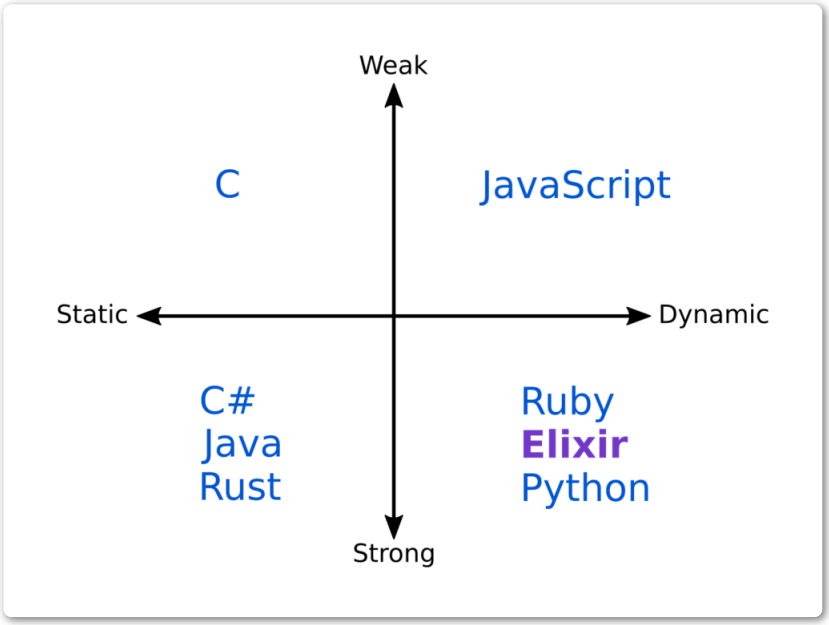
Python是强类型的动态脚本语言



好多人对python到底是强类型语言还是弱类型语言存在误解，其实，是否是强类型语言只需要一句话就可以判别。

> 强类型：不允许不同类型相加。例如：整型+字符串会报类型错误。

```
50+'python'

str(50)+'python'

50+int('5')
```

> 动态：可以不使用数据类型声明，且确定一个变量的类型是在给它赋值的时候来确定一个变量的类型

```
uid=100
type(uid)

uid="一百"
type(uid)
```

> 脚本语言：一般是指解释性语言，运行代码只需要一个解释器，不需要编译器

这也是这段代码 uid 上下类型不同也能正常运行的原因。

```
uid=100
type(uid)

uid="一百"
type(uid)
```

实时效果反馈

1. 以下说法不正确的是:

