

https://www.youtube.com/watch?v=_uQrJ0TkZlc&t=2850s

小练习：类型转换

双引号单引号问题

输出结果

三个单引号，可以写长文本

字符串数组

输出

复制

位置操作

Formatted string 格式化字符串

大小写操作

字符替换查找

数字运算操作

简化运算

运算法则

运算函数

round(x, n)

abs(x)

math module

Logical Operators

if

小练习

if and/or

Comparison Operators

practice

Loop

while

Guess number game

1:39:42 Car Game

For

range()

List

1D-List

append()

insert()

remove()

clear()

pop()

index()

in

count()

sort()

reverse()

copy()

not in

2D-List_Matrix

Tuple

Unpacking

Dictionary

get()

Update Value

Add new key value

exercise Phone number

split()

Function

pass information to your function

parameters

arguments

Return

Function&Return

max()

Exceptin

Classes

inherit

Module

Python Module Index

Generate Random value

PathLib

exists()

mkdir()

rmdir()

glob()

overwrite

package

Create Package

Import Package Module Function

Import Package Entire module

Install Package

openpyxl

```
1 birth_year=input('Birth_year')
2 print(type(birth_year)) # 输出birth_year的类型
3 age= 2019-int(birth_year) #将birth_year转换为int进行计算
4 print(type(age)) # 输出age的类型
5 print(age)
```

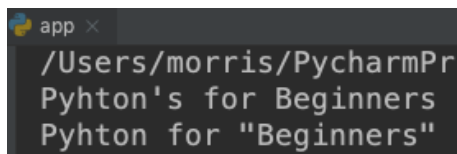
小练习：类型转换

```
1 weight=input('weight:') # input输入的类型string
2 kilogram_weight=int(weight)*10
3 print(kilogram_weight)
```

双引号单引号问题

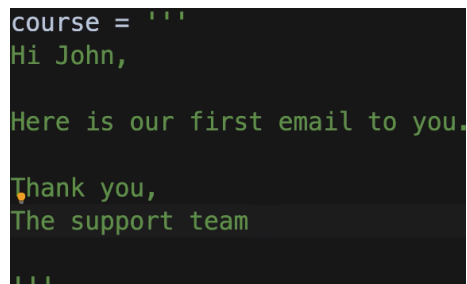
```
1 course="Python's for Beginners" # 需要用到'时用""
2 print(course)
3 course2='Python for "Beginners"' # 需要用到""时用'
4 print(course2)
```

输出结果



```
app x
/Users/morris/PycharmPr
Pyhton's for Beginners
Pyhton for "Beginners"
```

三个单引号，可以写长文本



```
course = '''
Hi John,

Here is our first email to you.

Thank you,
The support team
'''
```

字符串数组

输出

```
1 course="Python's for Beginners"
2     0123456
3 print(course[0]) 输出P
4 print(course[1]) 输出y
5 print(course[-1]) 输出s
```

```

6 print(course[-2]) 输出r
7 print(course[0:3]) 输出0-2的字符，不包括3位置的字符，输出Pyt
8 print(course[0:]) 输出全部的字符，输出Python's for Beginners
9 print(course[1:]) 输出从1位置之后的全部字符，包括1位置字符，输出ython's for Beginners
10 print(course[:5]) 输出从0位置之后的至5位置（不包括5位置）字符，包括0位置字符，输出Pytho
11 print(course[:]) 输出从全部字符，输出Python's for Beginners

```

复制

```

1 course="Python's for Beginners"
2 another=course[:] # 用于复制course中全部字符
3 print(another) 输出Python's for Beginners

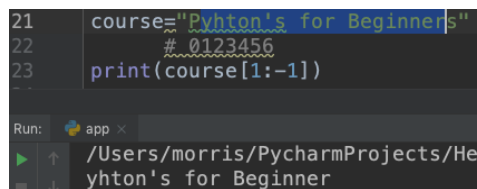
```

位置操作

```

1 course="Python's for Beginners"
2     # 0123456
3 print(course[1:-1])

```



Formatted string 格式化字符串

```

1 first='John'
2 last='Smith'
3 msg= f'{{first}} [{{last}}] is a coder'
4 # formatted 有格式的 f' '
5 print(msg)

```

```
John [Smith] is a coder
```

大小写操作

```

course='Python for Beginners'
print(course.upper()) #全部大写
print(course.lower()) #全部小写
print(course)

