使用 Python 做数字信号处理实验 FAQ

1. 极简入门代码

自编学习示例，如果 python 环境已配置好，可浏览本小节后再阅读 SciPy 库内容后开始实验。

1.1 交互式编程

```
#完成数学计算
3+5
17/5
17//5
2*8
8**2

#变量的使用
width=2
height=4
width*heigh

#退出 python 解释器
quit()

#启动 python 解释器
python

#换行符
print(“Two \
raws Test”)

#换行符的转义
print(“Two \\ raws Test”)
```

1.1.1 列表(List)的使用

```
#如果列表中存放的是数值，也被称作数组。

squares = [1, 4, 9, 16, 25]
squares
squares[0] # indexing returns the item
squares[3]
squares[-1]
squares[-2]
squares[-3]
len(squares)

#列表中也可以存放其他格式的数据
cells=[1,2,"3string"]
```

```
#列表可以添加
cells.append(2)
cells
cells.append("ttt")
cells
```

【例子，利用两个 list 分别保存城市名称和城市中确诊新型冠状病毒肺炎的人数】
【如果文件中有中文字符，文件的第一行需要如下所示，并将文件保存为 UTF-8 格式】

```
# -*- coding: UTF-8 -*-
city_name = ["武汉", "黄冈", "孝感", "随州", "襄阳", "荆州", "宜昌", "黄石"]
infection_number = [6384, 1422, 1120, 641, 632, 613, 452, 405]

i = 0
i_sum = 0
print("\n 城市：确诊人数\n")
for city in city_name:
    print(city, " : ", infection_number[i])
    i = i + 1
    i_sum = i_sum + infection_number[i]

print("\nTotal affected: ", i_sum)
```

1.1.2 一个简单的程序

写一个生成 菲波那契 子序列的程序，如下所示:

```
>>> # Fibonacci series:
... # the sum of two elements defines the next
... a, b = 0, 1
>>> while b < 10:
...     print(b)
...     a, b = b, a+b
...
1
1
2
3
5
8
```

```
#注意，缩进的地方必须缩进，否则会报语法错误（SyntaxError:）
#一般建议行缩进使用 4 个空格或者 1 个 TAB
```

用一个逗号结尾就可以禁止输出换行:

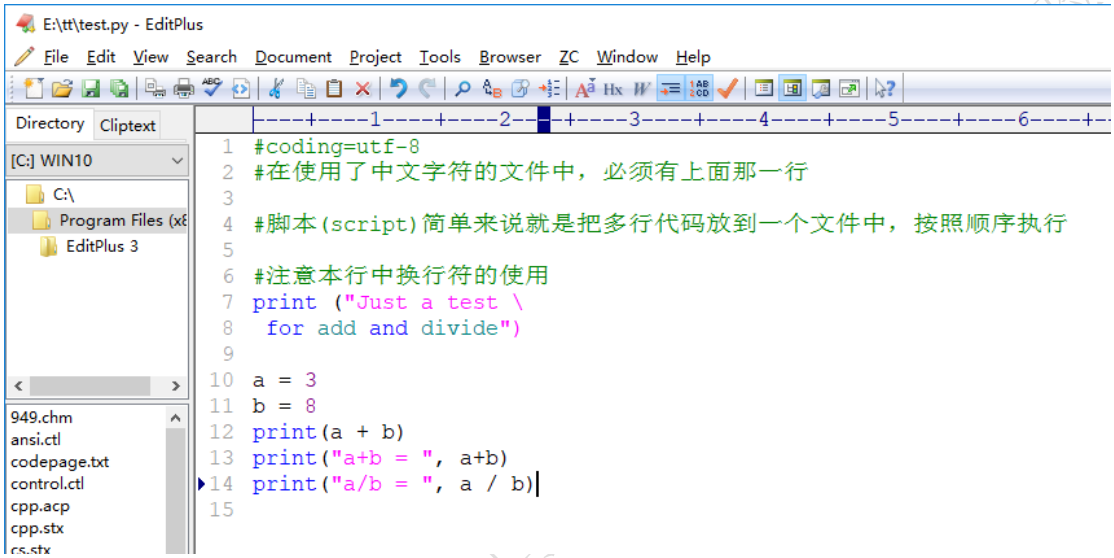
```
>>> a, b = 0, 1
>>> while b < 1000:
...     print(b, end=',')
...     a, b = b, a+b
...
1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,
```

