

# PYTHON STARTUP

1

# 时局图

- 网站开发
  - 游戏制作
  - 自动处理
  - 数据分析
  - 科学研究
  - 人工智能
  - .....
- 2014年美国拨款推进“Code for All”计划
  - 2015年进入天津初中信息技术教材
  - 2017年进入浙江高中信息技术教材并纳入2018高考内容
  - 2018年进入山东小学信息技术教材
  - 2018年进入天津高中信息技术教材
  - .....

# 语言特点



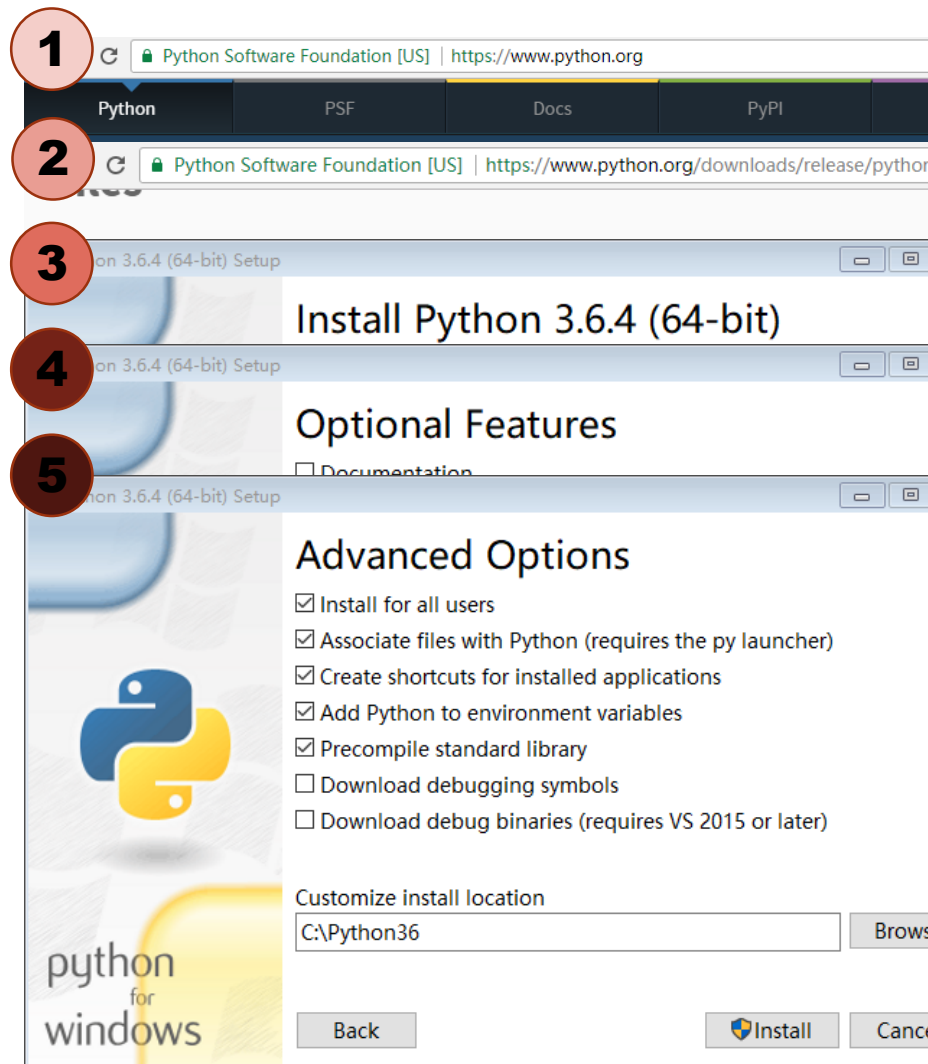
人生苦短  
我用Python

# PYTHON 环境安装

- Windows通过安装程序进行安装