```

```
PYTHON FOR BEGINNERS
python for beginners
Python for Beginners
```

字符替换查找

```

1 course='Python for Beginners'
2 print(course.find('o')) # 查找o在字符串中的位置

```

```

3 print(course.replace('Beginners'), 'Absolute Beginners')
4 # 替换字符
5 course='Python for Beginners'
6 print('Python' in course)
7 # 得到字符是否在字符串内, 返回true false
8 print(len(course))
9 # 得到字符串长度

```

数字运算操作

```

1 print(10/3)
2 # 得到除不尽浮点数
3 print(10//6)
4 # 得到整数部分
5 print(10%6)
6 # 得到余数部分 4
7 print(10**3)
8 # 得到10的3次方 1000

```

简化运算

```

1 x=10
2 x-=3
3 print(x) 7
4 y=10
5 y+=3
6 print(y) 13

```

运算法则

```

1 x=10+3*2**2
2 print(x)
3 # 先运算2的平方, 然后乘3

```

运算函数

round(x, n)

Python 中关于 round 函数的小坑<https://www.runoob.com/w3cnote/python-round-func-note.html>

在python2.7的doc中, round()的最后写着, "Values are rounded to the closest multiple of 10 to the power minus ndigits; if two multiples are equally close, rounding is done away from 0." 保留值将保留到离上一位更近的一端（四舍六入），如果距离两端一样远，则保留到离0远的一边。所以round(0.5)会近似到1，而round(-0.5)会近似到-1。

但是到了python3.5的doc中, 文档变成了"values are rounded to the closest multiple of 10 to the power minus ndigits; if two multiples are equally close, rounding is done toward the even choice." 如果距离两边一样远，会保留到偶数的一边。比如round(0.5)和round(-0.5)都会保留到0，而round(1.5)会保留到2。

所以如果有项目是从py2迁移到py3的, 可要注意一下round的地方（当然, 还要注意/和//, 还有print, 还有一些比较另类的库）。

简单的说就是，`round(2.675, 2)` 的结果，不论我们从python2还是3来看，结果都应该是2.68的，结果它偏偏是2.67，为什么？这跟浮点数的精度有关。我们知道在机器中浮点数不一定能精确表达，因为换算成一串1和0后可能是无限位数的，机器已经做出了截断处理。那么在机器中保存的2.675这个数字就比实际数字要小那么一点点。这一点点就导致了它离2.67要更近一点点，所以保留两位小数时就近似到了2.67。

以上。除非对精确度没什么要求，否则尽量避开用`round()`函数。近似计算我们还有其他的选择

abs(x)

绝对值函数

math module

ceil(x) floor(x)

```
1 import math
2 x=23.6
3 print(math.ceil(x))
4 print(math.floor(x))
5 # ceil(x) 往大了算取整 24
6 # floor(x) 往小了算取整 23
7 x=-23.6
8 print(math.ceil(x))
9 print(math.floor(x))
10 # ceil(x) 往大了算取整 -23
11 # floor(x) 往小了算取整 -24
```

math — Mathematical functions¶ <https://docs.python.org/3/library/math.html>

Logical Operators

if

```
1 is_hot=True
2 if is_hot:
3     print("It's a hot day")
4     # if内执行语句
5 print("Enjoy your day")
6 # if外执行语句
7 #
8 is_hot=False
9 is_cold=True
10 if is_hot:
11     print("It's a hot day")
12     # if内执行语句
13 elif is_cold: # elif是else if 简写
14     print("It's a cold day")
15 else:
16     print("It's a cold day")
```

小练习

Price of a house is \$1M.

If buyer has **good credit**,
they need to put down **10%**

Otherwise

they need to put down **20%**

Print the down payment

```
1 good_credit=True
2 Price= 1000000
3 if good_credit:
4     print("They need to put down 10%")
5     Price=Price * 0.1
6     # if内执行语句
7 else:
8     print("They need to put down 20%")
9     Price = Price * 0.2
10 # if外执行语句
11 print(f"Your final price is:${Price}")
```

if and/or

```
1 good_credit=True
2 good_boy=True
3
4 if good_credit and good_boy: #both
5     print("They need to put down 20%")
6
7 good_credit=True
8 good_boy=True
9
10 if good_credit or good_boy: #at least one
11     print("They need to put down 20%")
12
13 good_credit=True
14 criminal_boy=False
15
16 if good_credit and not criminal_boy: #将criminal_boy变为相反
17     print("They need to put down 20%")
```