1.2 使用脚本编程

如果你退出 Python 解释器并重新进入，你做的任何定义（变量和方法）都会丢失。因此，如果你想要编写一些更大的程序，为准备解释器输入使用一个文本编辑器会更好，并以那个文件替代作为输入执行。这就是传说中的 脚本。

1.2.1 简单的例子

简单说，把一些语句放置到一个文本文件中。例如，



1.2.2 输入和输出

```
print("Pls input a number")
a = int(input())
#上面语句中，input()表示获取输入的内容,int()表示转换为整数
print("u have input: ", a)
b = 8
print("a+b = ", a+b)
print("a/b = ", a / b)
```

1.3 流程控制

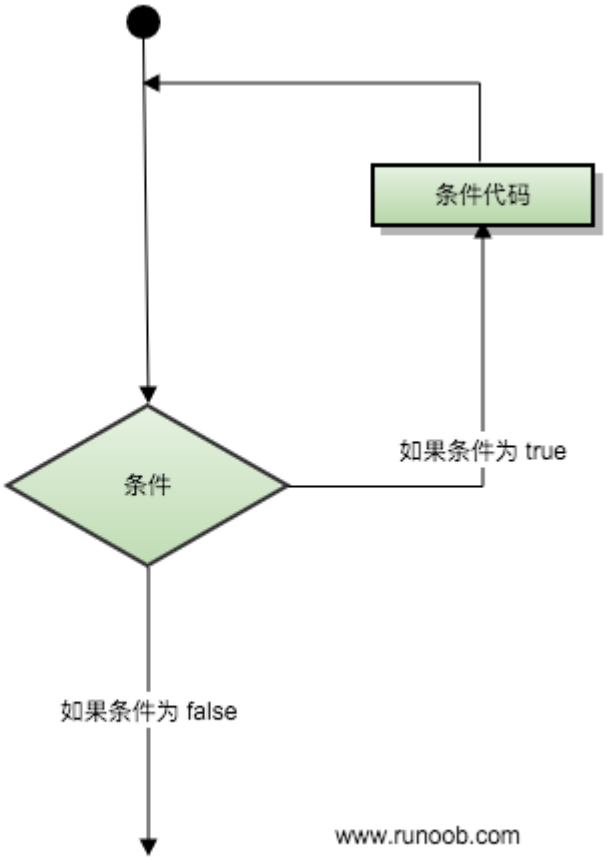
1.3.1 条件语句

```
print("Pls input a number")
a = int(input())
#上面语句中，input()表示获取输入的内容,int()表示转换为整数
if (a == 3) :
    print("Thanks")
```

```
    print("u r the best!")
else:
    print("en heng")
```

参考 https://www.runoob.com/python/python-if-statement.html
写一段代码，如果输入的口令是“abc”，显示“OK”，否则显示“your are denied.”

1.3.2 循环语句



[例：for 循环]

```
for letter in "My Python":
    print("current:",letter)
```

[例：while 循环]

```
i = 0
while (i<=10):
    if i % 3 == 0:
        print("multiple of 3, ",i)
    else:
        print("not multiple of 3,",i)
    i = i + 1
```

写一段代码，允许输入 3 遍口令。
如果输入的口令是“abc”，显示“OK”，否则显示“wrong password, pls input again”

1.4 函数(function)

函数是组织好的，可重复使用的，用来实现单一，或相关联功能的代码段。
你可以定义一个由自己想要功能的函数，以下是简单的规则：

- 函数代码块以 **def** 关键词开头，后接函数标识符名称和圆括号**()**。
- 任何传入参数和自变量必须放在圆括号中间。圆括号之间可以用于定义参数。
- 函数的第一行语句可以选择性地使用文档字符串—用于存放函数说明。
- 函数内容以冒号起始，并且缩进。
- **return [表达式]** 结束函数，选择性地返回一个值给调用方。不带表达式的 return 相当于返回 None。

1.4.1 自定义函数

参考 https://www.runoob.com/python/python-functions.html
例如，

```
def my_add(a,b):
    return a + b

a = 3
b = 5
print(my_add(a,b))
```

1.4.2 Python 内置函数

https://www.runoob.com/python/python-built-in-functions.html

abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reverse()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	exec 内置表达

				式

1.5 模块(module)