- 其他操作系统自带Python环境

# WINDOWS 下 安 装 PYTHON

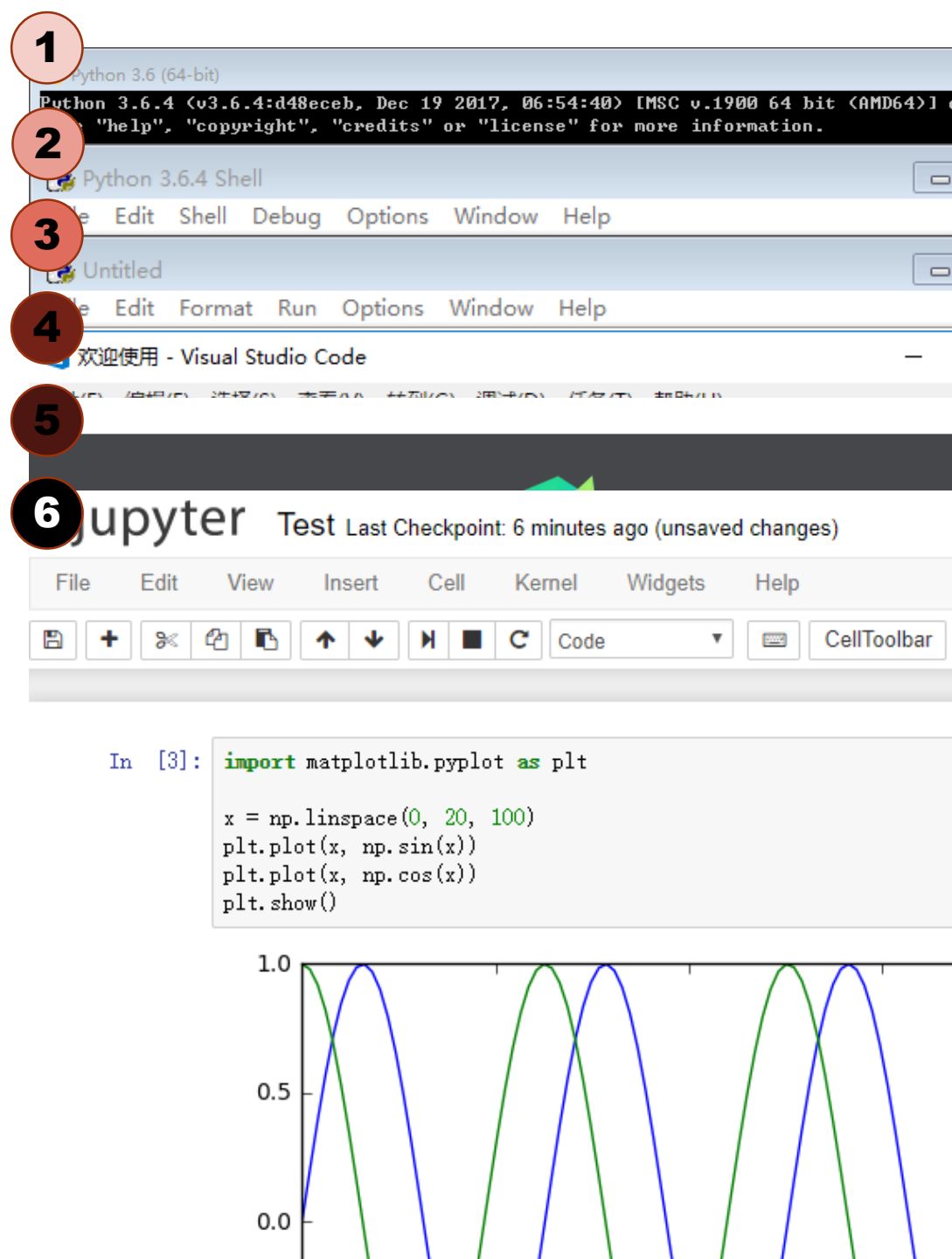
- 访问[www.python.org](https://www.python.org)，点击“Downloads” 区块中的“Python 3.x.x”
- 滚动到最下方，找到“Windows x86-64 executable installer”，点击下载
- 建议选择“Customize installation”，可以自行设置安装信息。
- “pip”和“tcl/tk and IDLE”必选，其他两个基本没有用处。
- 高级选项如图，安装位置建议放到C盘根目录