Comparison Operators

>= > != < <=

practice

```
1 name = 'QIANJIANHENG'
2 char_numbers = len(name) #12
3 print(char_numbers)
4 if char_numbers < 3:
5     print("name must be at least 3 characters")
6 elif char_numbers > 50:
7     print("name can be maximum of 50 characters")
8 else:
9     print("name looks good")
```

Loop

while

```
1 i=1
2 while i<=5:
3     print(i)
4     i=i+1
5 print("Done")
6 # 输出结果为:
7 # 1
8 # 2
9 # 3
10 # 4
11 # 5
12 # Done
13
14 i=1
15 while i<=5:
16     print('*' * i) # 重复输出字符
17     i=i+1
18 print("Done")
19
20 # *
21 # **
22 # ***
23 # ****
24 # *****
25 # Done
```

Guess number game

```
1 secret_number = 9
2 guess_count = 0
```

```

3 guess_limit = 3
4 # in order to make your code become more readable
5 while guess_count < guess_limit:
6     guess = int(input('Guess:'))
7     guess_count = guess_count + 1
8     if guess == secret_number:
9         print("You win!")
10        break # terminate all loops
11 else: # while 也有else 若没被猜中 while 循环结束执行
12     print("You don't get the right secret_number")

```

1:39:42 Car Game

```

1 start = 'start'
2 stop = 'stop'
3 quit = 'quit'
4 help = 'help'
5 is_start = False
6 # in order to make your code become more readable
7 while True:
8     user_input = input(">").lower()
9     if user_input == start:
10        if is_start == True:
11            print("The car has already started")
12        else:
13            print("Car started... Ready to go!")
14            is_start = True
15    elif user_input == help: # 注意这里要用elif
16        print("start - to start the car")
17        print("stop - to stop the car")
18        print("quit - to exit")
19    elif user_input == stop: # 注意这里要用elif
20        if is_start:
21            print("Car stopped.")
22            is_start = False
23        else:
24            print("The car has already stopped")
25    elif user_input == quit: # 注意这里要用elif
26        break
27    else:
28        print("I don't understand that...")

```

For

```

1 for item in 'Python':

```

```
2     print(item)
3 # P
4 # y
5 # t
6 # h
7 # o
8 # n
9
10 for item2 in ['QIAN', 'ZHENG', 'FAN']:
11     print(item2)
12 # QIAN
13 # ZHENG
14 # FAN
```

range()

```
1 for item2 in range(10):
2     print(item2)
3 # 0
4 # 1
5 # 2
6 # 3
7 # 4
8 # 5
9 # 6
10 # 7
11 # 8
12 # 9
13
14 for item2 in range(5, 10):
15     print(item2)
16 # 5
17 # 6
18 # 7
19 # 8
20 # 9
21
22 for item2 in range(5, 10, 2):
23     print(item2)
24 # 5
25 # 7
26 # 9
27
28 prices = [10, 20, 30]
29 total = 0
```

```

30 for price in prices:
31     total += price
32 print(f"Total:{total}")
33
34 for x in range(4):
35     print(f"x:{x}")
36     # x: 0
37     # x: 1
38     # x: 2
39     # x: 3
40
41     for x in range(4):
42     for y in range(3):
43         print(f"(x,y):{x, y}")
44     # (x, y): (0, 0)
45     # (x, y): (0, 1)
46     # (x, y): (0, 2)
47     # (x, y): (1, 0)
48     # (x, y): (1, 1)
49     # (x, y): (1, 2)
50     # (x, y): (2, 0)
51     # (x, y): (2, 1)
52     # (x, y): (2, 2)
53     # (x, y): (3, 0)
54     # (x, y): (3, 1)
55     # (x, y): (3, 2)
56
57 numbers = [5, 2, 5, 2, 2]
58 for x in numbers:
59     print('x' * x)
60 # xxxxx
61 # xx
62 # xxxxx
63 # xx
64 # xx
65
66 another way:
67 numbers = [5, 2, 5, 2, 2]
68 for x in numbers:
69     output = ''
70     for y in range(x):
71         output += 'x'
72     print(output)

