

课程报告三：Python 与命令行环境

杜培绪

2024 年 9 月 11 日

题目 1.

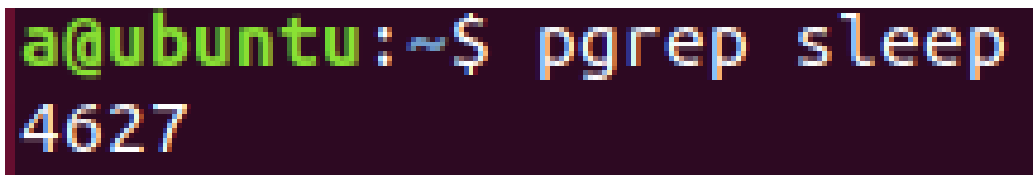
但我们其实有更好的方法来做这件事。在终端中执行 `sleep 10000` 这个任务。然后用 `Ctrl-Z` 将其切换到后台并使用 `bg` 来继续允许它。现在，使用 `pgrep` 来查找 `pid` 并使用 `pkill` 结束进程而不需要手动输入 `pid`。

解决方法：

先使用

`sleep 10000`

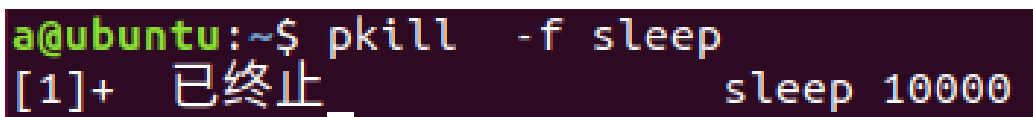
进入睡眠状态，输入 `ctrl+z` 和 `bg`，将其切换到后台并使用 `bg` 来继续允许它。接下来使用 `pgrep` 来允许查找 `pid`。下一步使用 `pkill` 结束进程，这里



```
a@ubuntu:~$ pgrep sleep
4627
```

图 1: 查找 pid

不需要手动输入 `pid`。



```
a@ubuntu:~$ pkill -f sleep
[1]+ 已终止 sleep 10000
```

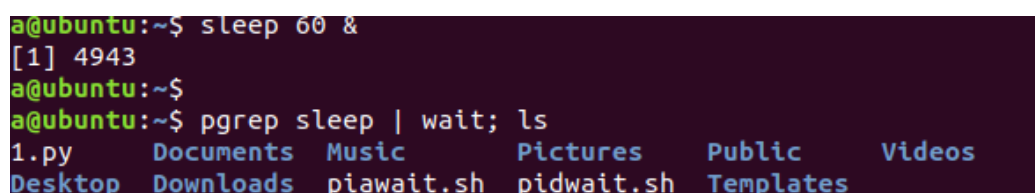
图 2: `pkill` 指令

题目 2.

如果您希望某个进程结束后再开始另外一个进程, 应该如何实现呢? 在这个练习中, 我们使用 `sleep 60 &` 作为先执行的程序。一种方法是使用 `wait` 命令。尝试启动这个休眠命令, 然后待其结束后再执行 `ls` 命令。

解决方法:

使用 `wait` 命令, 待睡眠结束后执行 `ls` 命令。



```
a@ubuntu:~$ sleep 60 &
[1] 4943
a@ubuntu:~$
a@ubuntu:~$ pgrep sleep | wait; ls
1.py      Documents Music      Pictures  Public    Videos
Desktop   Downloads piawait.sh pidwait.sh Templates
```

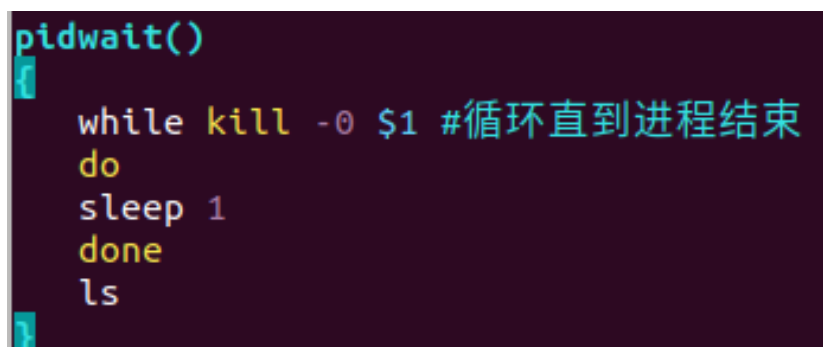
图 3: `wait` 命令

题目 3.