随着你的程序变得越来越长，你可能想要将它分割成几个更易于维护的文件。你也可能想在不同的程序中使用顺手的函数，而不是把代码在它们之间中拷来拷去。
为了满足这些需要，Python 提供了一个方法可以从文件中获取定义，在脚本或者解释器的一个交互式实例中使用。这样的文件被称为 模块；模块中的定义可以 导入 到另一个模块或 主模块 中（在脚本执行时可以调用的变量集位于最高级，并且处于计算器模式）。
模块是包括 Python 定义和声明的文件。文件名就是模块名加上 .py 后缀。模块的模块名（做为一个字符串）可以由全局变量 __name__ 得到。例如，你可以用自己惯用的文件编辑器在当前目录下创建一个叫 fibo.py 的文件，录入如下内容:

1.5.1 自定义模块

参考 https://www.runoob.com/manual/pythontutorial3/docs/html/modules.html
写一个模块

```
#my-math.py
def add(a,b):
    return a + b

def mul(a,b):
    return a * b

def div(a,b):
    return a / b

def mod(a,b):
    return a % b
```

编写另外一个脚本文件，调用上面模块中定义的函数

```
#my_math.py 是自己定义的 module，里面写好了一些函数
import my_math

print("pls input 1st number")
i = int(input())
print("pls input 2nd number")
j = int(input())

print(my_math.add(i,j))
print(my_math.mod(i,j))
```

写一段代码，输入一个整数，计算其阶乘的结果并输出。

1.5.1.1 Import 和 From…import 的区别

https://jingyan.baidu.com/article/15622f242e15b6fdfcbea5b5.html

1.5.2 使用 Python 标准库

【例，标准库 time】
```
import time;  # 引入 time 模块
ticks = time.time()
print "当前时间戳为:", ticks
```

【例，标准库 os】
```
import os
os.getcwd()          # 返回当前的工作目录
os.chdir('/server/accesslogs')    # 修改当前的工作目录
os.system('mkdir today')    # 执行系统命令 mkdir
```

【例，标准库 math】
```
import math
math.cos(math.pi / 4)
math.log(1024, 2)
```

Python 标准库提供了非常多的功能，详情可参阅
https://www.runoob.com/python3/python3-stdlib.html
https://www.cnblogs.com/yfacesclub/p/10782601.html

1.5.3 使用第三方的模块

1.5.3.1 播放声音

https://pythonbasics.org/python-play-sound/

##方法 1## snack sound kit
```
from playsound import playsound
playsound('audio.mp3')
```

##方法 3## snack sound kit
```
from Tkinter import *
import tkSnack

root = Tk()
tkSnack.initializeSnack(root)

snd = tkSnack.Sound()
snd.read('sound.wav')
snd.play(blocking=1)
```

##方法 4## native player
```
import os
file = "SOS.mp3"
os.system("mpg123 " + file)
```

1.5.3.2 基于 scikit-mage 包把彩色图转换成灰度图

【例子 1】
https://scikit-image.org/docs/dev/auto_examples/color_exposure/plot_rgb_to_gray.html#sphx-glr-auto-examples-color-exposure-plot-rgb-to-gray-py
【所有的例子】
https://scikit-image.org/docs/dev/auto_examples/

1.6 最简单的算法：冒泡排序

算法原理参考
https://www.cnblogs.com/SteveWesley/p/10007987.html

```
def bubble_sort(nums):
    for i in range(len(nums) - 1):
        for j in range(len(nums) - i - 1):
            if nums[j] > nums[j + 1]:
                nums[j], nums[j + 1] = nums[j + 1], nums[j]
    return nums
```

下面做题，

```
# -*- coding: UTF-8 -*-
city_name = ["武汉", "孝感", "襄阳", "荆州", "黄冈",\
"随州", "黄石"， "宜昌", "咸宁", "荆门",\
"鄂州","十堰","仙桃","恩施州","天门",\
"潜江", "神农架林区"]

infection_number = [8351, 1462, 735, 713, 1645,\
 706, 509, 496 , 384, 422,\
 382, 318, 225, 138, 128,\
54, 10,]
```

写 python 脚本，按照确诊人数递增（或者递减）顺序排序城市。
也就是说，把 city_name 和 infection_number 两个列表重新排序。

2. Python 的开发环境

使用 Python 解释器运行自己编写的 python 脚本文件，每次采用命令行的方式比较麻烦。所以有人开发图形化的集成环境 IDE，在集成了多种功能的图形界面软件中，可以十分方便地完成脚本的编写、运行等多个步骤。