```

List

1D-List

```
1 names = ['A', 'B', 'C', 'D', 'E']
2 print(names[0])
3 print(names[2])
4 print(names[-1])
5 print(names[-2])
6 # A
7 # C
8 # E
9 # D
10
11 names = ['A', 'B', 'C', 'D', 'E']
12 print(names[2:4])
13 print(names[2:])
14 print(names[:])
15 print(names)
16 # ['C', 'D']
17 # ['C', 'D', 'E']
18 # ['A', 'B', 'C', 'D', 'E']
19 # ['A', 'B', 'C', 'D', 'E']
20
21 # find the largest number in list
22 y = 0
23 numbers = [100, 2, 6, 3, 7, 5, 10, 2, 6, 3, 8]
24 for x in numbers:
25     if y < x:
26         y = x
27 print('the largest number in list is:', y)
28 # the largest number in list is: 100
```

append()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers.append(20) # Append object to the end of the list.
3 # [5, 2, 1, 7, 4, 5, 20]
```

insert()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers.insert(0, 1) # Insert object before index.
3 # [1, 5, 2, 1, 7, 4, 5]
```

remove()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers.remove(5) # Remove first occurrence出现 of value.
3 # [2, 1, 7, 4, 5]
```

clear()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers.clear() # Remove all items from list.
3 # []
```

pop()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers.pop() # Remove and return item at index (default last).
3 # [5, 2, 1, 7, 4]
```

index()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 print(numbers.index(7)) # Return first index of value.
3 # 3
```

in

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 print(80 in numbers)
3 # False
4 # 80不在里面
```

count()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 print(numbers.count(5)) # Return number of occurrences of value.
3 # 2
```

sort()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers.sort() # Stable sort *IN PLACE*.
3 print(numbers)
4 # [1, 2, 4, 5, 5, 7]
```

reverse()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers.sort() # Stable sort *IN PLACE*.
3 numbers.reverse()
4 print(numbers)
5 # [7, 5, 5, 4, 2, 1]
6 # 倒序
7
8 numbers = [5, 2, 1, 7, 4, 5]
9 numbers.reverse()
```

```
10 print(numbers)
11 # [5, 4, 7, 1, 2, 5]
12 # 倒置
```

copy()

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 numbers2 = numbers.copy() # Return a shallow copy of the list.
3 numbers.append(10)
4 print(numbers)
5 print(numbers2)
6 # [5, 2, 1, 7, 4, 5, 10]
7 # [5, 2, 1, 7, 4, 5]
8 # 拷贝
```

not in

```
1 numbers = [5, 2, 1, 7, 4, 5]
2 uniques=[]
3 for number in numbers:
4     if number not in uniques: # 如果uniques中没有number元素
5         uniques.append(number) # 则添加
6 print(uniques)
7
8 # [5, 2, 1, 7, 4]
```

2D-List_Matrix

矩阵

```
1 matrix = [
2     [1, 2, 3],
3     [4, 5, 6],
4     [7, 8, 9]
5 ]
6 print(matrix[0][1])
7 print(matrix[1][1])
8 print(matrix[2][1])
9 print(matrix[0][0])
10 print(matrix[0][1])
11 print(matrix[0][2])
12 print(matrix[0])
13 print(matrix[1])
14 print(matrix[2])
15 # 2
16 # 5
17 # 8
18 # 1
19 # 2
```

```

20 # 3
21 # [1, 2, 3]
22 # [4, 5, 6]
23 # [7, 8, 9]
24 for row in matrix:
25     for item in row:
26         print(item)
27 # Output every elements of this matrix:
28 # 1
29 # 2
30 # 3
31 # 4
32 # 5
33 # 6
34 # 7
35 # 8
36 # 9

```

Tuple

```

1 numbers = (1,2,3)
2 numbers[0]=10
3 print(numbers)
4 # 'tuple' object does not support item assignment
5 # 圆括号使用, 则为元组, 不能被分配、赋值

```

Unpacking

```

1 # 不止Tuple能用
2 coordinates = (1, 2, 3)
3 x, y, z = coordinates # Unpacking
4 print(x, y, z)
5
6 # 相当于
7 # x=coordinates[0]
8 # y=coordinates[1]
9 # z=coordinates[2]
10 # 1 2 3

```

Dictionary

get()

```

1 # Dictionary 用{ }表示
2 customer = {
3     "name": "John Smith",
4     "age": 30,
5     "is_verified": True

```