编写一个 `bash` 函数 `pidwait`, 它接受一个 `pid` 作为输入参数, 然后一直等待直到该进程结束。

解决方法:

编写 `pidwait.sh` 代码如下: 运行后发现能够正确允运行。



```
pidwait()
{
    while kill -0 $1 #循环直到进程结束
    do
        sleep 1
    done
    ls
}
```

图 4: `pidwait.sh` 代码

题目 4.

学习如何使用 tmux 实现终端多路复用。

解决方法:

首先在 Linux 系统中下载 tmux, 再使用 tmux 打开界面。学习 tmux 需要掌握快捷键的用法。按下 ctrl+ “水平分割, ctrl+% 为垂直分割, ctrl+ 方向键则是再多个窗口间移动。此外还有其他命令, 例如 ctrl+d 关闭 tmux, 创建新窗口、跳转窗口等。

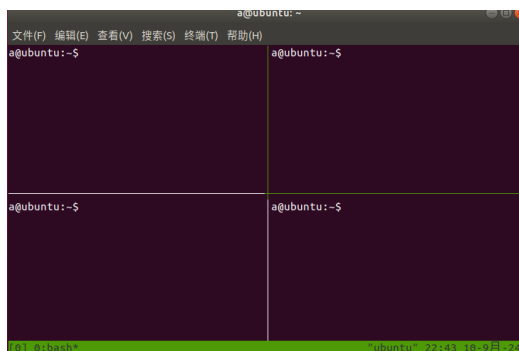


图 5: tmux 窗口

题目 5.

创建一个 dc 别名, 它的功能是当我们错误的将 cd 输入为 dc 时也能正确执行。

解决方法:
$$\text{alias dc = cd}$$

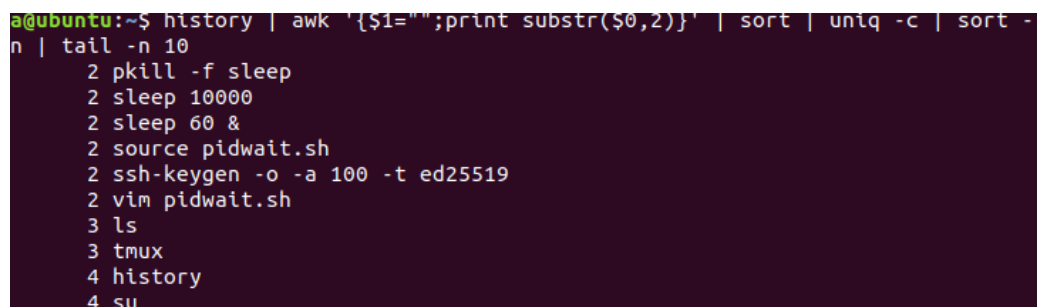
使用 alias 命令创建别名, 使 dc 和 cd 有相同的效果。

题目 6.

执行 `history | awk '{ $1="" ; print substr($0,2)}' | sort | uniq -c | sort -n | tail -n 10` 来获取您最常用的十条命令，尝试为它们创建别名。

解决方法：

执行该命令，显示出最常用的十条命令。接下来尝试为 `sleep` 创建别名，



```

a@ubuntu:~$ history | awk '{ $1="" ; print substr($0,2)}' | sort | uniq -c | sort -n | tail -n 10
  2 pkill -f sleep
  2 sleep 10000
  2 sleep 60 &
  2 source pidwait.sh
  2 ssh-keygen -o -a 100 -t ed25519
  2 vim pidwait.sh
  3 ls
  3 tmux
  4 history
  4 su
  
```

图 6: 十条最常用的命令

别名设置为 `mysleep`，之后使用这两条命令，发现有相同的效果。



```

25 alias sleep=mysleep
  
```

图 7: `sleep` 的别名

题目 7.

编写一个程序，找到 1000 以内所有能被 7 整除但不能被 4 整除的数字。

解决方法：

使用 `for` 循环遍历 1000 以内的数字，若满足条件则保存下来。在 `python` 的 `for` 循环，可以使用 `range` 语句来表示范围。代码如下所示：

```
l=[]
for i in range(0, 1000):
    if (i%7==0) and (i%4!=0):
        l.append(str(i))

print (' , '.join(l))
```

图 8: 2.py 代码

运行代码，返回了所有满足条件的数字。

```
a@ubuntu:~$ python 2.py
7,14,21,35,42,49,63,70,77,91,98,105,119,126,133,147,154,161,175,182,189,203,210,
217,231,238,245,259,266,273,287,294,301,315,322,329,343,350,357,371,378,385,399,
406,413,427,434,441,455,462,469,483,490,497,511,518,525,539,546,553,567,574,581,
595,602,609,623,630,637,651,658,665,679,686,693,707,714,721,735,742,749,763,770,
777,791,798,805,819,826,833,847,854,861,875,882,889,903,910,917,931,938,945,959,
966,973,987,994
```

图 9: 2.py 运行结果

题目 8.