A 计算机不能直接理解高级语言，只能直接理解机器语言，所以必须要把高级语言翻译成机器语言，计算机才能执行高级语言编写的程序

B 解释性语言在运行程序的时候才会进行翻译

C 编译型语言写的程序在执行之前，需要一个专门的编译过程，把程序编译成机器语言（可执行文件）

D Python是编译型语言

2. 以下哪行代码在运行时会报错：

A `50+'python'`

B `str(50)+'python'`

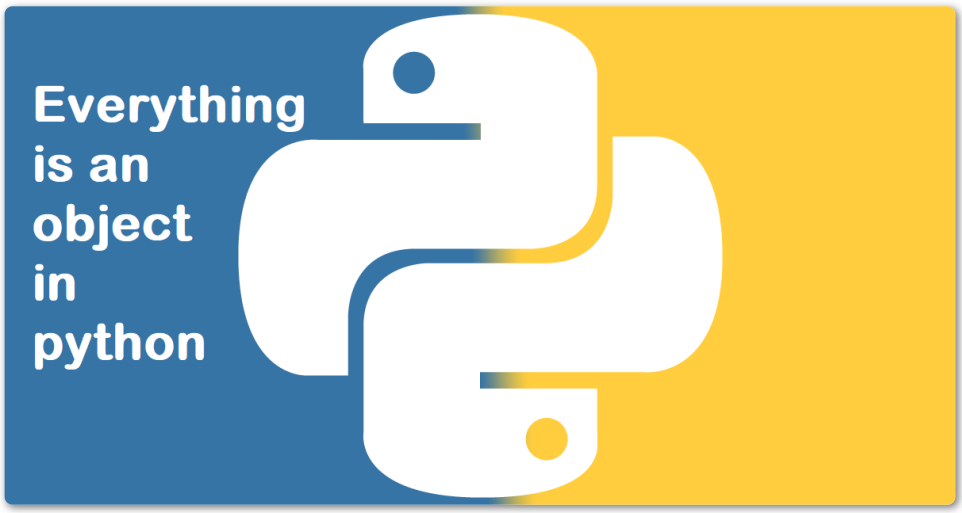
C `50+int('5')`

D `type(50)`

答案

1=>D　　2=>A

Python一切皆对象



Python中，一切皆对象。每个对象由：标识（identity）、类型（type）、value（值）组成。对象的本质就是：一个内存块，拥有特定的值，支持特定类型的相关操作。

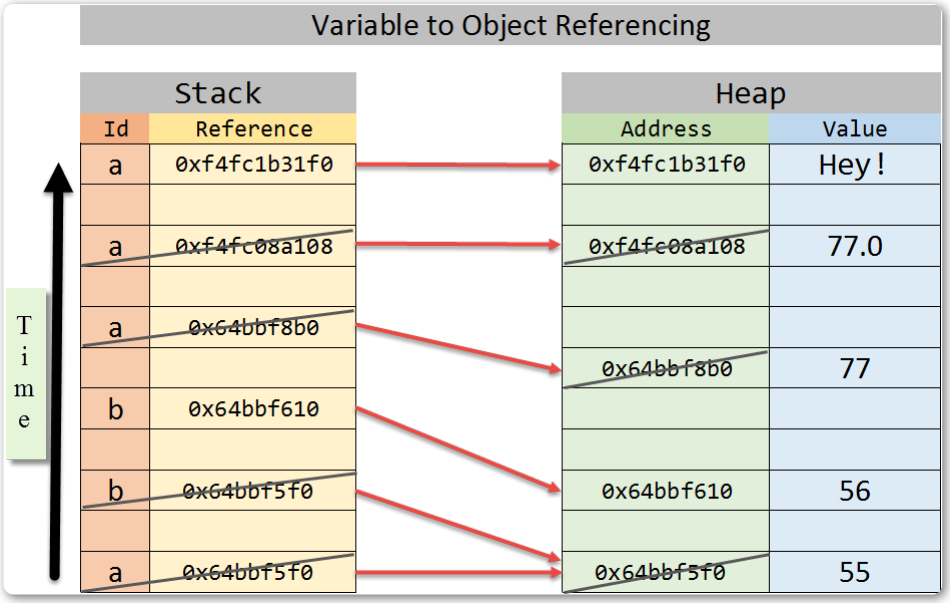
1. 标识用于唯一标识对象，通常对应于对象在计算机内存中的地址。使用内置函数id(obj)可返回对象obj的标识。
2. 类型用于表示对象存储的“数据”的类型。类型可以限制对象的取值范围以及可执行的操作。可以使用type(obj)获得对象的所属类型。
3. 值表示对象所存储的数据的信息。使用print(obj)可以直接打印出值。

```
id(uid)

type(uid)

print(uid)
```

在Python中，变量也成为：对象的引用。因为，变量存储的就是对象的地址。变量通过地址引用了“对象”。

变量位于：栈内存。

对象位于：堆内存。



在Python语言中，声明变量的同时需要为其赋值，毕竟不代表任何值的变量毫无意义，Python中也不允许有这样的变量。

```
#声明变量my_name没有赋值
my_name
print(my_name)#输出my_name报异常
```

实时效果反馈

1. 以下说法不正确的是:

A 变量是到内存空间的一个指针

B 对象是一块内存

C 引用就是自动形成的从变量到对象的指针

D 对象存放于栈内存

2. 以下说法不正确的是：

```
a = 'python'
b = a
```

A a 和 b 是一样的，他们指向同一片内存，b 不过是 a 的别名，是引用

B a 和 b 是不一样的，他们指向不同的内存

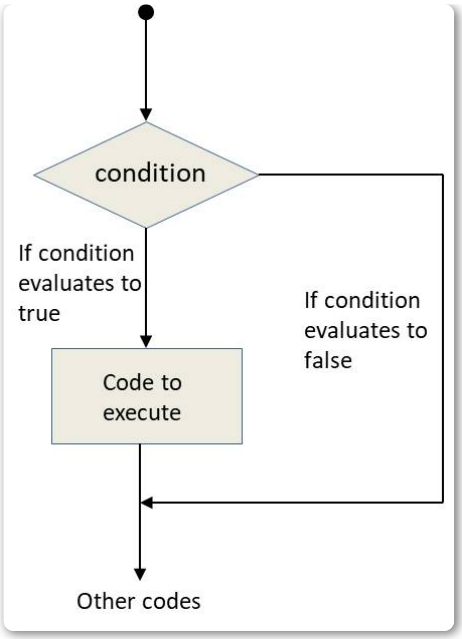
C 我们可以使用 b is a 去判断，返回 True，表明他们地址相同，内容相同

D 可以使用 id() 函数来查看两个列表的地址是否相同

答案

1=>D　　2=>B

Python控制语句



在Python语言底层，会将布尔值True看作1，将布尔值False看作0，尽管从表面上看，True和1、False和0是完全不同的两个值，但实际上，它们是相同的。

单分支控制语句，注意双等号==、缩进（4个空格，不使用{}）等问题

```
uid=None

if uid==0:
    print('root')
```

```
num = input("请输入一个整数：")
if int(num)<10:     #获取的num是字符串，所以使用int()函数进行转换
    print('您输入的整数是:',num)
```

双分支控制语句：

```
if uid==None:
    print('root')
else:
    print('abc')
```

三元运算符（三目运算符），

```
print('root') if uid==None else print('abc')
```

多分支控制语句，

```
score = 88.8
level = int(score%10)
if level >= 10:
    print('Level A+')
elif level==9:
    print('Level A')
elif level==8:
    print('Level B')
elif level==7:
    print('Level C')
elif level==6:
    print('Level D')
else:
    print('Level E')
```

选择结构嵌套，

```
score = 88.8
level = int(score%10)
if level >= 9:
    if level>=10
        print('Level A+')
```

```
    else:
        print('Level A')
else:
    if level==8:
        print('Level B')
    elif level==7:
        print('Level C')
    elif level==6:
        print('Level D')
    else:
        print('Level E')
```

实时效果反馈

1. 如果uid为空值，运行下面代码的结果是:

```
print('root') if uid==None else print('abc')
```

A root

B abc

C 运行不通过

D 以上答案都不是

2. 运行下面代码的结果是:

```
print("True+False+20的计算结果:",True+False+20)
```

A True+False+20的计算结果: 20

B True+False+20的计算结果: 19

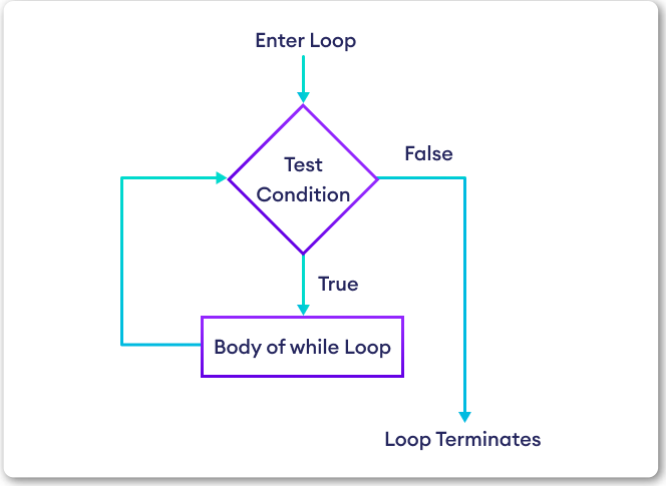
C True+False+20的计算结果: 21

D Boolean类型和Integer类型直接相加会报错

答案

1=>A　　2=>C

while循环语句



当条件满足的时候，里面的循环体会执行，执行之后会再去进行条件判断，满足就会再次执行，如果一直满足条件，就一直循环反复的执行

```
count = 0
while count<=9:
    print(count)
    count += 1
```