```

6 }
7 print(customer["name"])
8 # 获取字典中name项的值
9 # John Smith
10 print(customer.get("birthday"))
11 # 在字典中找不到此项
12 # None
13 print(customer.get("birthday", "Jan 1 1980"))
14 # 因为临时赋值"Jan 1 1980"
15 # Jan 1 1980
16 print(customer.get("birthday"))
17 # 在字典中找不到此项
18 # None
19
20

```

Update Value

```

1 customer = {
2     "name": "John Smith",
3     "age": 30,
4     "is_verified": True
5 }
6 # Update Value
7 customer["name"] = "QIAN"
8 print(customer["name"])
9 # QIAN

```

Add new key value

```

1 customer = {
2     "name": "John Smith",
3     "age": 30,
4     "is_verified": True
5 }# Add new key value
6 customer["Birthdate"] = "March 5 1996"
7 print(customer["Birthdate"])
8 # March 5 1996

```

exercise Phone number

```

1 # Dictionary 用{ }表示
2 digits_mapping = {
3     "1": "one",
4     "2": "two",
5     "3": "three",
6     "4": "four"

```

```

7 }
8 # exercise Phone number
9 pho_num = input("Phone:")
10 output=''
11 for ch in pho_num:
12     # 此时pho_num当做字符，赋给ch
13     # 例如输入'12345' 则ch依次为1 2 3 4 5 字符型
14     output+=digits_mapping.get(ch,'!')+ ' '
15     # ch为字符型，所以可以进行get
16     # 若字典里没有某项，则为'!'
17 print(output)
18 # Phone:123456
19 # one two three four ! !

```

split()

```

1 # split
2 message= input(">")
3 words= message.split(' ')# 将输入字符分开存储
4 print(words)
5 # I am good
6 # ['I', 'am', 'good']
7
8 # split
9 message = input(">")
10 words = message.split(' ') # 将输入字符按' '分开存储
11 emojis = {
12     ".):": "😄",
13     ":((": "😞"
14 }
15 output = ''
16 for word in words:
17     output += emojis.get(word, word) + ' '
18 print(output)
19 # i am :(
20 # i am 😞

```

Function

```

1 # Function定义
2 def greet_user():
3     print('Hi there!')
4     print('Welcome abroad')
5 print("Start")# step 1
6 greet_user()# step 2
7 print("Finish")# step 3

```

```
8 # Start
9 # Hi there!
10 # Welcome abroad
11 # Finish
```

pass information to your function

```
1 # Function-pass information to your function
2 def greet_user(name):
3     print(f'Hi {name}!')
4     print('Welcome abroad')
5 print("Start")# step 1
6 greet_user("QIAN")# step 2
7 greet_user("ZHENG")# step 3
8 print("Finish")# step 4
9 # Start
10 # Hi QIAN!
11 # Welcome abroad
12 # Hi ZHENG!
13 # Welcome abroad
14 # Finish
15
16 def greet_user(first_name, last_name):
17     print(f'Hi {first_name} {last_name}!')
18     print('Welcome abroad')
19 print("Start") # step 1
20 greet_user("JIANHENG", "QIAN") # step 2
21 greet_user("SHIYU", "ZHENG") # step 3
22 print("Finish") # step 4
23 # Start
24 # Hi JIANHENG QIAN!
25 # Welcome abroad
26 # Hi SHIYU ZHENG!
27 # Welcome abroad
28 # Finish
```

Difference between parameters参数, 参量 and arguments

parameters

定义在函数中用于接收信息

arguments

我们传递给函数的实际信息

```
# Function-pass information to your function
def greet_user(name):
    print(f'Hi {name}!')
    print('Welcome abroad')

print("Start") # step 1
greet_user("QIAN") # step 2
greet_user("ZHENG") # step 3
print("Finish") # step 4
```

Parameters (points to `name` in the function definition)

Arguments (points to the string values in the function calls)

```
1 # Function-pass information to your function
2 def greet_user(first_name, last_name):
3     print(f'Hi {first_name} {last_name}!')
4     print('Welcome abroad')
5 print("Start") # step 1
6 greet_user(first_name="JIANHENG", last_name="QIAN")
7 # step 2可以更加明确Arguments所代表的意思，位置与顺序
8 # cal_cost(total=50,shipping=5,discount=0.1)
9 print("Finish") # step 3
10 # Start
11 # Hi JIANHENG QIAN!
12 # Welcome abroad
13 # Finish
```

