day11

今日内容

- 函数小高级
- lambda 表达式
- 内置函数

内容回顾

1. 函数基本结构
2. 参数
    - 形参
        - 基本参数：def func(a1,a2):pass
        - 默认值：def func(a1,a2=123):pass 注意： 默认值如果是不可变类型，随便玩。 可变类型：有坑
        - 无敌：*args，**kwargs
    - 实参
        - 位置传参
        - 关键字传参
3. 返回值
    - 默认返回：None
    - 特殊情况

```
def func():
    return 1,2,32,4
```

4. 作用域
    - 全局和局部
    - 自己有找自己，自己没有找父级。
    - 子作用域只能读取或修改父级的值，不能重新赋值。

```
NUM = [1,2,3]
def func():
    # NUM.append(99) # 可以修改父级的值
    global NUM
    NUM = 23123
func()
```

5. 函数的嵌套

```python
def func():
    name = 'oldboy'
    def inner():
        print(name)
    name = 'alex'
    inner()
    print(name)
func()
```

内容详细

1. 函数小高级

```python
a = 123
name = '老男孩好'
nums = [11,22,33,4]
data = nums

def func():
    pass # 100行代码

# func = 函数
```

1.1 函数名当作变量来使用

```python
def func():
    print(123)

v1 = func

func()
v1 ()
```

```python
def func():
    print(123)

func_list = [func, func, func]
# func_list[0]()
# func_list[1]()
# func_list[2]()
for item in func_list:
    v = item()
    print(v)
```

```
def func():
    print(123)

def bar():
    print(666)

info = {'k1': func, 'k2': bar}

info['k1']()
info['k2']()
```

混淆你

```
def func():
    return 123

func_list1 = [func,func,func]
func_list2 = [func(),func(),func()]

print(func_list1)
print(func_list2)

info = {
    'k1':func,
    'k2':func(),
}

print(info)
```

1.2 函数可以当作参数进行传递

```
def func(arg):
    print(arg)

func(1)
func([1,2,3,4])

def show():
    return 999
func(show)
```

```
def func(arg):
    v1 = arg()
    print(v1)

def show():
    print(666)

func(show)
```

```python
def func(arg):
    v1 = arg()
    print(v1)

def show():
    print(666)

result = func(show)
print(result)
```

```python
# 面试题

def func():
    print('花费查询')


def bar():
    print('语音沟通')


def base():
    print('xxx')


def show():
    print('xxx')


def test():
    print('xxx')

info = {
    'f1': func,
    'f2': bar,
    'f3':base,
    'f4':show,
    'f5':test
}
choice = input('请选择要选择功能：')
function_name = info.get(choice)
if function_name:
    function_name()
else:
    print('输入错误')
```

2. lambda表达式

用于表示简单的函数。

```python
# 三元运算，为了解决简单的if else的情况，如：
if 1 == 1:
    a = 123
else:
    a = 456

a =  123  if 1 == 1 else 456

# lambda表达式，为了解决简单函数的情况，如：
def func(a1,a2):
    return a1 + 100

func = lambda a1,a2: a1+100
```

```python
func1 = lambda : 100

func2 = lambda x1: x1 * 10

func3 = lambda *args,**kwargs: len(args) + len(kwargs)

DATA = 100
func4 = lambda a1: a1 + DATA
v = func4(1)
print(v)


DATA = 100
def func():
    DATA = 1000
    func4 = lambda a1: a1 + DATA
    v = func4(1)
    print(v)
func()


func5 = lambda n1,n2: n1 if n1 > n2 else n2
v = func5(1111,2)
print(v)
```

练习题

```python
# 练习题1
USER_LIST = []
def func0(x):
    v = USER_LIST.append(x)
    return v

result = func0('alex')
print(result)


# 练习题2
```

```python
def func0(x):
    v = x.strip()
    return v

result = func0(' alex ')
print(result)

############## 总结：列表所有方法基本上都是返回None；字符串的所有方法基本上都是返回新值
#################
# 练习题3
USER_LIST = []
func1 = lambda x: USER_LIST.append(x)

v1 = func1('alex')
print(v1)
print(USER_LIST)

# 练习题4
func1 = lambda x: x.split('l')

v1 = func1('alex')
print(v1)

# 练习题5
func_list = [lambda x:x.strip(), lambda y:y+199,lambda x,y:x+y]

v1 = func_list[0]('alex ')
print(v1)

v2 = func_list[1](100)
print(v2)

v3 = func_list[2](1,2)
print(v3)
```

3. 内置函数

- 自定义函数
- 内置函数
  - 其他
    - len
    - open
    - range
    - id
    - type
  - 输入输出
    - print
    - input
  - 强制转换

- dict()
- list()
- tuple()
- int()
- str()
- bool()
- set()