可以打印的时候，让每一次的时候不自动换行

```
count = 0
while count<=9:
    print(count, end=' ')
    count += 1
print()
print('Done')
```

拼接字符串，最后一起打印

```
result=''
count = 0
while count<=9:
    result+=(str(count)+' ')
    count += 1
print(result)
```

实时效果反馈

1. 下面代码做的事情是:

```
num = 0
while num<=10:
    print(num)
    num += 1
```

A 利用for循环打印从1-10的数字

B 利用for循环打印从0-10的数字

C 利用while循环打印从1-10的数字

D 利用while循环打印从0-10的数字

2. 运行下面代码的结果是：

```
num = 1
sum_even = 0
while num<=100:
    if num%2==0:
        sum_even += num
    num += 1
print(sum_even)
```

A 计算1-100之间所有数的累加和

B 计算1-100之间奇数的累加和

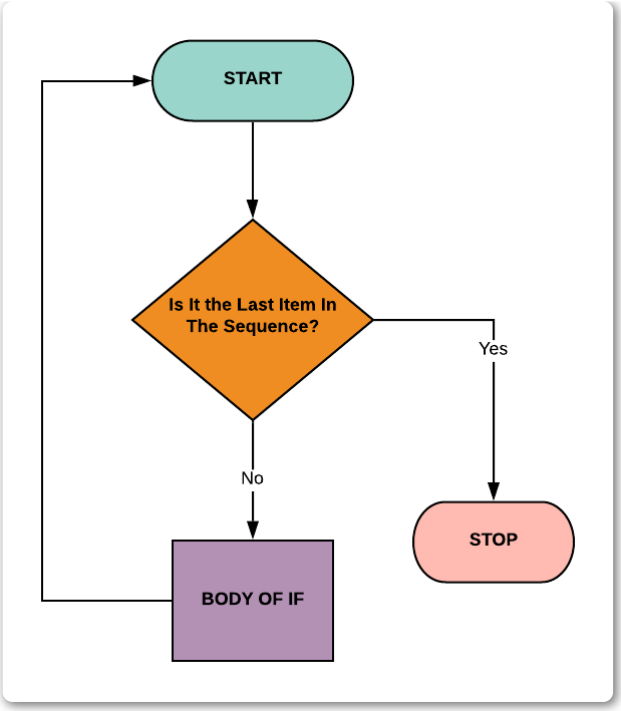
C 计算1-100之间偶数的累加和

D 以上都不是

答案

1=>D　　2=>C

for循环语句



for循环用于遍历一个集合，每次循环，会从集合中取得一个元素，并执行一次代码块，直到集合中所有的元素都获取，for循环才结束。

Python可以遍历的对象有序列(字符串、列表、元组)、字典、迭代器对象(iterator)、生成器函数文件对象。前面已经学习了字符串，通过循环来遍历字符串。

```
names = ['Tom', 'Peter', 'Jerry', 'Jack']
for name in names:
    print(name)
```

range(start, end [,step])

```
list(range(4))
```

```
for i in range(len(names)):
    print(names[i])
```

实时效果反馈

1. 下面代码做的事情是:

```
for i in range(3,10,2)
```

A 产生序列：0 1 2 3 4 5 6 7 8 9

B 产生序列：3 4 5 6 7 8 9

C 产生序列：3 5 7 9

D 以上都不是

2. 运行下面代码的结果是:

```
sum_odd = 0
for num in range(101):
    if num%2==1:
        sum_odd += num
print(sum_odd)
```