```
print("Start") # step 1
greet_user("JIANHENG", last_name="QIAN")
# step 2可以更加明确Arguments所代表的意思，位置与顺序
# cal_cost(total=50,shipping=5,discount=0.1)
print("Finish") # step 3
# Start
# Hi JIANHENG QIAN!
# Welcome abroad
# Finish
```

Position argument (points to `"JIANHENG"`)

keyword argument (points to `last_name="QIAN"`)

Keyword argument 要使用在Position argument 之后

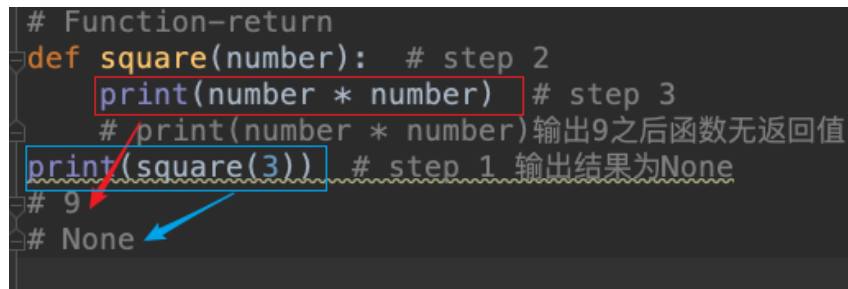
Return

```
1 # Function-return
2 def square(number): # step 2
3     return number * number # step 3
4 print(square(3)) # step 1
5 # 9
```

```

1 # Function-return
2 def square(number): # step 2
3     print(number * number) # step 3
4     # print(number * number)输出9之后函数无返回值
5 print(square(3)) # step 1 输出结果为None
6 # 9
7 # None

```



```

# Function-return
def square(number): # step 2
    print(number * number) # step 3
    # print(number * number)输出9之后函数无返回值
print(square(3)) # step 1 输出结果为None
# 9
# None

```

Function&Return

```

1 # return
2 def emoji_convert(message):
3     words = message.split(' ') # 将输入字符按' '分开存储
4     emojis = {
5         ":)": "😊",
6         ":(": "😞"
7     }
8     output = ''
9     for word in words:
10         output += emojis.get(word, word) + ' '
11     return output # 赋予返回值
12
13
14 message = input(">")
15 print(emoji_convert(message)) # 输出返回值
16 # i am :(
17 # i am 😞

```

max()

```

1 # 找出最大值
2 numbers = [1, 2, 5, 80, 6, 20, 30, 40, 8, 9, 55, 64]
3 print(max(numbers))
4 # 80

```

Exceptin

```

1 # Exceptin

```

```

2  try:
3      age=int(input('age:'))
4      print(age) # 输出返回值
5  except ValueError:
6      # 值错误时
7      print('Invalid value')
8
9      # Exceptin
10 try:
11     age = int(input('age:'))
12     income = 20000
13     risk = income / age
14     print(age) # 输出返回值
15 except ZeroDivisionError: # 此时能用除数不能为0错误
16     print('Age cannot be 0')
17 except ValueError:
18     # 此时不能用值错误, 可以同时使用两个exception
19     print('Invalid value')
20
21 # age:0
22 # Age cannot be 0
23 # age:asd
24 # Invalid value
25 # age:123
26 # 123

```

Classes

define new type and model real concepts

```

1  # Class
2  class Point:
3      def move(self):
4          print("move")
5
6      def draw(self):
7          print("draw")
8
9
10 point1 = Point() # 定义新类
11 point1.x = 10
12 point1.y = 20
13 print(point1.x)
14 point1.draw()
15
16 # 10

```

```

17 # draw
18
19 # Class
20 class Point:
21     def __init__(self, x, y):
22         self.x = x
23         self.y = y
24
25     def move(self):
26         print("move")
27
28     def draw(self):
29         print("draw")
30
31
32 point1 = Point(10, 20) # 定义新类
33 point1.x=12
34 print(point1.x)
35 # 12

```

```

1 # Class
2 class Person:
3     def __init__(self, name):
4         self.name = name
5
6     def talk(self):
7         print(f"Hi,I am {self.name}") # 定义函数内变量
8
9
10 person1 = Person("QIAN") # 定义新类
11 person1.talk()
12 # Hi,I am QIAN
13
14 bob = Person("Bob Smith")
15 bob.talk()
16 # Hi,I am Bob Smith

```

inherit

继承

We should not define something twice