编写一段代码，返回给定参数的阶乘。

解决方法：

计算阶乘可以使用递归的算法。在 python 中读入参数可以使用 input 函数。编写完毕后确定程序能够正常运行。

```
def fact(x):  
    if x == 0:  
        return 1  
    return x * fact(x - 1)  
x=int(input())  
print (fact(x))
```

图 10: 3.py 代码

```
a@ubuntu:~$ python 3.py  
10  
3628800
```

图 11: 3.py 运行结果

题目 9.

python 中的列表与 C 语言中的数组有什么区别?

解决方法:

python 中的列表可以存放不同数据类型的数据, 而 C 语言中只能储存一种特定的数据。例如

```
list = ['name', 'sex', 10, 2024]
```

此外, 这种列表不需要像 C 语言中那样通过循环遍历来访问多个数据, 而可以直接通过下标进行。例如

```
list[0 : 3]
```

将依次输出列表中的前四个数据。

题目 10.

python 中的列表有哪些可用函数?

解决方法:

列表可以自由增添数据, 而无需考虑列表的数据个数上限。这一点类似于 C++ 中的 vector 容器。

list.append()

使用这个函数可以在列表末尾增添括号里的数据。

del list[x]

可以使用 del 函数删除列表中下标为 x 的数据。

还有许多其他功能的函数, 例如:

list.sort()

可以进行排序。

list.reverse()

可以将列表反向。

题目 11.

定义一个类, 能将输入的字母转变为大写字母输出。

解决方法:

python 中同样有类的概念, 在类中定义两个函数, 一个用于接收字符串, 另一个用于将其转变为大写形式并输出。

```
class A(object):
    def __init__(self):
        self.s = ""

    def getString(self):
        print('输入字符串: ')
        self.s = input()

    def printString(self):
        print (self.s.upper())

strr = A()
strr.getString()
strr.printString()
```

图 12: 4.py 代码

题目 12.

python 中的函数与 C 语言有什么区别?

解决方法:

python 函数使用 def 作为关键词开头，后接函数名和参数。函数体部分以冒号开头，且要求首行缩进。这种函数还可以接收不定长度的参数，并将多出来的参数放在以 * 开头的变量中，类似于默认参数函数但不限制参数个数，更为方便。例如

```
def function(k,*i):
```

题目 13. 编写一个程序，接受单词序列，并按字母顺序排序后打印单词。**解决方法:**

使用循环遍历字符串，设置用空格分隔。使用列表的排序功能后输出列表。编写代码如下：

```
i=[x for x in input().split(' ')]  
i.sort()  
print (' '.join(i))
```

图 13: 5.py 代码

题目 14.

编写一个程序，判断输入的一个正整数是否为素数。

解决方法:

根据所学算法进行判断，编写代码及运行结果如下：


```

num = int(input("please input num: "))
ii = True

if num < 2:
    ii = False
else:
    for i in range(2, int(num**0.5) + 1):
        if num % i == 0:
            ii = False
            break

if ii:
    print("is")
else:
    print("isn't")

```

图 14: 6.py 代码

```

a@ubuntu:~$ python 6.py
please input num: 79
is

```

图 15: 6.py 运行结果

题目 15.

编写一个能够判断单词出现次数的程序。

解决方法:

在 python 中完成这个问题较为简单，不需要像 C 语言中那样逐一判断，只需要以空格为间隔读入单词，并计算其中出现次数即可。代码如下：

```

text = input("Please input a string: ")
words = text.split()
word_count = {}

for word in words:
    word_count[word] = word_count.get(word, 0) + 1

for word, count in word_count.items():
    print(f"{word}: {count}")

```

图 16: 7.py 代码

运行结果如下：

```
a@ubuntu:~$ python3 7.py
Please input a string: w ww www w
w: 2
ww: 1
www: 1
```

图 17: 7.py 运行结果

题目 16.

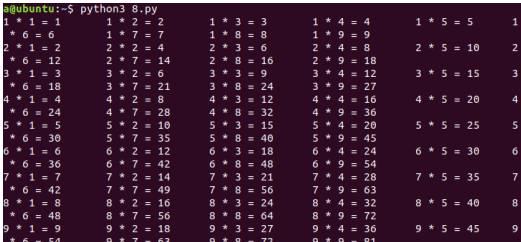
用 python 打印出九九乘法表。

解决方法：

使用循环语句输出运算表格，代码及运行结果如下：

```
for i in range(1, 10):
    for j in range(1, 10):
        print(f"{i} * {j} = {i * j}", end="\t")
    print()
```

