LECTURE 1-2
PYTHON II

Min Sup Hur

Mar. 2 2023

Outline

- Data Types
- Type Conversion
- File I/O of Data

Data Types

- As in other languages, python also distinguishes the data type.

Standard Data Types of Python

Types	Description	Remarks
Numbers	Numeric values	Int, long, float,complex
String	A contiguous set of characters	Enclosed by ‘ ‘ or “ “
List	A list of (whatever types of) values. Use [ ] to create a list.	Heterogeneous lists allowed.
Tuple	Similar to List, but read-only. Use ( ) to create a tuple.	Once created, not updatable.
Dictionary	A list of key-value pairs; Ex. {‘a’:’alpha’, ‘b’:’beta’,…}. Use { } to create one.	‘a’, ‘b’,.. are keys and ‘alpha’,.. are values.

Array defined in numpy module

- Similar to ‘list’ but homogenous (elements should be the same type).
- Behaves like a vector – advantageous in numerical calculation

List

- List is literally the list of elements. To create this, use square brackets [].
- A heterogeneous group of elements can be a list.
- The index of a list starts from 0 (like C).

Example 1. Create a list of integers from 1 to 5, a string 'py' and another list [9,10]. Print the element of index 5 (actually the 6'th element).

Answer: >>> x=[1,2,3,4,5,'py', [9,10]]; print(x[5])

py

>>>

The 0th element

Heterogenous group of elements (integers, strings, another lists).

Example 2. Print all the elements after and before the index 3 of the list created in Ex. 1.

Answer: >>> print(x[3])

[4, 5, 'py', [9,10]] ← Note: the element of index 3 is included.

>>> print(x[:3])

[1, 2, 3] ← Note: the element of index 3 is not included.

List

- Two lists can be merged by + operation.
- A list can be repeated by * operation.

Example 3. Create any two lists. Merge (concatenate) them to the 3rd list.

Answer: >>> x=[1,2,3]; y=['a',3.2,"john"]; z=x+y; print(z)
[1,2,3,'a',3.2,'john']

Example 4. Create a list [1,2,3,1,2,3,1,2,3] by using * operator.

Answer: >>> x=[1,2,3]*3; print(x)
[1,2,3,1,2,3,1,2,3]
>>>

Example 5. Make any list and print the last element.

Answer: >>> x=[-1,0,1,2,3,4,5]; print(x[-1])
5
>>>

Tuples

- Almost the same as the list, except that it's element cannot be overwritten.
- ( ) is used to create one.
- + and * operations work in the exactly same manner as in the list.

Example 6. Create a tuple of two floating point numbers, one list, and one tuple. Print the 2nd element.

Answer:
```
>>> x=(1.5, 3.2, [1,2], (3,4))
>>> print (x[1])
3.2
>>>
```

Example 7. Try to update the 2nd element by a different value. What happens?

Answer:
```
>>> x[1]=0.5
----- Some Error Messages ------
```

String

- A string is contiguous set of characters.
- Enclosed by ‘ ‘ or “ “.
- Each character of a string can be accessed by the same manner as in the tuple. But not updatable.
- + or * operators work in the same manner as in the tuple.

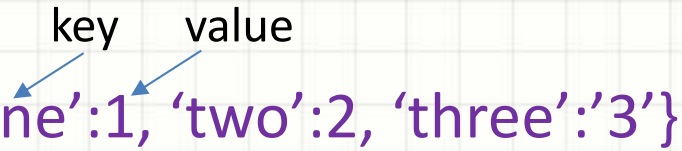
Example 8. Create a string ‘hello, world’. Print every character after ’w’.

Answer: >>> x=‘hello, world’; print(x[7:])

world

Dictionary

- Dictionary is a list of key-value pairs. Use { } to create a Dictionary.
- By accessing the 'key', you get the 'value', as in the dictionary.



Example 9. Create a dictionary of name, date and year. Access the name.

Answer: >>> x={'name':'John', 'date':'1 Jul.', 'year':'2000'}; print(x['name'])
John
>>>

Example 10. Print all the keys and values of the dictionary.

Answer: >>> x.keys()
dict_keys(['name', 'data', 'year'])
>>> x.values()
dict_values(['John', '1 Jul.', '2000'])
>>>

Types of Numeric Data

- The type of a number is determined by its format of expression. See below.

