

# 暨南大学本科实验报告专用纸

课程名称：Python 程序设计实验

实验项目名称：循环与选择结构 指导教师：林聪

实验项目编号：实验 03 实验项目类型：验证性实验

实验地点：实 C302

学生姓名：陈旭天 学号：2021100733

学院：智能科学与工程学院 专业：人工智能

实验时间：2022 年 3 月 2 日下午--2022 年 3 月 2 日下午



## 一、实验目的

熟悉使用 python 中的 for 循环、while 循环和 if 选择结构，并利用简易的代码框架来测试使用循环与选择结构和相关的关键字与函数，完成思考题。

## 二、实验原理

## 三、实验结果

### For 循环

```
users = {
    'aeinstein':{
        'first':'albert',
        'last':'einstein',
        'location':'princeton',
    },
    'mcurie':{
        'first':'marie',
        'last':'curie',
        'location':'paris'
    },
}
```

```
for username,user_info in users.items():
    print(f"\nUsername:{username}")
    full_name = f"{user_info['first']}{user_info['last']}"
    location = user_info['location']

    print(f"\tFull name:{full_name.title()}")
    print(f"\tLocation:{location.title()}")
```

```
Username:aeinstein
    Full name:Alberteinstein
    Location:Princeton

Username:mcurie
    Full name:Mariecurie
    Location:Paris
```

### While 循环

```

height = input("How tall are you?")
height = int(height)
while height<=170:
    print("Eat more,little guy!")
    break

```

```

PS C:\Users\36126\learning_log> python -u "c:\Users\36126\learning_log\tempCodeRunnerFile.e.python"
How tall are you?160
Eat more,little guy!
PS C:\Users\36126\learning_log> python -u "c:\Users\36126\learning_log\tempCodeRunnerFile.e.python"
How tall are you?180
PS C:\Users\36126\learning_log> █

```

Break 跳过本轮循环，并结束本次循环

Continue 跳过本次循环，但不会结束循环本身

下面用 python 代码来实现寻找素数的算法题

```

n = input("Please enter a number:\n")
n = int(n)
k = 0
for i in range(n):
    m = i+1
    for j in range(i):
        l = j+1
        if m%l==0&l!=1:
            continue
        elif j>i/2:
            print(f"{str(m)} is a prime number.")
            k += 1
            break
print(f"{str(k)}")

```

```

pelase enter a number:
500
2 is a prime number.
3 is a prime number.
5 is a prime number.
7 is a prime number.
11 is a prime number.
13 is a prime number.
17 is a prime number.
19 is a prime number.
23 is a prime number.
29 is a prime number.
31 is a prime number.
37 is a prime number.

```

## 迭代

练习使用 `zip()`, `enumerate()`, `range()` 函数, 要注意的是, `zip()` 函数操作两个可迭代对象最后生成的是一个迭代器, 将其 `print` 之后只能得到它的地址, 需要用 `list` 函数将其生成一个列表才能打印其内容。Input 函数输入的内容数据类型是字符串。

```
1 a = input("please enter the length of list")
2 a = int(a)
3 list1 = []
4 for i in range(a):
5     n = input("please enter yuansu in list:")
6     list1.append(n)
7
8 list2 = [4,5,6]
9 m = list(zip(list1,list2))
10 print(m)
11
12 for i in enumerate(m):
13     print("现在迭代第{}个元素。".format(i[0]+1))
14     print(i)
15
16 for i in range(len(m)):
17     print(f"现在迭代第{str(i+1)}个元素")
```

```
PS C:\Users\36126\c语言> python -u "c:\Users\36126\learning_log\tempCodeRunnerFile.pytho
n"
please enter the length of list3
please enter yuansu in list:1
please enter yuansu in list:2
please enter yuansu in list:3
[('1', 4), ('2', 5), ('3', 6)]
现在迭代第1个元素。
(0, ('1', 4))
现在迭代第2个元素。
(1, ('2', 5))
现在迭代第3个元素。
(2, ('3', 6))
现在迭代第1个元素
现在迭代第2个元素
现在迭代第3个元素
```

## ReLU 函数的实现

```
def leaky_relu_v1(x):
    if x>0:
        out = x
    else:
        out = 0.2*x
    return out

n = input("please enter a number:\n")
n = int(n)
m = leaky_relu_v1(n)
print(str(m))
```

```

PS C:\Users\36126\c语言> python -u "c:\Users\36126\learning_log\tempCodeRunnerFile.pytho
n"
please enter a number:
7
7
PS C:\Users\36126\c语言> python -u "c:\Users\36126\learning_log\tempCodeRunnerFile.pytho
n"
please enter a number:
-1
-0.2

```

V2 不用 if-else 则是两个 if 判断结构

```

1  def leaky_relu_v2(x,a=0.2):
2      if x>0:
3          out = x
4      if x<=0:
5          out = a*x
6
7      return out
8
9  n = input("please enter a number:\n")
10 n = int(n)
11 m = leaky_relu_v2(n)
12 print(str(m))

```

问题 2 输出 调试控制台 终端

```

PS C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
please enter a number:
0
0.0
PS C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
please enter a number:
100000
100000
PS C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
please enter a number:
-100000
-20000.0
PS C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
please enter a number:
1
1
PS C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
please enter a number:
-1
-0.2
PS C:\Users\36126>