2.1 集成开发环境（IDE: Integrated Development Environment）

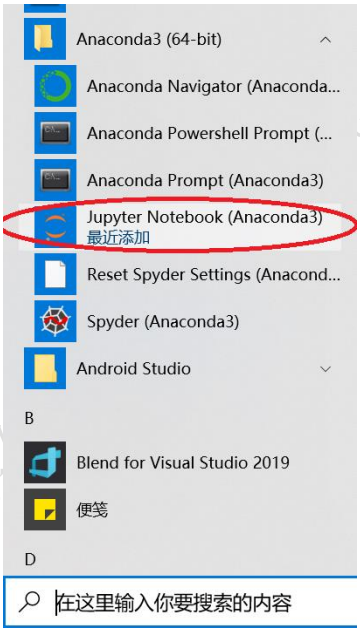
PyCharm 教育版

https://www.jetbrains.com/education/download/#section=pycharm-edu

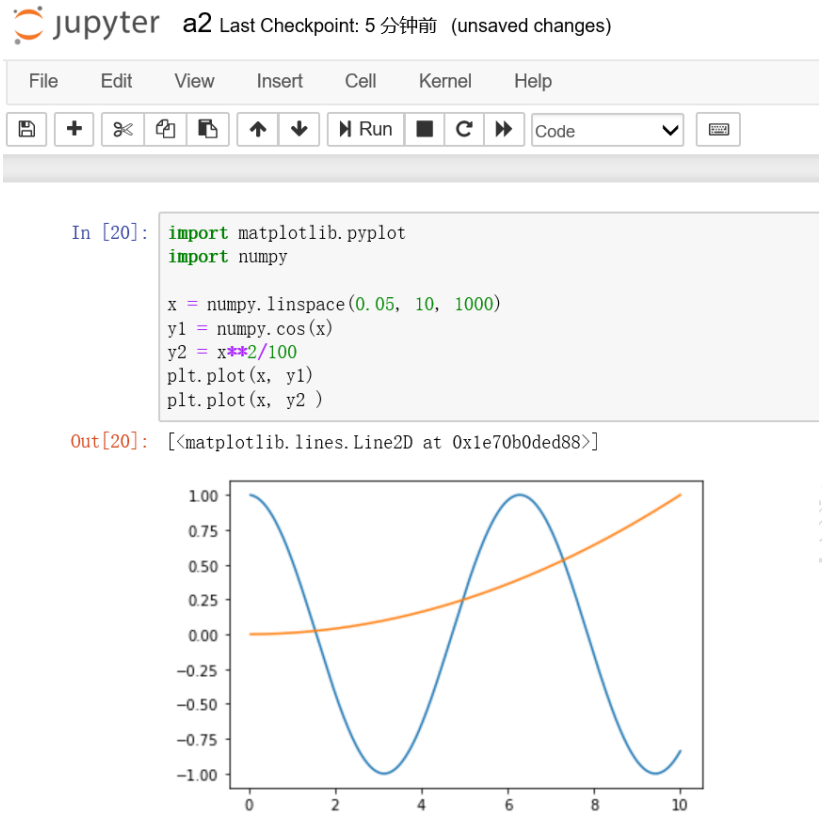
- 打开已有的 Python 文件，执行。
- 新建 Python 文件，执行。
- Python 文件可以调试（Debug）、单步（Step）执行等。

2.2 使用 Jupyter Notebook

Jupyter Notebook 的本质是一个 Web 应用程序，便于创建和共享文学化程序文档，支持实时代码，数学方程，可视化和 markdown。用途包括：数据清理和转换，数值模拟，统计建模，机器学习等等。Jupyter Notebook 这一名称实际上是分为两部分，前半部分是结合了木星 Juipter 一词，并对 Julia（新兴语言）、Python、R 这三门编程语言的参考组合而来。



使用 Jupyter Notebook，可以把代码和代码执行的结果优雅地结合在一起。




```
import matplotlib.pyplot
import numpy

x=numpy.linspace(0.05,10,1000)
y1=numpy.cos(x)
y2=x**2/100
plt.plot(x,y1)
plt.plot(x,y2)
```

2.2.1 Jupyter Notebook 有两种不同的键盘输入模式

Jupyter 笔记本有两种不同的键盘输入模式. 编辑模式允许您将代码或文本输入到一个单元格中，并通过一个绿色的单元格来表示 命令模式将键盘与笔记本级命令绑定在一起，并通过一个灰色的单元格边界显示，该边框为蓝色的左边框。

Jupyter 的使用有很多快捷键。
例如，按下 ESC 键之后，按 H，可以得到快捷键的帮助信息如**错误!未找到引用源。**所示。

2.2.2 如何执行 Shell 命令?