Types of Numbers

Types	Description	Remarks
int	Signed integer. 2, -1, 0, 0xae1, etc. 0x.. is a hexadecimal.	0xa (hexadecimal)=10 (decimal).
long	Long integer; the representable range is doubled. End with 'L'. Ex, 1344083L.	A small l can be used instead of the large L.
float	Floating point real values. 1.5, -20.957, -1.6e-19, 9.11e+31, etc.	'e-19' means $x10^{-19}$.
complex	Complex numbers. 2.4+1.2j, -11+1.3e-31j, etc.	'j' is the imaginary i.

This is obsolete in python 3.7. It works with python 2.7.

Remainder

- Use % symbol to get a remainder of division.

Example 11. Find the remainder of 12053/3.

Answer:
```
>>> print(12053%3)
2
```

Data Type Conversion

- Very frequently you need to convert a string into a number 123.
- Python supports diverse methods of data type conversion.

Example 12. Convert a string '1687' to an integer.

Answer:
```
>>> a='1687';  x=int(a);
>>> print(x+1)
1688
>>>  print(a+1)
------ Error ------
```

Example 13. Convert a string '110' of base 2 to an integer.

Answer:
```
>>> x='110';  y=int(x,2);
>>> y
6
>>>
```

Data Type Conversion

- ***float(x)*** converts x to a floating point number.
- ***list(s)*** converts a string *s* into a list of characters.

Example 14. Convert a string ’12a3’ into a list.

Answer: >>> a=list(’12a3’); print(a)
['1', '2', 'a', '3']
>>>

Example 15. Get two integers. Find the quotient of division of the 1st number by the 2nd number.

Answer: >>> a=int(input(’1st= ‘)); b=int(input(‘2nd=‘)); print(int(a/b))
1st=3
2nd=2
1

Integer truncation = to omit decimals

File I/O

- Many cases, you want to read data from files and save the process results also to files.
- ***open()***, ***read()***, ***write()***, ***close()*** functions provide a way to do that.

Example 16. Read the file 'L1-2-data1' in the Sample Data folder of Blackboard and print it.

Answer:
```
>>> fp=open('L1-2-data1', 'r')   ← Open the file in the reading mode.
>>> s=fp.read()                  ← Read the contents of the file as a string.
>>> fp.close()                   ← Close the file.
>>> print(s)                     ← Look at the read data.
```

Example 17. Get two numbers, and write the product of them, along with the original numbers, to a file 'result'.

Answer:
```
>>> a=input('N1='); b=input('N2='); c=a+', '+b+', '+str(int(a)*int(b))
                                 ↓ Make a string to write.
>>> fp=open('result', 'w')       ← Open the file in the writing mode. New file is created.
>>> fp.write(c)                  ← Write the string c to the open file.
>>> fp.close()
```

File Access Modes

File Access Modes

Access Modes	Description
r	Read only
r+	Read and write
w	Write only
w+	Read and write
a	Append to the end of the file (to new line)
a+	Read and append

Conditional Statements using ‘if’

Example 1. Predict the results of the following if-statements.

Answer:

```
>>> x=0;  y=5
>>> if x < y:
...         print (‘T’)
>>> if x:
...         print (‘T’)
>>> if y:
...         print (‘T’)
```

```
>>> if x or y:
...         print (‘T’)
>>> if x and y:
...         print (‘T’)
>>> if not x:
...         print (‘T’)
```

Example 2. Predict the results of the following if-statements.

Answer:

```
>>> if ‘tho’ in ‘python’:
...         print (‘T’)
>>> if ‘apple’ in [‘grape’, ‘melon’, ‘orange’]:
...         print (‘T’)
```

if, elif, else

Example 3. Get a number and compare its size with 1 and 2.

Answer:
```
>>> x=float(input())     # Enter number here
>>> if x > 2:
. . .        print ('max')
. . . elif x>1:
. . .        print ('intermediate')
. . . else:
. . .        print ('min')
```

‘for’-Loop

Example 4. Using a for-loop, print the letters in ‘Python class’ one by one.

Answer:
```
>>> for let in ‘Python class’:
. . .       print (let)
```
The variable to read the data one by one

Example 5. Using a for-loop, read the numbers in a list and print their squares one by one.