```

1 # Class inherence
2 class Mammal:
3     def walk(self):

```

```

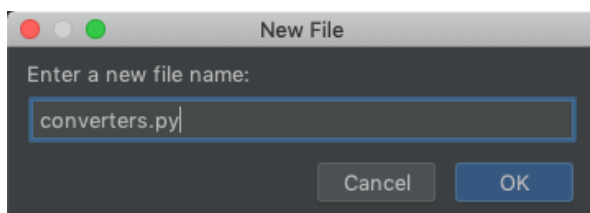
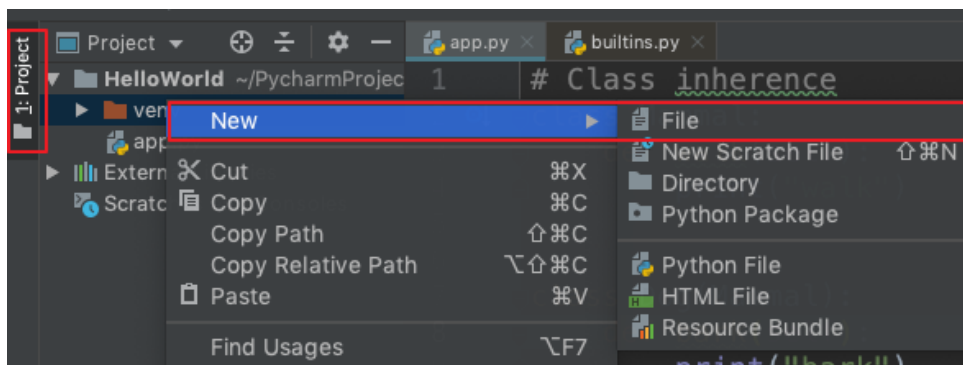
4         print("walk")
5
6
7     class Dog(Mammal):
8         def bark(self):
9             print("bark")
10        # Dog 继承Mammal类
11        # 并再定义自己的函数
12
13
14    class Cat(Mammal):
15        def be_annoying(self):
16            print("annoying")
17        # Cat 继承Mammal类
18        # 并再定义自己的函数
19
20
21    dog1 = Dog()
22    dog1.bark()
23    cat1 = Cat()
24    cat1.be_annoying()
25    # bark
26    # annoying

```

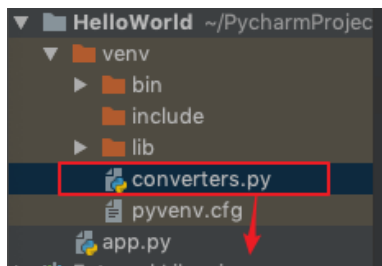
Module

模块

新建文件



拖拽出来



导入全体converters Module

```
1 # Module
2 import converters
3 # 导入全体converters Module
4 converters.Hi()
5 # Hello 🤖
```

导入converters Module 中的函数

```
1 from converters import Hi
2 # 导入converters Module 中的函数
3 Hi()
4 # Hello 🤖
```

converters.py

```
1 # converters.py
2 def numbers_sort(numbers):
3     max = numbers[0]
4     for number in numbers:
5         if number > max:
6             max = number
7     print(f'最大值为: {max}')
8     # or return max
```

app.py

```
1 # app.py Module
2 from converters import numbers_sort
3
4 # 导入converters Module 中的函数
5 # 并进行module 的函数执行, 例: 排序
6 numbers = [1, 2, 5, 80, 6, 20, 30, 40, 8, 9, 55, 64]
7 numbers_sort(numbers)
8 # or print(numbers_sort(numbers))
9 # 最大值为: 80
```

Python Module Index

intimidate [in'timideit]

vt. 恐吓

<https://docs.python.org/3/py-modindex.html>

Generate Random value

```
1 # Generate Random value
2 import random
3 for i in range(3):
4     print(random.random())
5
6 # 生成3随机数
7
8 # Generate Random value
9 import random
10 for i in range(3):
11     print(random.randint(10,20))
12
13 # 生成3个10-20 随机整数
14
15 import random
16
17 member = ['QIAN', 'ZHENG', 'HENG', 'LOVE']
18 leader = random.choice(member)
19 print(leader)
20 # 从list中随机挑选
21
22 # Generate Random int value array
23 import random
24
25
26 class Dice:
27     def roll(self):
28         first = random.randint(1, 6)
29         second = random.randint(1, 6)
30         return first, second
31
32
33 num = Dice()
34 print(num.roll())
35 # 从list中随机挑选1-6整数 数组
36 # (3, 5)
```

PathLib

exists()