- 数学相关
  - abs，绝对值

```
v = abs(-1)
print(v)
```

  - float，转换成浮点型（小数）

```
v = 55
v1 = float(55)
print(v1)
```

  - max，找到最大值

```
v = [1,2,311,21,3,]
result = max(v)
print(result)
```

  - min，找最小值

```
v = [1,2,311,21,3,]
result = min(v)
print(result)
```

  - sum，求和

```
v = [1,2,311,21,3,]
result = sum(v)
print(result)
```

  - divmod，两数相除的商和余数

```
a,b = divmod(1001,5)
print(a,b)
```

```
# 练习题　请通过分页对数据进行展示
"""
要求:
    每页显示10条数据
    让用户输入要查看的页面：页码
"""
```

```
USER_LIST = []
for i in range(1,836):
    temp = {'name':'你少妻-%s' %i,'email':'123%s@qq.com' %i }
    USER_LIST.append(temp)

# 数据总条数
total_count = len(USER_LIST)

# 每页显示10条
per_page_count= 10

# 总页码数
max_page_num,a = divmod(total_count,per_page_count)
if a>0:
    max_page_num += 1

while True:
    pager = int(input('要查看第几页: '))
    if pager < 1 or pager > max_page_num:
        print('页码不合法，必须是 1 ~ %s' %max_page_num )
    else:
        """
        # 第1页: USER_LIST[0:10] -> 0123456789
        # 第2页: USER_LIST[10:20]
        # 第3页: USER_LIST[20:30]
        ...
        """
        start = (pager-1) * per_page_count
        end = pager * per_page_count
        data = USER_LIST[start:end]
        for item in data:
            print(item)
```

- 进制转换相关
  - bin，将十进制转化成二进制

```
num = 13
v1 = bin(num)
print(v1)
```

  - oct，将十进制转换成八进制

```
num = 8
v1 = oct(num)
print(v1)
```

  - int，将其他进制转化成十进制

```
# 二进制转化成十进制
v1 = '0b1101'
```

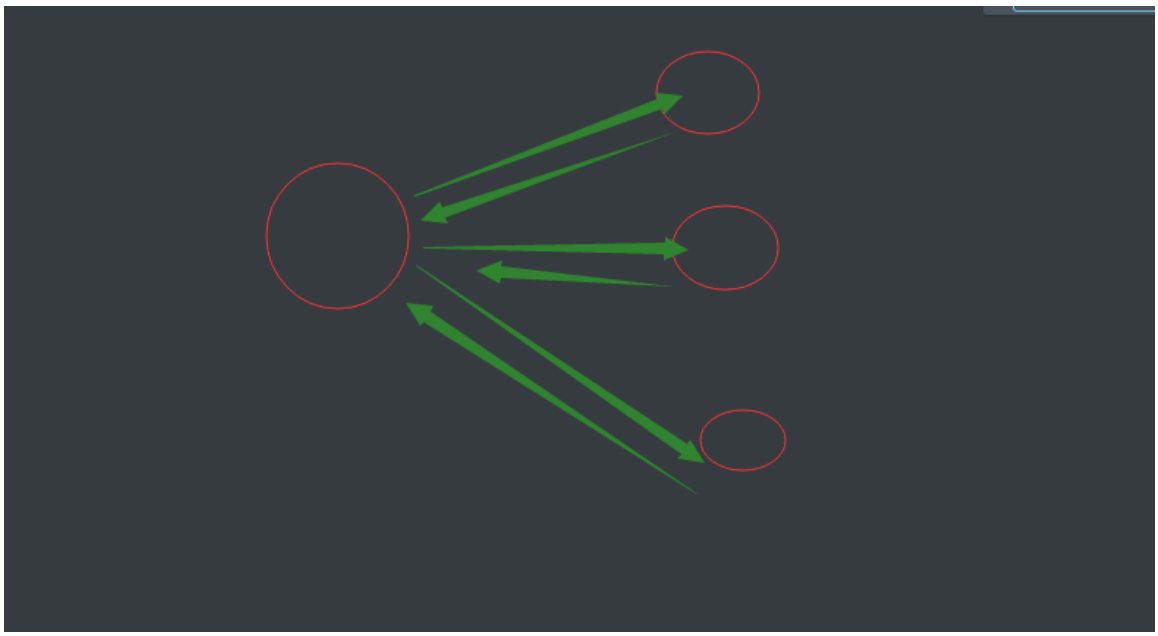
```
result = int(v1,base=2)
print(result)

# 八进制转化成十进制
v1 = '0o1101'
result = int(v1,base=8)
print(result)

# 十六进制转化成十进制
v1 = '0x1101'
result = int(v1,base=16)
print(result)
```

- hex，将十进制转换成十六进制

```
num = 16
v1 = hex(num)
print(v1)
```



- 面试题

```
# 1字节等于8位
# IP: 192.168.12.79  ->  001010010 . 001010010 . 001010010 . 001010010

# 1. 请将 ip = "192.168.12.79" 中的每个十进制数转换成二进制并通过,连接起来生成一个新的字符串。
ip = "192.168.12.79"
ip_list = ip.split('.') # ['192','168','12','79']
result = []
for item in ip_list:
    result.append(bin(int(item)))
print(','.join(result))
```

```
# 2. 请将 ip = "192.168.12.79" 中的每个十进制数转换成二进制:
#          00101001000010100100010100100010100010 -> 十进制的值。

# 3232238671
```

今日总结

- 函数当作是一个变量：参数传值 / 当元素嵌套到字典列表中。 5*
- lambda 表达式 3*
- 内置函数 3*