A 计算1-100之间所有数的累加和

B 计算1-100之间奇数的累加和

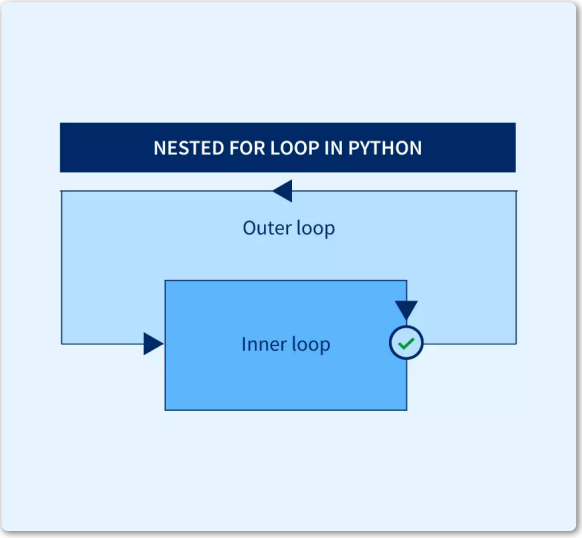
C 计算1-100之间偶数的累加和

D 以上都不是

答案

1=>C　　2=>B

循环嵌套（多重循环）



一个循环体内可以嵌入另一个循环，一般称为“嵌套循环”，或者“多重循环”。

> 每天吃饭3顿饭可以是循环嵌套
>
> 每天学习可以是循环嵌套
>
> ……

```
for year in years:
    for day in days:
        for eat in ['breakfast', 'lunch', 'dinner']
            print("must have a meat")
```

```
for epoch in epochs:
    for batch in batches:
        print("机器学习一个批次的数据")
```

打印九九乘法表

```
j=1
while j<=9:
    i=1
    while i<=j:
        print('%d*%d=%d'%(j,i,i*j), end='\t')
        i+=1
    print()
    j+=1
```

```
for j in range(1, 10):
    for i in range(1, j+1):
        print('%d*%d=%d'%(j,i,i*j), end='\t')
    print()
```

实时效果反馈

1. 下面说法正确的是:

A 嵌套循环只能是双层for循环

B 嵌套循环只能是for循环

C 嵌套循环只能是while循环

D 嵌套循环可以是for循环也可以是while循环

2. 运行下面代码的结果是:

```python
list01 = [1,5,7,4,5,0,5,6,9]
for r in range(len(list01) - 1):
    for c in range(r + 1, len(list01)):
        if list01[r] > list01[c]:
            # 交换
            list01[r], list01[c] = list01[c], list01[r]

print(list01)
```