链接：https://www.zhihu.com/question/266988943/answer/632279672

Shell 是一种与计算机进行文本交互的方式。
一般来讲，当你正在使用 Python 编译器，需要用到命令行工具的时候，要在 shell 和 IDLE 之间进行切换。
但是，如果你用的是 Jupyter，就完全不用这么麻烦了，你可以直接在命令之前放一个“!”，就能执行 shell 命令，完全不用切换来切换去，就能在 IPython 里执行任何命令行。

例如，列出当前目录下的所有文件
!dir

2.2.3 nbextensions

链接：https://www.zhihu.com/question/266988943

笔记本扩展（nbextensions）是一种 JavaScript 模块，可以加载到笔记本前端页面上，可以大

大提升用户体验。
比如下面这些扩展工具，简直能让效率提升 10000 倍。
Hinterland
Hinterland 功能可以让你每敲完一个键，就出现下拉菜单，可以直接选中你需要的词汇。

2.2.4 如何打开*.ipynb 文件？

原文链接：https://blog.csdn.net/qq_16633405/article/details/80198648

.ipynb 文件的三种打开方式：
1，GitHub 中可以直接打开 .ipynb 文件。
2，可以把 .ipynb 文件对应的下载链接复制到 https://nbviewer.jupyter.org/ 中查看。
3，安装 Anaconda，从开始菜单中打开 jupyter notebook 的快捷方式（prompt 中用该命令打开同理），默认启动路径在 C:\Users\yourname 类似的文件夹。把 .ipynb 文件复制到这个目录下面，找到并打开即可查看。

【例如】
存放在 http://staff.ustc.edu.cn/~cxh/jupyter/img_basic.ipynb
访问 https://nbviewer.jupyter.org/，输入上述 URL，则跳转到，
https://nbviewer.jupyter.org/url/staff.ustc.edu.cn/~cxh/jupyter/img_basic.ipynb
可以看到对应的 jupyter notebook 的结果已经显示出来了。

3. Python 中的科学计算库

Python 在科学计算领域有三个非常受欢迎库，numpy、SciPy、matplotlib。

- NumPy（Numerical Python）是 Python 的一种开源的数值计算扩展。这种工具可用来存储和处理大型矩阵，比 Python 自身的嵌套列表（nested list structure)结构要高效的多（该结构也可以用来表示矩阵（matrix)），支持大量的维度数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。教程可参考[1]或中文资料[2]。
- SciPy 构建在 numpy 的基础之上，它提供了许多的操作 numpy 的数组的函数。SciPy 是一款方便、易于使用、专为科学和工程设计的 python 工具包，它包括了统计、优化、整合以及线性代数模块、傅里叶变换、信号和图像图例，常微分方差的求解等，SciPy 完整的教程参见[3]。数字信号处理算法会设计到 SciPy 库的多个模块，如傅立叶变换[4]、信号处理/滤波器设计[5] 有关的模块。在实验前建议浏览目录。
- Matplotlib 则是 Python 的绘图库。它可与 NumPy 一起使用，提供了一种有效的 MatLab 开源替代方案[6]，中文资料可阅读[7]。

3.1 NumPy

入门参考中文资料[2]即可。

3.2 SciPy

SciPy (pronounced "Sigh Pie") 是一个基于 Python 的开源[8]软件生态系统，面向数学、科学、工程等领域，可以处理插值、积分、优化、图像处理、常微分方程数值解的求解、信号处理等问题。其核心包如图 1 所示。