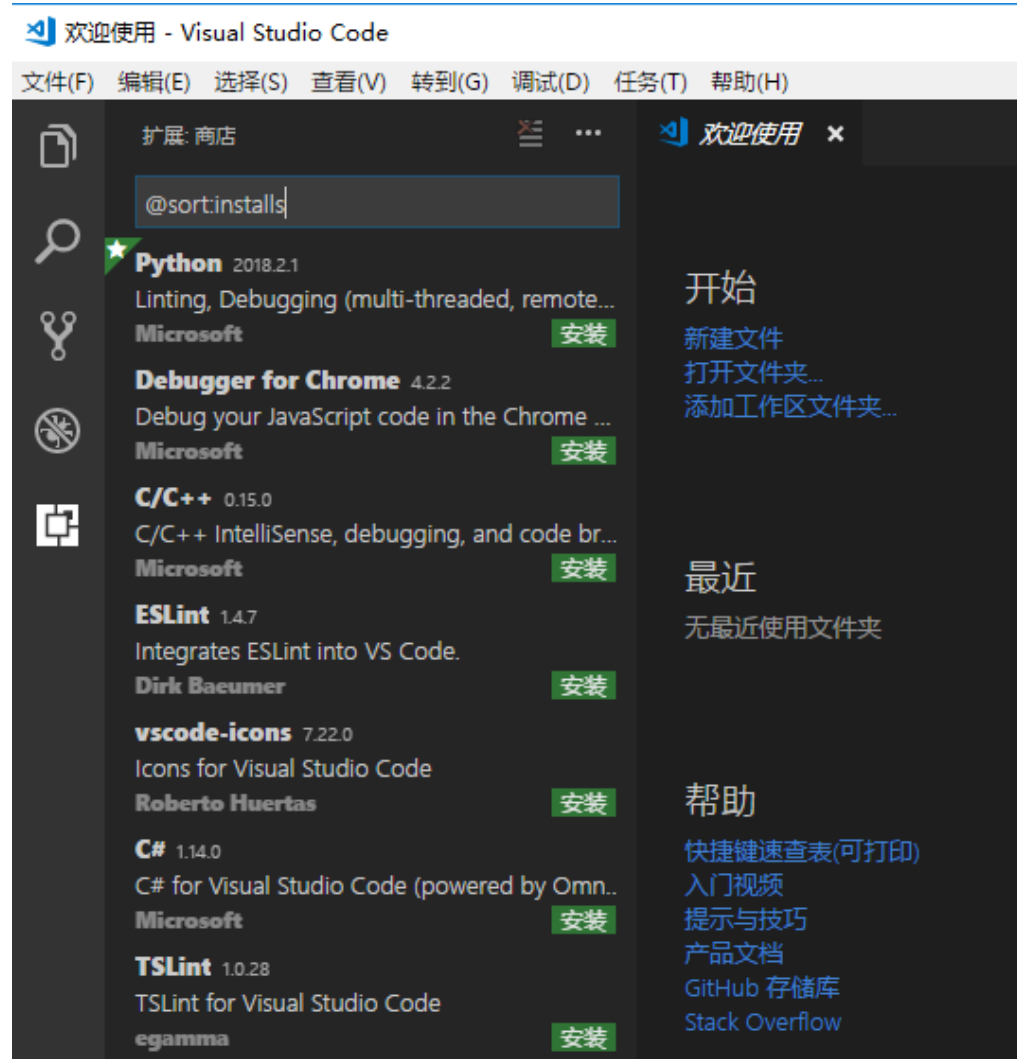
# PYTHON 编辑代码的几种方式

- Python 解释器
- IDLE 交互环境
  - Shell
  - 编辑器
- 第三方编辑器
  - Visual Studio Code
  - PyCharm Community Edition
- Jupyter
- IDLE 和输入法之间有兼容问题，强烈建议使用 VS Code