A 对列表进行正序排序

B 对列表进行倒序排序

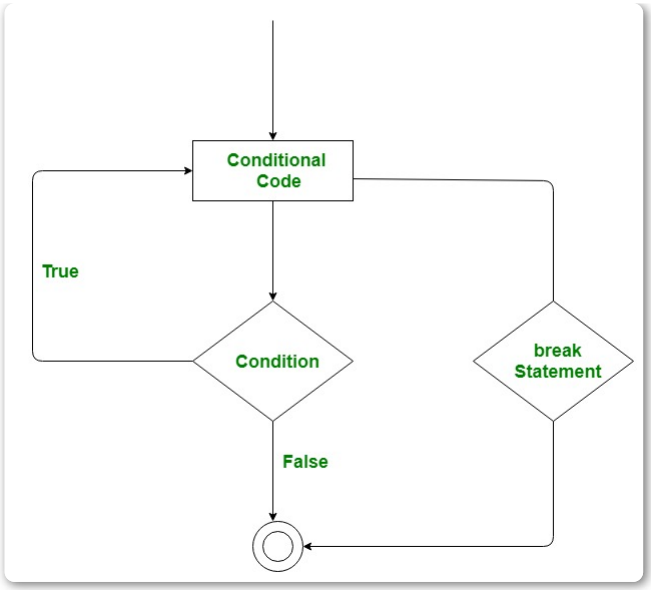
C 查看列表中是否具有相同元素

D 以上都不是

答案

1=>D　　2=>A

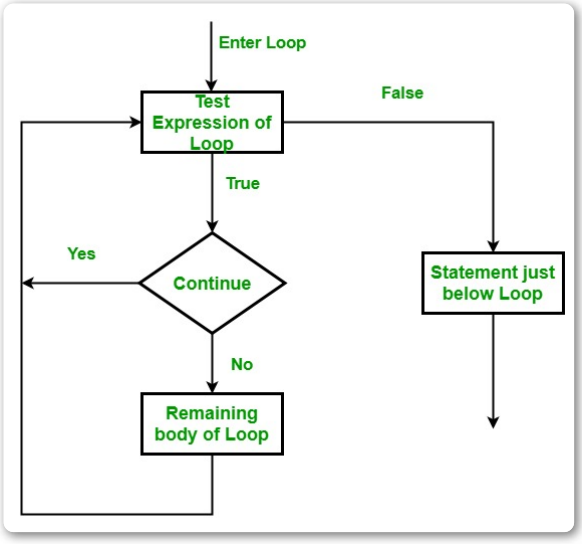
Break 与 Continue 关键字



break语句可用于while和for循环，用来结束整个循环。当有嵌套循环时，break语句只能跳出最近一层的循环。

```
for i in range(10):
    if i==5:
        break
    print(i)
```

```
names = ['Tom', 'Peter', 'Jerry', 'Jack']
for i in range(len(names)):
    if i>=2:
        break
    print(names[i])
```



与break语句对应的还有另一个continue语句，与break语句不同的是，continue用于结束本次循环，继续下一次。多个循环嵌套时，continue也是应用于最近的一层循环。而break语句用来彻底退出循环。

```
for i in range(10):
    if i%2==0:
        continue
    print(i)
```

```
names = ['Tom', 'Peter', 'Jerry', 'Jack']
for i in range(len(names)):
    if i<2:
        continue
    print(names[i])
```

实时效果反馈

1. 下面代码的逻辑是:

```
while True:
    a = input("请输入一个字符（输入Q或q结束）")
    if a.upper()=='Q':
        print("循环结束，退出")
        break
    else:
        print(a)
```

A 程序是一个死循环，无论如何都无法结束

B 程序是一个死循环，输入"Q"将通过break语句跳出死循环

C 程序是一个死循环，输入"Q"将通过continue语句跳出死循环

D 以上都不是

2. 运行下面代码的结果是:

```
state = False
list01 = [1,5,7,4,5,0,5,6,9]
# 取出前面的元素
for r in range(len(list01)-1):
    #取出后面的元素
    for c in range(r+1,len(list01)):
        if list01[r] == list01[c]:
            state = True
            break
    if state:
        break
if state:
    print("True")
else:
    print("False")
```