Answer:
```
>>> x=[1,2,3,5,8,13,21]
>>> for var in x:
. . .       print (var**2)
```

Example 6. Using a for-loop, reverse the string, ‘Python class’.

Answer:
```
>>> rs=''
>>> for  letter  in ‘Python class’:
. . .       rs=letter+rs
>>> print (rs)
```

‘while’-Loop

Example 7. Using a while-loop, calculate exp(-2) accurately to four decimal places. You can use the Taylor series; $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$

Answer: >>> trm=1; n=1; v=1

```
>>> while abs(trm) > 1e-4:
. . .       trm *=-2.0/n;   v += trm;  n+=1
>>> print (v)
```

The variable to read the data one by one

Loop Control

- Loop can be controlled by pass, continue, and break.

Example 8. Using nested for-loops, find all the prime numbers between 2 and 100.

Answer:

```
>>> import numpy as np
>>> x=np.arange(2,101)
>>> p =[]
>>> for a in x:                    # Loop 1
...     for b in x:                # Loop 2
...         if a%b==0:
...             break   ← exit out of the current loop, in this case, Loop 2.
...     if b==a:
...         p.append(a)
...
>>> print (p)
```

Loop Control

Example 9. Compare the 'pass' and 'continue' by the following script.

Answer:
```
>>> for element in [-1,0,1]:
...         if element:
...                 pass        ← It does nothing.
...         print(element)    # will print after pass

>>> for element in [-1,0,1]:
...         if element:
...                 continue    ← The statements beyond this are not executed
...         print(element)    # will not print after continue
```

Loop Control

Example 10. Using a for-loop, find the root of $x^2 - x - 1 = 0$ between 0 and 2, accurate up to 3 decimal places.

Hint: Create an array, from 0 to 2 with a gap 0.5×10^{-3} (i.e. 3 decimal places). Sweep this array to find the point where the functional value changes the sign.

Answer:
```
>>> import numpy as np
>>> r=np.arange(0.0, 2.0, 0.5e-3)
>>> for x in r:
. . .         if  x*x – x – 1 >=0 :
. . .                     break
>>> print(x)
```

Reading Numeric Data from a File

- read() reads the contents of the file as a string.
- Use split() function and data conversion to get the numeric data

Example 11. Split the string '1 2 3 4 5 6' by a white space.

Answer: >>> s='1 2 3 4 5 6'; x=s.split(' '); print (x)

['1', '2', '3', '4', '5', '6']

Separator by which you want to split the string. In this case, a white space.

Example 12. Convert the list of characters obtained from the previous example to a list of integers.

Answer:
```
>>> z=[]          ← Create an empty list.
>>> for var in x :
. . .       z.append(int(var))   ← Repeately append integer-converted element to z
. . .
>>> print (z)
[1, 2, 3, 4, 5, 6]
>>>
```

Built-in and User-defined Functions

- Mathematical built-in functions usually allows the lists or arrays as their inputs.
- You can define your own functions by using def keyword.

Example 13. Run the following script to see how the math functions get the lists as inputs and yields another lists as the output.

Answer:
```
>>> import numpy as np
>>> x=[1,2,3,4,5,6,7];  print (np.sin(x))
```

Example 14. Define a function which returns the sum of the elements of a list.

Answer:
```
>>> def mysum(x):
...         sm=0
...         for var in x:
...                 sm += var
...         return sm
>>> x=[1,2,3,4,5];   print(mysum(x))
```

Drawing Graphs of Functions

- 2D plot becomes available by importing *pyplot* module of the *matplotlib* library.

Example 1. Draw graphs of $\sin x$ and $\cos x$ for x from 0 through 4π.