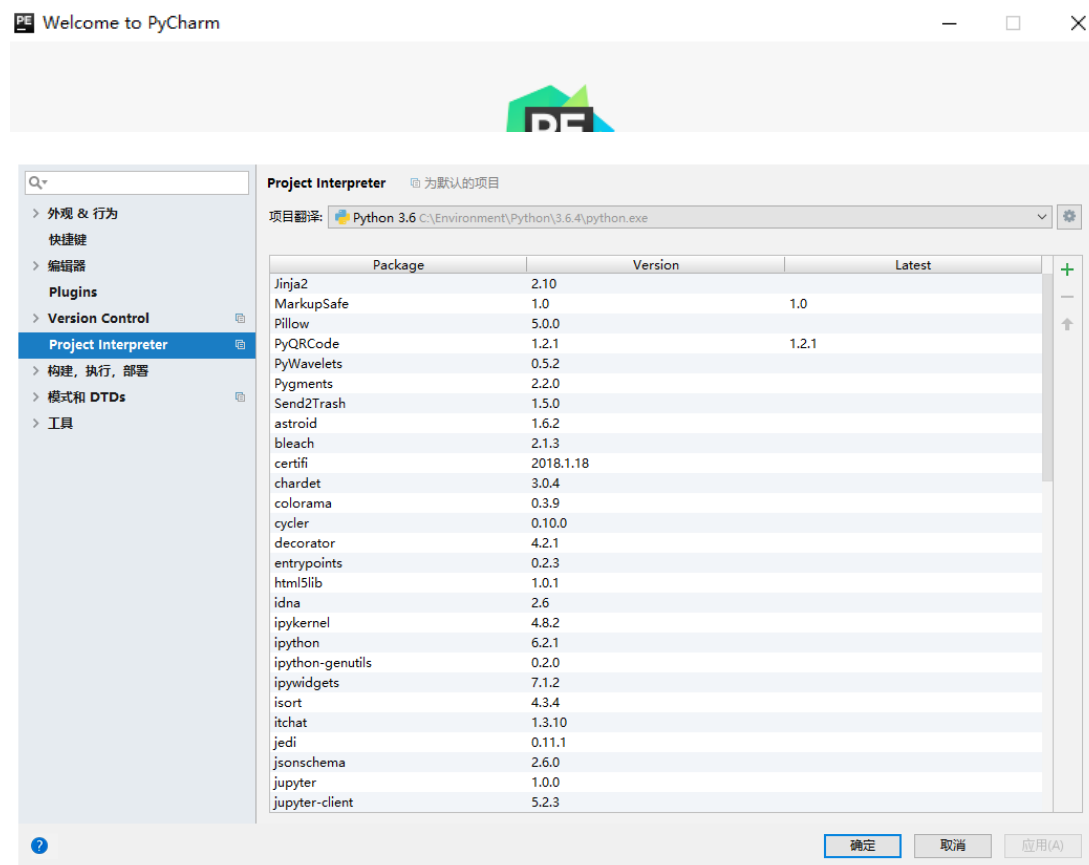
# VSCODE环境搭建

- 访问<https://code.visualstudio.com/>
- 下载安装程序
- 安装Visual Studio Code
- 打开VSCode，点击左侧第五个图标
- 安装Python扩展，重新加载VSCode
- 教学时通过和Jupyter配合使用，效果更佳（后面详细介绍）
- 点击文件，打开文件夹，选择Python项目的位置
- 编写Python代码，通过第四个图标可以调试运行（后面详细介绍）



# PYCHARM EDU 环境搭建

- 访问<https://www.jetbrains.com/pycharm-edu/download/>
- 下载安装程序
- 安装PyCharm Edu
- 复制汉化文件到PyCharm安装路径下的lib文件夹中
- 打开PyCharm
  - 右下角
    - » 设置
    - » Project Interpreter
    - » 选择系统的Python程序位置
- 创建新项目





# 编写第一个程序

1. `print('Hello World!')` # 输出字符串
2. `print(42)` # 输出数值
3. `print(3 + 4j)` # 输出复数
4. `print(True)` # 输出布尔值

# 基本数据类型

- Number ( 数值型 )
  - int ( 整数 )
  - float ( 小数 )
  - complex ( 复数 )
- bool ( 布尔型 )
  - True
  - False
- None ( 空值 )
- 特殊值
  - float('inf'), float('-inf'), float('nan')
- Sequence ( 序列型 )
  - str ( 字符串 )
  - list ( 列表 )
  - range ( 范围 )
  - dict ( 字典 )
  - tuple ( 元组 )
  - set ( 集合 )

# 数学运算

- +
- -
- \*
- /
- //
- %
- \*\*

- 加
- 减
- 乘
- 除
- 整除
- 求余
- n次方

# 比较运算

- <
- <=
- >
- >=
- ==
- !=
- is
- is not
- in
- not in

- 小于
- 小于等于
- 大于
- 大于等于
- 等于
- 不等于
- 是同一个值
- 不是同一个值
- 值在序列中
- 值不在序列中

# 逻辑运算

- and
- or
- not

- 与
- 或
- 非

# 序列类型

- 区别与其他类型的运算
- `in` / `not in`
- `+`
- `*`
- `[]`

元素在/不在序列中

连接序列

重复序列

取元素



# 字符串 ( STR )

- 由单引号、双引号括起来的各种字符
  - 'Hello World!','Hello Python!'
- 还可以是有三组 ( 单、双 ) 引号括起来的字符，可以多行
  1. """
  2. 举杯邀明月
  3. 把酒问青天
  4. 云霞出海曙
  5. 江柳共风烟
  6. """

# 字符串操作函数

- 使用分隔符拆分字符串

`str.split(separator)`

- 计算子字符串出现的次数

`str.count(substring)`

- 判断字符串是不是以前缀开始

`str.startswith(prefix)`

- 判断字符串是不是以后缀结束

`str.endswith(suffix)`

- 返回子字符串第一次出现的位置，没有找到返回-1

`str.find(substring)`      # 左边第一次出现

`str.rfind(substring)`      # 右边第一次出现

# 字符串格式化

- 一般情况下使用“+”连接字符串，但是还有很多情况简单连接不够优雅，所以有专门的字符串格式化函数：

`str.format()`

- `"{} {}".format("hello", "world")` # 按顺序填充
- `"{1} {0} {1}".format("hello", "world")` # 按指定位置填充
- `"Hello: {name}".format(name="slobber")` # 按名称填充
- `"π ≈ {:.2f}".format(3.1415926)` # 保留两位小数
- `"{:0>5d}".format(10)` # 整数不足五位，左边补零
- `"{value:#x}".format(value=250)` # 按十六进制显示整数

