refered', 'python', 'IE']
b1: ['List', 6, 'data', 'python', 'IE']
```

Compound Data Types
Tuples

- Tuples can be heterogeneous

```
tu = ('List',6,['sublist', 3],(3,6,'tuple'),"python",'IE')
print(tu)
tu[2:]
```

```
('List', 6, ['sublist', 3], (3, 6, 'tuple'), 'python', 'IE')
```

- But you cannot assign value to tuple

```
# you can assign value to list but tuple
print(tu)
tu[2]='assigned value'
print(tu)
```

```
('List', 6, ['sublist', 3], (3, 6, 'tuple'), 'python', 'IE')

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-88-90aa8d39b7ae> in <module>()
      1 # you can assign value to list but tuple
      2 print(tu)
----> 3 tu[2]='assigned value'
      4 print(tu)

TypeError: 'tuple' object does not support item assignment
```

Compound Data Types
Dictionary

```
dct={'name':"John",'age':42,'kids':[{'name':"Tom",'age':12},{'name':"Jack",'age':18}]}

print(dct['name'])
print(dct['age'])
print(dct['kids'][0]['name'])
print(dct['kids'][1]['age'])
```

```
John
42
Tom
18
```

Adding and removing dictionary entries

```
#adding new entries
dct['job']='programmer'
dct
```

```
{'age': 42,
 'job': 'programmer',
 'kids': [{'age': 12, 'name': 'Tom'}, {'age': 18, 'name': 'Jack'}],
 'name': 'John'}
```

```
# remove dictionary entries
del dct['kids']
dct
```

```
{'age': 42, 'job': 'programmer', 'name': 'John'}
```

Compound Data Types Dictionary

Useful accessor methods

```
dct
```

```
{'age': 42, 'job': 'programmer', 'name': 'John'}
```

```
dct.keys()
```

```
dict_keys(['job', 'name', 'age'])
```

```
dct.values()
```

```
dict_values(['programmer', 'John', 42])
```

```
dct.items()
```

```
dict_items([('job', 'programmer'), ('name', 'John'), ('age', 42)])
```

dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))

Operators
Math Operators

- + - * /
- Power: **

```
5**2
```

```
25
```

- Integer division: //

```
5//2
```

```
2
```

It works on strings and lists as well!!

```
"hello"+"world"
```

```
'helloworld'
```

```
"hello"*3
```

```
'hellohellohello'
```

```
[1,3,4]+[4,6,7,3]
```

```
[1, 3, 4, 4, 6, 7, 3]
```

```
["d",3,"a"]+[4,6,7,3]
```

```
['d', 3, 'a', 4, 6, 7, 3]
```

```
[1,3,4]*3
```

```
[1, 3, 4, 1, 3, 4, 1, 3, 4]
```

Conditional expressions

- One of the most amazing things of Python

```
x = 1 if a0 > a1 else 0
print(x)
```

```
0
```

- It replaced the redundant codes to one simple expression

```
if a0> a1:
    x = 1
else:
    x = 0

print(x)
```

```
0
```

```
A0=dict(zip(('a','b','c','d','e'),(1,2,3,4,5)))
A1 = range(10)
A2 = sorted([i for i in A1 if i in A0])
```

A few key notes

- Everything is an object
- Input is a string by default
- Strings can be indexed
- Operator + and * work on strings
- Operator + works between the same type of variable
- Converting variables is simple:
  int(x)/float(x)/str(x), where x is a variable
- for (idx,x) in enumerate(x): action
- The **TAB** if change the line

Useful links

- Python: https://www.python.org/
- Python 3 Tutorial: https://pythonprogramming.net/introduction-to-python-programming/ (include videos and codes)
- Programming for Everybody (Getting Started with Python):
  - https://www.coursera.org/learn/python