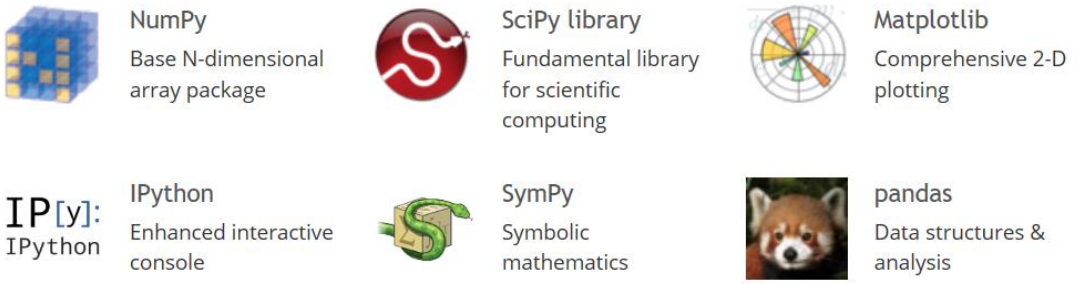


图 1 SciPy 科学计算库中的包[9]

SciPy 库建立在 Numpy 库之上，提供了大量科学算法，主要包括这些主题：

模块名	应用领域
scipy.cluster	向量计算/Kmeans
scipy.constants	物理和数学常量
scipy.fftpack	傅立叶变换[4]
scipy.integrate	积分程序
scipy.interpolate	插值
scipy.io	数据输入输出
scipy.linalg	线性代数程序
scipy.ndimage	n 维图像包
scipy.odr	正交距离回归

scipy.optimize	优化
scipy.signal	信号处理[5]
scipy.sparse	稀疏矩阵
scipy.spatial	空间数据结构和算法
scipy.special	一些特殊的数学函数，如超越函数
scipy.stats	统计

3.2.1 scipy.signal

依据官网[5]，scipy.signal 包含如下模块

子模块名称	功能描述
Convolution	
B-splines	
Filtering	
Filter design	
Matlab-style IIR filter design	
Continuous-Time Linear Systems	
Discrete-Time Linear Systems	
LTI Representations	
Waveforms	
Window functions	
Wavelets	
Peak finding	
Spectral Analysis	

3.2.2 scipy.fftpack.fft

https://docs.scipy.org/doc/scipy/reference/generated/scipy.fftpack.fft.html#scipy.fftpack.fft

scipy.fftpack.fft(*x, n=None, axis=-1, overwrite_x=False*)[source]¶

Return discrete Fourier transform of real or complex sequence.

The returned complex array contains `y(0), y(1),..., y(n-1)` where

`y(j) = (x * exp(-2*pi*sqrt(-1)*j*np.arange(n)/n)).sum()`.

Parameters

***x*array_like**

Array to Fourier transform.

n*int, optional*

Length of the Fourier transform. If `n < x.shape[axis]`, *x* is truncated. If `n > x.shape[axis]`, *x* is zero-padded. The default results in `n = x.shape[axis]`.

axis*int, optional*

Axis along which the fft's are computed; the default is over the last axis (i.e., `axis=-1`).

overwrite_x*bool, optional*

If True, the contents of *x* can be destroyed; the default is False.

Returns

***z*complex ndarray**

with the elements:

```
[y(0),y(1),..,y(n/2),y(1-n/2),...,y(-1)]        if n is even
[y(0),y(1),..,y((n-1)/2),y(-(n-1)/2),...,y(-1)]  if n is odd
```

where:

```
y(j) = sum[k=0..n-1] x[k] * exp(-sqrt(-1)*j*k* 2*pi/n), j = 0..n-1
```

Notes

The packing of the result is “standard”: If `A = fft(a, n)`, then `A[0]` contains the zero-frequency term, `A[1:n/2]` contains the positive-frequency terms, and `A[n/2:]` contains the negative-frequency terms, in order of decreasingly negative frequency. So for an 8-point transform, the frequencies of the result are [0, 1, 2, 3, -4, -3, -2, -1]. To rearrange the fft output so that the zero-frequency component is centered, like [-4, -3, -2, -1, 0, 1, 2, 3], use **`fftshift`**.

Both single and double precision routines are implemented. Half precision inputs will be converted to single precision. Non floating-point inputs will be converted to double precision. Long-double precision inputs are not supported.

This function is most efficient when *n* is a power of two, and least efficient when *n* is prime.

Note that if `x` is real-valued then `A[j] == A[n-j].conjugate()`. If `x` is real-valued and `n` is even then `A[n/2]` is real.

If the data type of *x* is real, a “real FFT” algorithm is automatically used, which roughly halves the computation time. To increase efficiency a little further, use **`rfft`**, which does the same calculation, but only outputs half of the symmetrical spectrum. If the data is both real and symmetrical, the **`dct`** can again double the efficiency, by generating half of the spectrum from half of the signal.