A 对列表进行正序排序

B 对列表进行倒序排序

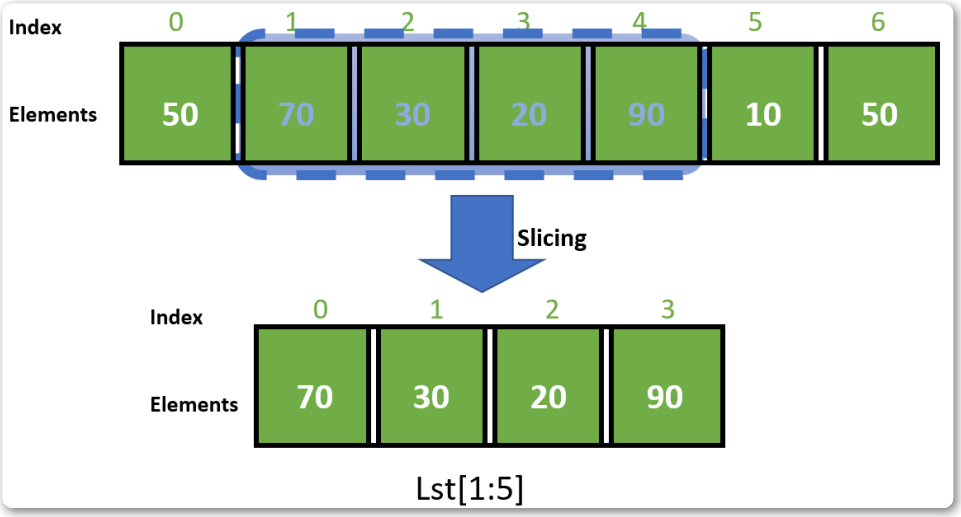
C 查看列表中是否具有相同元素

D 以上都不是

答案

1=>B　　2=>C

Python切片操作

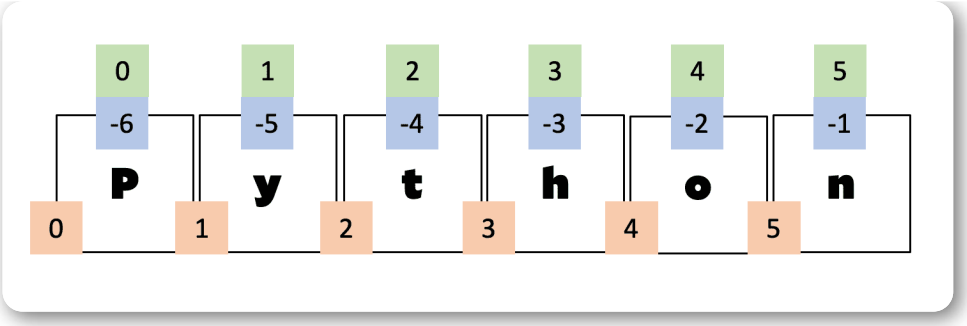


切片是Python序列及其重要的操作，适用于列表、元组、字符串等等。

> 列表[起始偏移量start : 终止偏移量end [:步长step]]

切片操作是从列表A中获取一个子列表B。列表A可以称为父列表。从A中获取B，需要指定B在A中的开始索引和结束索引，因此，切片操作需要指定两个索引。

```
pystr='Python'
pystr[:]

pystr[2:]

pystr[:5]

pystr[1:5]

pystr[1:5:2]
```




```
pystr[:-3]

pystr[-5:-3]

pystr[::-1]

pystr[::-2]

pystr[1:-3]
```

机器学习中切片可用于分割数据集为训练集和测试集，假如总共有10000条样本数据：

```
TrainSet=Data[:8000]
TestSet=Data[8000:]

batch_size=100
num_batches=len(TrainSet)/batch_size

for epoch in epoches:
    for index in num_batches:
        batch = TrainSet[batch_size*index:(batch_size+batch_size*index)]
        # 接下来应用这个批次的数据去训练模型
```

要加个 range ()

实时效果反馈

1. 运行下面代码的结果是:

```
a=[10,20,30,40,50,60,70]
print(a[1:6:2])
```

A [10, 20, 30, 40, 50, 60, 70]

B [20, 30, 40, 50, 60, 70]

C [20, 40, 60]