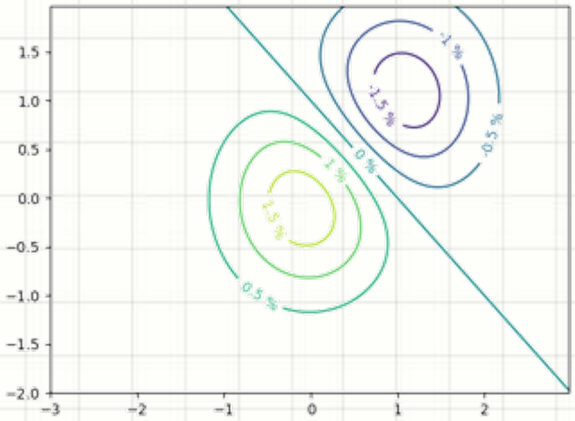
Answer: >>> import numpy as np; import matplotlib.pyplot as plt
>>> x=np.arange(0,4*np.pi,0.1)
>>> plt.plot(x, np.sin(x), x, np.cos(x)); plt.show()

Example 2. Draw graphs of $J_0(x)$ and $Y_0(x)$ (the zero'th order Bessel and Neumann functions) for x from 0 through 30.

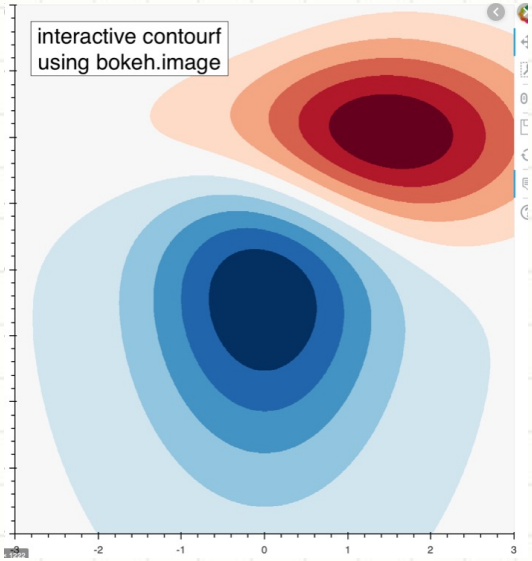
Answer: >>> import scipy.special as sp; import numpy as np;
>>> import matplotlib.pyplot as plt
>>> x=np.arange(0,30,0.1)
>>> plt.plot(x,sp.jv(0,x),x,sp.yv(0,x)); plt.show()

Contour Plot and Color Maps

Data=f (parameter1, parameter2) or $z = f(x, y)$



Contour plot



Color map

Contour Plot of Functions

Example 3. Make a contour plot of $xe^{-x^2-2y^2}$ over the x-y domain $[-3,3]\times[-2,2]$. Mark the contour level for each contour.

Answer:
```
>>> import matplotlib.pyplot as plt; import numpy as np
>>> x=np.arange(-3,3,0.1);  y=np.arange(-2,2,0.1)
>>> X,Y=np.meshgrid(x,y)
>>> Z=X*np.exp(-X**2-2*Y**2)
>>> cs=plt.contour(X, Y, Z, 10, colors='blue')
>>> plt.clabel(cs)
>>> plt.show()
```
Number of levels

Example 4. Make the same contour plot as in Ex. 3, but now fill it with colors. Instead of contour levels, show the colorbar.

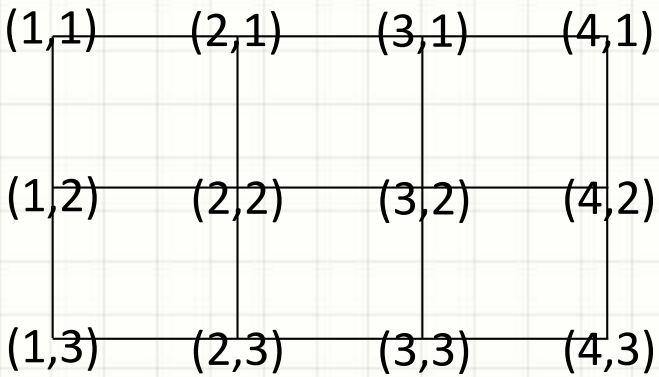
Answer:
```
>>> cs=plt.contourf(X, Y, Z,10)
>>> plt.colorbar(cs)
>>> plt.show()
```
Number of levels

meshgrid() of numpy Package

Example 5. Run the following script.