# 列表 ( LIST )

- 有序的元素序列

- 初始化方法:

```
l = list()
```

```
l = list(另一个序列)
```

```
l = []
```

```
l = [1, 2, 3, 4]
```

- 索引操作:

```
l[0] == 1          # 从左向右取元素
```

```
l[-1] == 4         # 从右向左取元素
```

```
l[:2] == [1, 2]    # 从下标0到2之前
```

```
l[-2:] == [3, 4]   # 从下标-2到-0之前
```

```
l[1:-1] == [2, 3]  # 从下标1到-1之前
```

列表: [ 1, 2, 3, 4 ]

正下标: 0 1 2 3 4

负下标: -4 -3 -2 -1 -0 ← -0不可用

## 操作函数

- 列表最后添加元素 x

```
list.append(x)
```

- 在列表 index 位置插入元素 x

```
list.insert(index, x)
```

- 删除列表中第一个x

```
list.remove(x)
```

- 删除列表index位置的元素, 如果不设置index则删除列表最后一个元素

```
list.pop(index)
```

- 列表排序

```
list.sort()          # 元素之间要可以比较
```

# 范围 ( RANGE )

- 一个整数等差数列，常用于循环
- 初始化方法：

```
r = range(10)
```

```
r = range(5, 15)
```

```
r = range(0, 10, 2)
```

```
r = range(10, 0, -3)
```

- 转换为list

```
l = list(range(100, 150, 5))
```

# 字典 ( DICT )

- 键值对的集合

- 初始化方法:

```
d = dict()
```

```
d = {}
```

```
d = dict(one=1, two=2, three=3)
```

```
d = {'one': 1,  
     'two': 2,  
     'three': 3}
```

- 转换为列表:

```
l = list(d)           # 键的列表
```

```
l = list(d.values())  # 值的列表
```

## 操作函数

- 获取键值对视图

```
dict.items()
```

- 获取键视图

```
dict.keys()
```

- 获取值视图

```
dict.values()
```

- 按键获取值

```
dict.get(key)
```

- 按键删除键值

```
dict.pop(key)
```



# 元组 ( TUPLE ) 、 集合 ( SET )

- 元组是一个不可变的列表

- 初始化方法：

```
t = (1, 2)
```

```
t = tuple(另一个序列)
```

- 用途：
- 函数多返回值
- 可以作为dict的键

- 集合是无序无重复的键表

- 初始化方法：

```
s = set()
```

```
s = {1, 2, 3}
```

```
s = set(另一个序列)
```

用途：

```
s = {1, 2, 3} & {2, 3, 4} # 交
```

```
s = {1, 2, 3} | {2, 3, 4} # 并
```

```
s = {1, 2, 3} - {2, 3, 4} # 差
```

# 序列元素的嵌套

- 除字符串外，一个序列可以同时包含各种类型的元素
- 例如：

```
[  
    1, 2, 'Bye',  
    {'a': [1, 2, 3], 'b': True},  
    ('a', 'b', 'c')  
]
```

# 常用内置函数

- 类型转换
- `type(x)`
- `len(x)`
- `input()`
- `print(x)`
- `open(x)`
- `sorted(x, key=cmp_function, reverse=False)`
- `dir(x)`
- `help(func)`
- `str(x), int(x), float(x), bool(x)`
- 获得一个值的类型
- 获得一个序列的长度
- 从命令行输入字符串
- 在命令行输出文本
- 打开文件
- 序列排序
- 查看对象的所有函数
- 获取帮助文档

# 三种程序结构

- 顺序
- 选择
- 循环

Python 通过缩进来表示代码的层级关系

```
1. for i in range(4):  
2.     print(i)  
  
3. with open('text.txt') as file:  
4.     while True:  
5.         line = file.readline()  
6.         if not line:  
7.             break  
8.         print(line)
```

# 函数

- 格式：

**def** 函数名(参数):  
    balabala  
    bilibili

**return** value

参数有三种形式：

1. 只有参数名“param”，必带参数
2. 带着默认值“param=value”，可选参数，使用时两种方法：
  1. 按参数顺序依次填入参数
  2. 参数名赋值跳过一些可选参数
3. 不确定参数个数“\*params”

返回值有三种形式：

1. 没有返回值，只写“return”，或什么都不写
2. 返回一个值“return value”
3. 返回多个值“return value1, value2”