```

计算两种激活函数运行 2000000 次的时间

```
import time

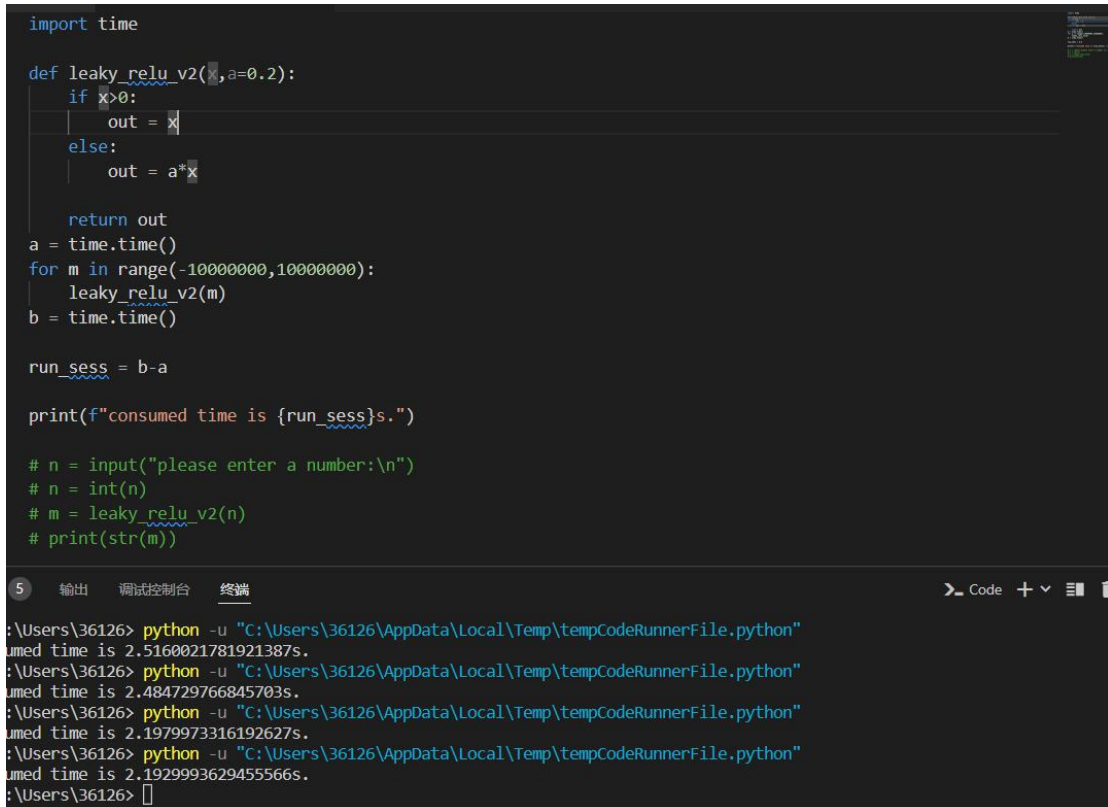
def leaky_relu_v2(x,a=0.2):
    if x>0:
        out = x
    if x<=0:
        out = a*x

    return out
a = time.time()
for m in range(-10000000,10000000):
    leaky_relu_v2(m)
b = time.time()

run_sess = b-a

print(f"consumed time is {run_sess}s.")
```

上图是双 if 结构消耗的时间



```
import time

def leaky_relu_v2(x,a=0.2):
    if x>0:
        out = x
    else:
        out = a*x

    return out
a = time.time()
for m in range(-10000000,10000000):
    leaky_relu_v2(m)
b = time.time()

run_sess = b-a

print(f"consumed time is {run_sess}s.")

# n = input("please enter a number:\n")
# n = int(n)
# m = leaky_relu_v2(n)
# print(str(m))
```

5 输出 调试控制台 终端

```
C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
consumed time is 2.5160021781921387s.
C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
consumed time is 2.484729766845703s.
C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
consumed time is 2.1979973316192627s.
C:\Users\36126> python -u "C:\Users\36126\AppData\Local\Temp\tempCodeRunnerFile.python"
consumed time is 2.1929993629455566s.
C:\Users\36126>
```

上图是 if-else 结构消耗的时间

不难看出，if-else 结构比没有 else 的结构大约可以快 0.3s 左右的时间

## #思考题

1. 为什么循环结构中不应改变正在迭代的序列的长度? 能否用代码写出一个反例?  
改变序列的长度会导致循环容易导致程序无法跳出循环, 同时也会导致循环的结果出错, 并且这种错误不会报错, 非常隐蔽难以发现, 不利于程序的测试与维护。

2. 为什么设计语言需要循环和选择结构?

使用循环、选择、顺序就可以实现简单或是复杂的所有算法。同时循环和选择结构更有助于程序的结构化编程。按结构化程序设计方法设计出的程序优点是: 结构良好、各模块间的关系清晰简单、每一模块内都由基本单元组成。这样设计出的程序清晰易读, 可理解性好, 容易设计, 容易验证其正确性, 也容易维护。同时, 由于采用了“自顶向下、逐步细化”的实施方法, 能有效地组织人们的智力, 有利于软件的工程化开发。

## 四、实验总结

这次实验对 python 的选择和循环结构进行了学习, 同时也实现了一些简单的 DNN 激活函数, 同时也学会用 time 模块中的 time () 函数记录程序运行的时间戳, 对不同结构的函数运行效率进行了对比, 同时也深入思考了程序语言中循环和选择的意义, 对于迭代器也有了更深的认知。