```
1 # Pathlib
2 from pathlib import Path
3 path =Path("ecommerce")
```

```

4 print(path.exists())
5 # 检测ecommerce是否存在
6
7 # True

```

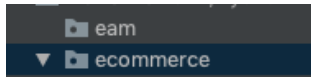
mkdir()

```

1 # Pathlib
2 from pathlib import Path
3
4 path = Path("eam")
5 print(path.mkdir())
6 # make a directory
7 # None

```

创建了directory 目录



rmdir()

```

1 # Pathlib
2 from pathlib import Path
3
4 path = Path("eam")
5 print(path.rmdir())
6 # remove a directory
7 # None

```

移除了directory 目录

glob()

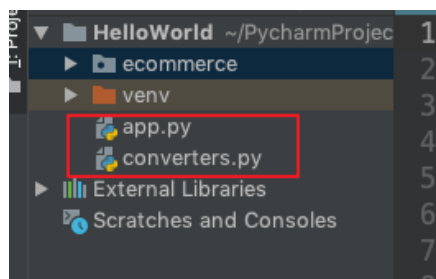
Iterate迭代 over this subtree子树 and yield产生 all existing files (of any kind, including directories) matching the given relative pattern.

```

1 # Pathlib
2 from pathlib import Path
3
4 path = Path()
5 for file in path.glob('*.py'):
6     # 第一个*代表任何文件名
7     # '.py'代表py格式的文件
8     print(file)
9     # 输出所有当前目录下的.py文件名
10 # app.py
11 # converters.py

```

输出结果即为这两个文件文件名



overwrite

覆盖

```
def numbers_sort(numbers):
    max = numbers[0]
    for number in numbers:
        if number > max:
            max = number
    print(f'最大值为: {max}')
# 或
# return max
```

覆盖了Python内建的max函数

redefining this max function

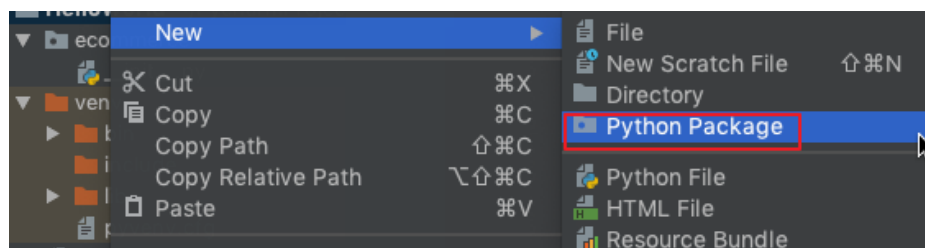
Changing the meaning of built-in function of python, this is a bad practice

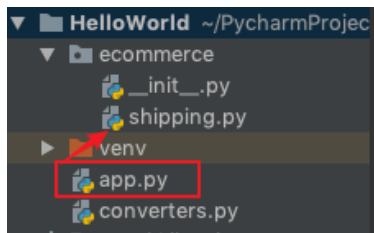
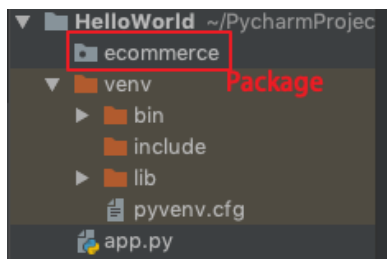
package

A better approach to organize related modules inside of a package, so a package is a container for multiple modules



Create Package





Import Package Module Function

```
1 # app.py Import Package Function
2 import ecommerce.shipping
3 ecommerce.shipping.calc_shipping()
4 # 访问Package module 中的Function
5
6 Another way
7 # app.py Import Package Function
8 from ecommerce.shipping import calc_shipping
9
10 calc_shipping()
11 # 直接导入Package module 中的Function
12 # 便可直接使用function
```

Import Package Entire module

```
1 # app.py Import Package entire module
2 from ecommerce import shipping
3
4 shipping.calc_shipping()
5 shipping.abc()
6 shipping.ghj()
7 # 直接导入entire module
8 # 便可使用function
```

Install Package

Find, install and publish Python packages with the Python Package Index

<https://pypi.org/>

openpyxl

openpyxl 2.6.2

```
pip install openpyxl
```

```
Terminal: Local x +
(venv) MorrisdeMacBook-Pro:HelloWorld morris$
```