Examples

```
>>>
>>> from scipy.fftpack import fft, ifft
>>> x = np.arange(5)
>>> np.allclose(fft(ifft(x)), x, atol=1e-15)    # within numerical accuracy.
True
```

3.3 matplotlib

入门参考中文资料[7]即可。

4. 附录：电子书籍

4.1 Ebooks

文献[10]
https://learnku.com/docs/byte-of-python/2018
Python 简明教程 2018
《A Byte of Python》的中文译本，由社区维护，每年更新

文献[11]
Python 入门指南(Release: 2.7.13)
https://www.runoob.com/manual/pythontutorial/docs/html/

文献[12]
Python 入门指南(Release: 3.6.3)
https://www.runoob.com/manual/pythontutorial3/docs/html/

4.2 Python 和其他编程语言

2018 年，python 编程语言荣获“年度编程语言”称号！近 20 年来，C、C++和 Java 一直排在前 3 位，远远领先于其他语言。python 加入了这三种语言。它是当今大学最常教授的第一语言，在统计领域排名第一，在人工智能编程领域排名第一，在脚本编写方面排名第一，在系统测试方面排名第一。

Jan 2019	Jan 2018	Change	Programming Language	Ratings	Change
1	1		Java	16.904%	+2.69%
2	2		C	13.337%	+2.30%
3	4	↑	Python	8.294%	+3.62%
4	3	↓	C++	8.158%	+2.55%
5	7	↑	Visual Basic .NET	6.459%	+3.20%
6	6		JavaScript	3.302%	-0.16%
7	5	↓	C#	3.284%	-0.47%
8	9	↑	PHP	2.680%	+0.15%
9	-	⇈	SQL	2.277%	+2.28%
10	16	⇈	Objective-C	1.781%	-0.08%
11	18	⇈	MATLAB	1.502%	-0.15%
12	8	⇊	R	1.331%	-1.22%
13	10	↓	Perl	1.225%	-1.19%
14	15	↑	Assembly language	1.196%	-0.86%
15	12	↓	Swift	1.187%	-1.19%
16	19	↑	Go	1.115%	-0.45%
17	13	⇊	Delphi/Object Pascal	1.100%	-1.28%
18	11	⇊	Ruby	1.097%	-1.31%
19	20	↑	PL/SQL	1.074%	-0.35%
20	14	⇊	Visual Basic	1.029%	-1.28%

图 2 TIOBE 的世界编程语言排行榜-2019 年 1 月

5. 参考文献

[1] https://numpy.org/. *NumPy: the absolute basics for beginners*. Available: https://numpy.org/doc/stable/user/absolute_beginners.html

[2] 菜鸟教程 runoob.com. *NumPy 教程*. Available: https://www.runoob.com/numpy/numpy-tutorial.html

[3] S. developers. (2019, 2019-12-14). *SciPy Tutorial & Developer's Guide & API Reference*. Available: https://docs.scipy.org/doc/scipy/reference/index.html

[4] S. developers. (2019, 2019-12-14). *Discrete Fourier transforms (scipy.fftpack)*. Available: https://docs.scipy.org/doc/scipy/reference/fftpack.html

[5] S. developers. (2019, 2019-12-14). *Signal processing (scipy.signal)*. Available: https://docs.scipy.org/doc/scipy/reference/signal.html

[6] https://matplotlib.org/, *Matplotlib User's Guide*, 2020.

[7] 菜鸟教程 runoob.com. *Matplotlib 教程*. Available: https://www.runoob.com/w3cnote/matplotlib-tutorial.html

[8] S. developers. (2019, 2019-12-14). *SciPy Source Code*. Available: https://github.com/scipy/scipy

[9] S. developers. (2019, 2019-12-14). *SciPy Homepage*. Available: https://www.scipy.org/

[10] LearnKu. *《A Byte of Python》的中文译本*. Available: https://learnku.com/docs/byte-of-python/2018

[11] 菜鸟教程 runoob.com. (2017). *Python 入门指南 (Release: 2.7.13)*. Available: https://www.runoob.com/manual/pythontutorial/docs/html/

[12] 菜鸟教程 runoob.com. (2017). *Python 入门指南 (Release: 3.6.3)*. Available: https://www.runoob.com/manual/pythontutorial3/docs/html/