# 类

不带父类，默认父类为“object”类

- 格式：

```
class 类名(父类名):
```

```
    class_prop = []
```

← 类的属性

```
    def __init__(self):
```

← 实例初始化方法，在其中定义及初始化类实例的属性

```
        self.prop = 0
```

```
    def method(self, param):
```

← 类实例的方法

```
        self.prop += 1
```

```
        return self.prop
```

```
    def class_method(param):
```

← 类的方法

```
        return 0
```



# 导入模块

- `import module_name`
- `import library_name.module_name`
- `import module_name as alias_name`
  
- `from library_name import module_name`
- `from module_name import function_name`

# PYTHON标准库

- Turtle模块
- Math模块
- Random模块
- Time模块
- OS模块

# TURTLE模 块

方法	参 数	描 述
Turtle	无	创建并返回一个乌龟对象
forward	距离	前进
backward	距离	后退
right	角度	顺时针转动乌龟
left	角度	逆时针转动乌龟
up	无	乌龟抬起尾巴
down	无	乌龟放下尾巴
color	颜色名称	改变尾巴的颜色
fillcolor	颜色名称	改变尾巴绘制的多边形的填充颜色

方法	参 数	描 述
heading	无	得到当前海龟朝向
position	无	返回当前的位置
goto	x, y	移动乌龟到 x, y 坐标位置
begin_fill	无	准备填充颜色
end_fill	无	封闭图形然后填充当前的填充颜色
dot	无	在当前位置画一个点
stamp	无	在当前位置留下一个乌龟的标识
shape	形状名称	以下几种 ‘arrow’, ‘classic’, ‘turtle’, ‘circle’

# MATH模块

方法	参 数	描 述
ceil	浮 点 数	上 取 整
floor	浮 点 数	下 取 整
trunc	浮 点 数	截 取 整 数 部 分
gcd	整 数a, 整 数b	最 大 公 约 数
sqrt	数 值	开 方
log	数 值, 基	求 对 数
各种三角函数	数 值	sin, cos, tan, asin, acos, atan
角度弧度转换: degrees、radians	数 值	根据弧度算角度, 根据角度算弧度
常量pi、e、inf、nan		$\pi$ 、e、无穷大、不是数值

# RANDOM模块

方法	参数	描述
random	无参数	生成0~1之间的一个随机浮点数
uniform	a, b	生成a~b之间的一个随机浮点数
randint	a, b	生成a~b之间的一个随机整数
shuffle	序列	打乱序列
choice	序列	从序列中随机选出一个元素
sample	序列, 长度	从序列中随机选出n个元素

# TIME 模 块

时间戳：从格林尼治时间1970年1月1日0点0分0秒开始的秒数 1522317799

时间元组：九个元素的元组，分别表示年月日时分秒、周几、年中第几天、是否夏令时

方法	参数	描述
time	无参数	得到当前时间的时间戳
gmtime	无参数	得到当前格林尼治时间的元组
localtime	无参数	得到当前时区时间的元组
strftime	format, time	按照format字符串格式化时间 "%Y-%m-%d %H:%M:%S"
strptime	str_time, format	按照format字符串将str_time转换为元组
mktime	time_tuple	将元组转换为时间戳
sleep	seconds	程序休眠若干秒



# OS模块


os 模块封装了常见的文件和目录操作

方法	参数	描述
<code>mkdir</code>	<code>dir_name</code>	创建目录
<code>rmdir</code>	<code>dir_name</code>	删除目录
<code>rename</code>	<code>src, dst</code>	重命名
<code>remove</code>	<code>file_name</code>	删除文件
<code>getcwd</code>	无参数	获取当前工作路径
<code>walk</code>	<code>dir_name</code>	遍历目录
<code>path.join</code>	<code>dir_name, file_name</code>	连接目录与文件名

方法	参数	描述
<code>path.split</code>	<code>path</code>	分割文件名与目录
<code>path.abspath</code>	<code>path</code>	获取绝对路径
<code>path.dirname</code>	<code>path</code>	获取路径
<code>path.basename</code>	<code>path</code>	获取文件名或文件夹名
<code>path.splitext</code>	<code>path</code>	分离文件名与扩展名
<code>path.isfile</code>	<code>path</code>	路径是否是一个文件
<code>path.isdir</code>	<code>path</code>	路径是否是一个目录

# 异常的处理

- 格式:

 尝试运行以下代码  
`try:`


`balabala`

`bilibili`


`i = int('a')`

 字符串'a'无法转换成整数，引发异常

`except Exception as e:`

 捕获到异常，根据异常内容进行处理

`print(e)`

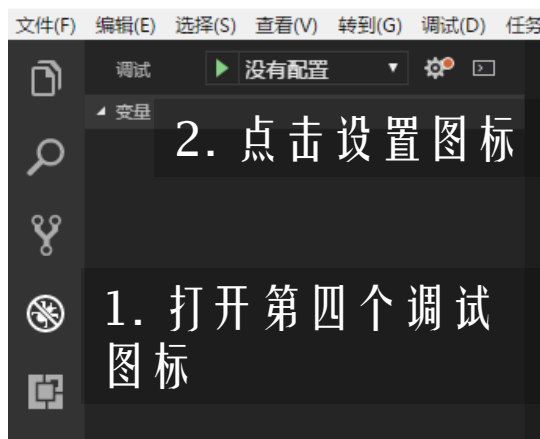
 最简单的操作，将异常信息打印出来

# 一个真实的异常处理

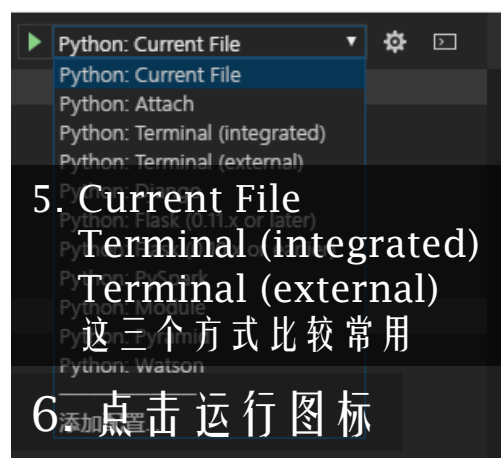
```
1. import errno
2. import os

3. def mkdir_p(path):
4.     try:
5.         os.makedirs(path)
6.     except OSError as e:
7.         if e.errno == errno.EEXIST and os.path.isdir(path):
8.             pass
9.         else:
10.            raise
```