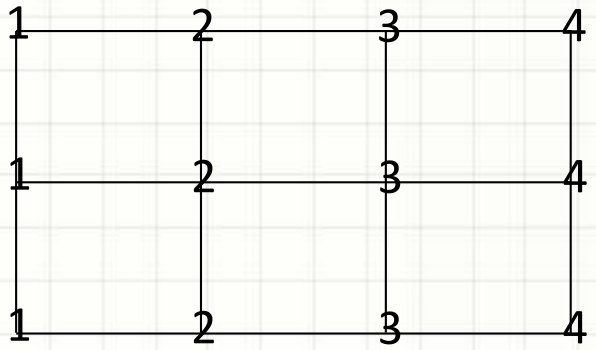
Answer:
```
>>> import numpy as np
>>> x=np.arange(1,5,1);  y=np.arange(1,4,1)
>>> X,Y=np.meshgrid(x,y)
>>> print(X);   print(Y)
```

(X,Y)



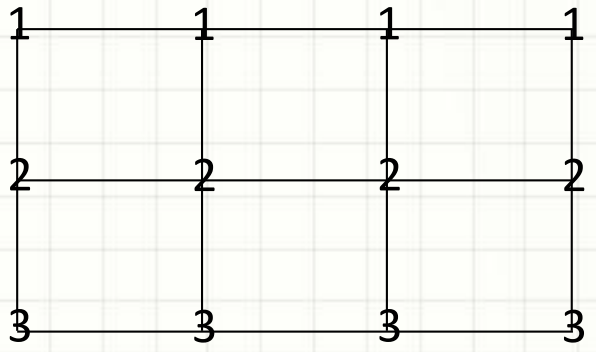
X

2D array of x-coordinates



Y

2D array of y-coordinates



Color Maps

Example 6. Make the same plot as in Ex. 3, but now with the color-map.

Answer:
```
>>> cs=plt.imshow(Z)
>>> plt.colorbar(cs)
>>> plt.show()
```

Example 7. Make the same plot as in Ex. 6, but now with the origin at lower left and the correct x-y axis range.

Answer:
```
>>> cs=plt.imshow(Z, extent=[-3,3,-2,2],origin='lower')
>>> plt.colorbar(cs)
>>> plt.show()
```

Plotting Data

Example 8. Plot the following six data points, (1,1), (2,4), (3,9), (4,16), (5,25), (6,36).

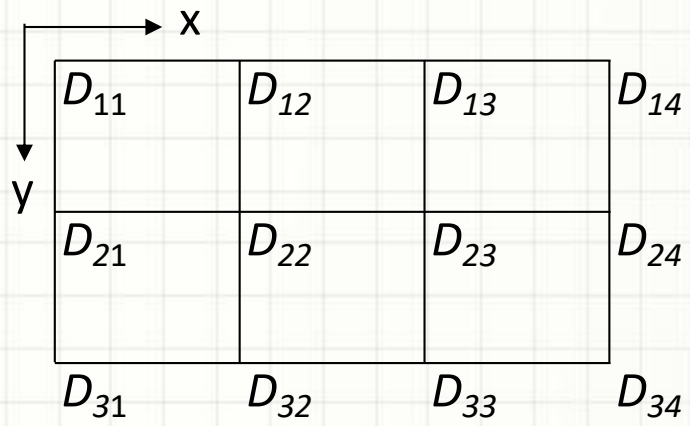
Answer:
```
>>> import matplotlib.pyplot as plt
>>> x=[1,2,3,4,5,6];  y=[1,4,9,16,25,36]
>>> plt.xlabel("x"); plt.ylabel("y")
>>> plt.plot(x,y,'ro'); plt.show()
```

Find details of point and line styles at

https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.plot.html

Contour Plots of Data

Example 9. Data file 'L2-2-data1' has a single long chain of the raster-scanned two-dimensional data, separated by white spaces. The number of meshes in x- and y-directions are 100 and 80, respectively. Make a filled contour plot of this data.



Raster-scanned chain of 2D data

$D_{11}\ D_{12}\ \ldots D_{14}\ D_{21}\ \ldots D_{24}\ D_{31}\ \ldots D_{34}$

Answer:

```
>>> import matplotlib.pyplot as plt
>>> from numpy import *
>>> fp=open("L-2-2-data1", "r");  y=fp.read();  fp.close()
>>> y=y.split(' ');
>>> data=[]
>>> for var in y:
. . .        data.append(float(var))
>>> data=reshape(data,(80, 100))
>>> cs=plt.contourf(data,20); plt.show()
```

Number of raws (hence, y)

Number of columns (hence, x)

End of Lecture 1-2