D 以上都不是

2. 运行下面代码的结果是:

```
a=[10,30,20,50,40,70,60]
print(a[::-1])
```

A 对列表进行正序排序

B 对列表进行倒序排序

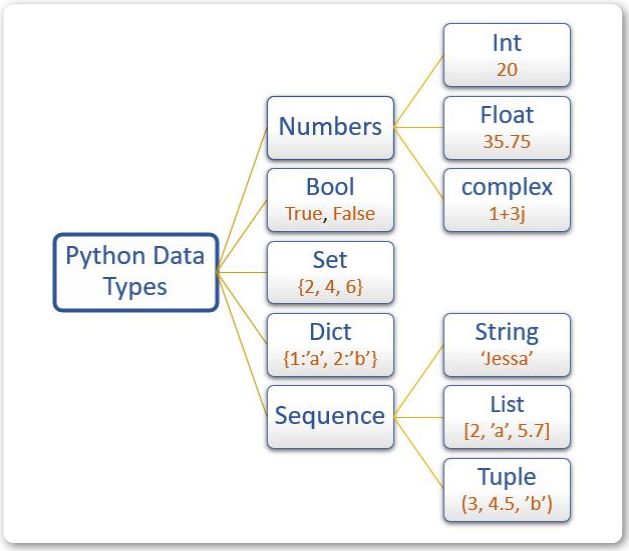
C 对列表进行反向提取

D 以上都不是

答案

1=>C　　2=>C

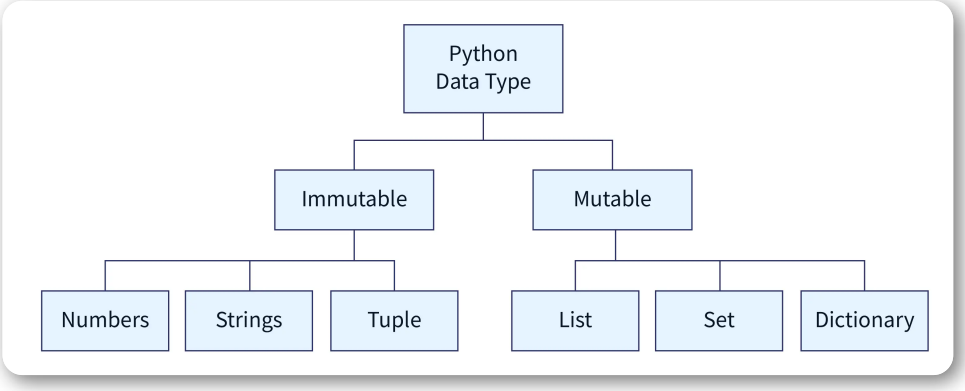
Python常用数据类型



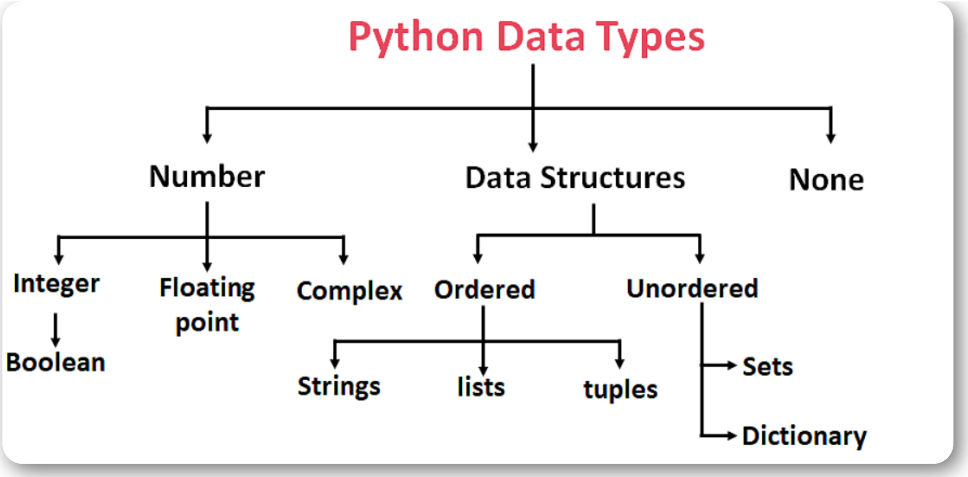
> type()函数可以查看对象数据类型

数据类型	细分	
Number	Integer（整型）	
	Float（双精度浮点型）	
	Complex（复数型）	
Boolean	True或False（布尔型）	
String	零个或多个字符组成的有限序列（字符串型）	Sequence
Tuple	内部元素不可修改（元组型）	Sequence
List	但内部元素可以修改（列表型）	Sequence
Set	内部元素相互之间无序的一组对象（集合型）	
Dictionary	拥有键/值对的特殊列表，形式key:value（字典型）	

可变类型与不可变类型



有序集合与无序集合



常用类型之间互相转换的函数

类型转换	
int(x)	将x转换为一个整数
float(x)	将x转换到一个浮点数
str(x)	将对象 x 转换为字符串
eval(str)	用来计算在字符串中的有效Python表达式,并返回一个对象
tuple(s)	将序列 s 转换为一个元组
list(s)	将序列 s 转换为一个列表
set(s)	转换为可变集合
dict(d)	创建一个字典。d 必须是一个序列 (key,value)元组
chr(x)	将一个整数转换为一个字符
unichr(x)	将一个整数转换为Unicode字符
ord(x)	将一个字符转换为它的整数值

```
eval('5+10')

a=10
b=20
eval('a+b')
```

实时效果反馈

1. 用python如何表示常量浮点值 $3.2 × 10^{-12}$:

A 3.2e-12

B 3.2e+12

C 3.2 * 10 ** -12

D 以上都不是

2. 运行下面代码的结果是:

```
print(type({}))
```

A <class 'tuple'>

B <class 'list'>

C <class 'set'>

D <class 'dict'>

答案

1=>A　　2=>D