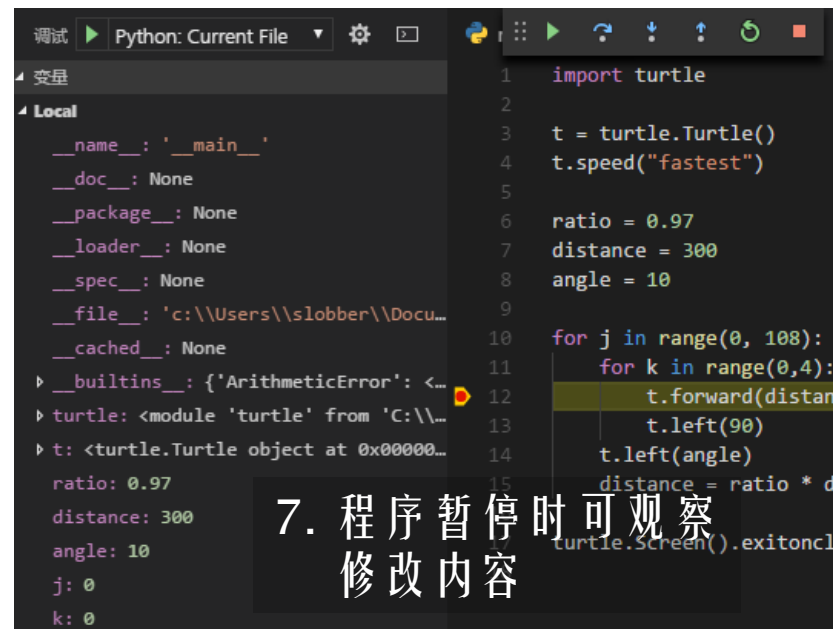
# 调试



2. 点击设置图标



6. 点击运行图标



# PIP的 使用

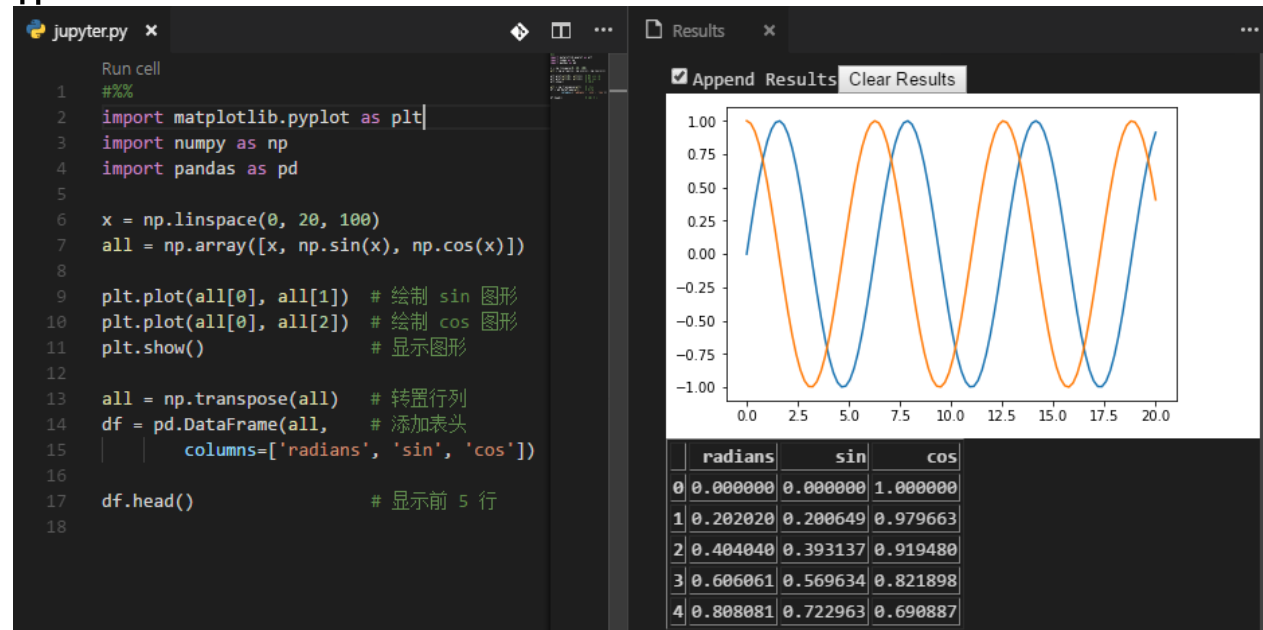
- 如果一个语言没有完善的扩展机制是没有生命力的
- Python可以非常容易的使用别人制作的扩展库
- 在命令行下，可以使用“pip”命令安装Python的扩展包

```
C:/> pip install 库名
```

```
C:/> pip install --no-index --find-links=path/of/offline/dir 库名
```

# VSCODE与JUPYTER整合

- 在VSCode中安装Jupyter插件
- 命令行下pip install jupyter
- 在代码窗口要运行的语句分组之前添加注释“#%%”
- 点击出现的“Run Cell”
- 右侧将出现Jupyter面板，包含输出内容



# 扩展库

- Sinomaps 库
- Numpy 库
- Matplotlib 库
- Pandas 库
- Requests 库
- PyQuery 库
- Itchat 库

# SINOMAPS 库

- C:\> pip install sinomaps
- 包含教材中所需的所有库和模块，自动安装 “numpy” 、 “matplotlib” 、 “pandas” 、 “requests” 、 “itchat”



# NUMPY 库

- 支持多维数组与矩阵运算，此外也针对数组运算提供大量的数学函数库。
- `np.array(a list)`
- `ndarray.shape`
- `ndarray.astype(new_type)`
- `ndarray.swapaxes(axis_a, axis_b)`
- `np.rollaxis(ndarray, old_axis, new_axis)`
- `np.zeros(dimission_tuple)`
- `np.ones(dimission_tuple)`

# MATPLOTLIB 库

- 绘图库，常用其中的pyplot模块
- `plt.plot(value_list)`
- `plt.plot(x_list, y_list)`
- `plt.show()`
- `plt.subplot(row_count, col_count, current_pos)`

# PANDAS 库

- Pandas 是基于 NumPy 的库，更适用于统计分析
- 一般支持二维表
- 涉及以下概念：
  - Series 一维数组
  - DataFrame 二维数组
  - Panel 三维数组

# REQUESTS 库

- Requests是一套HTTP操作库
- 用于从程序中发起HTTP请求
- `requests.get()`
- `requests.post()`

# PYQUERY 库

- 可以很方便的查询html中的内容

```
from pyquery import PyQuery as pq
```

```
doc = pq('html源代码')
```

```
p = doc('p') # 将html中所有 p 标签筛选出来
```

```
p_with_t = doc('p.title') # 将html中所有class="title"的p标签筛选出来
```

```
for pp in p:
```

```
    print(pp.text()) # 输出p中每个元素的文本
```

# ITCHAT 库

- Itchat是一个将个人微信转变成聊天机器人的库

- decorator (装饰器)

```
@装饰器函数  
def 函数(参数):  
    balabala  
    bilibili
```

- 作用：在调用函数前后注入一些其他代码，简化代码逻辑，提升可读性
- 一个隐藏概念：AOP (面向切面编程)