图 18: 8.py 代码



1 * 1 = 1	1 * 2 = 2	1 * 3 = 3	1 * 4 = 4	1 * 5 = 5	1
1 * 6 = 6	1 * 7 = 7	1 * 8 = 8	1 * 9 = 9		
2 * 1 = 2	2 * 2 = 4	2 * 3 = 6	2 * 4 = 8	2 * 5 = 10	2
2 * 6 = 12	2 * 7 = 14	2 * 8 = 16	2 * 9 = 18		
3 * 1 = 3	3 * 2 = 6	3 * 3 = 9	3 * 4 = 12	3 * 5 = 15	3
3 * 6 = 18	3 * 7 = 21	3 * 8 = 24	3 * 9 = 27		
4 * 1 = 4	4 * 2 = 8	4 * 3 = 12	4 * 4 = 16	4 * 5 = 20	4
4 * 6 = 24	4 * 7 = 28	4 * 8 = 32	4 * 9 = 36		
5 * 1 = 5	5 * 2 = 10	5 * 3 = 15	5 * 4 = 20	5 * 5 = 25	5
5 * 6 = 30	5 * 7 = 35	5 * 8 = 40	5 * 9 = 45		
6 * 1 = 6	6 * 2 = 12	6 * 3 = 18	6 * 4 = 24	6 * 5 = 30	6
6 * 6 = 36	6 * 7 = 42	6 * 8 = 48	6 * 9 = 54		
7 * 1 = 7	7 * 2 = 14	7 * 3 = 21	7 * 4 = 28	7 * 5 = 35	7
7 * 6 = 42	7 * 7 = 49	7 * 8 = 56	7 * 9 = 63		
8 * 1 = 8	8 * 2 = 16	8 * 3 = 24	8 * 4 = 32	8 * 5 = 40	8
8 * 6 = 48	8 * 7 = 56	8 * 8 = 64	8 * 9 = 72		
9 * 1 = 9	9 * 2 = 18	9 * 3 = 27	9 * 4 = 36	9 * 5 = 45	9
9 * 6 = 54	9 * 7 = 63	9 * 8 = 72	9 * 9 = 81		

图 19: 8.py 运行结果

题目 17.

编写一个接受句子的程序，并计算大写字母和小写字母的数量。

解决方法：

编写代码如下：

```
def count_case(sentence):
    upper_count = 0
    lower_count = 0

    for char in sentence:
        if char.isupper():
            upper_count += 1
        elif char.islower():
            lower_count += 1

    return upper_count, lower_count

# 主程序
if __name__ == "__main__":
    sentence = input("Please enter a sentence: ")
    upper_count, lower_count = count_case(sentence)

    print(f"Number of uppercase letters: {upper_count}")
    print(f"Number of lowercase letters: {lower_count}")
```

图 20: 9.py 代码

运行结果正确：

```
a@ubuntu:~$ python3 9.py
Please enter a sentence: WSDJINAFadawiubWBwjD
Number of uppercase letters: 10
Number of lowercase letters: 10
```

图 21: 9.py 运行结果

题目 18.

用 python 计算斐波那契数列。

解决方法：

根据所学知识，编写代码如下：

```
def recur_fibo(n):  
    if n <= 1:  
        return n  
    else:  
        return(recur_fibo(n-1) + recur_fibo(n-2))  
num = int(input("number"))  
for i in range(num):  
    print(recur_fibo(i))
```

图 22: 10.py 代码

运行结果:

```
a@ubuntu:~$ python3 1.py  
number 12  
0  
1  
1  
2  
3  
5  
8  
13  
21  
34  
55  
89
```

图 23: 10.py 运行结果

题目 19.

写一个能够返回两个数最大公因数的程序。

解决方法：

可以根据 C 语言代码进行改编，如下图所示：

```
def gcd(a, b):  
    while b:  
        a, b = b, a % b  
    return a  
  
num1 = int(input("请输入第一个整数: "))  
num2 = int(input("请输入第二个整数: "))  
print(f"{num1}和{num2}的最大公约数是: {gcd(num1, num2)}")
```

图 24: 11.py 代码

运行结果正确，能够正常计算。

```
a@ubuntu:~$ python3 11.py  
请输入第一个整数: 100  
请输入第二个整数: 10  
100和10的最大公约数是: 10
```

图 25: 11.py 运行结果

题目 20.

编写一个 Python 程序，打印一个由星号 (*) 组成的等腰直角三角形。

解决方法：

编写代码及运行结果：

github 链接及提交记录:

GitHub 链接: <https://github.com/28935/23020007016>

commit 记录截图:

commit 记录截图:

































python	
 28935 committed Sep 11, 2024	 89c6720  
Commits on Sep 10, 2024	
2	
 28935 committed Sep 10, 2024	 776d5d2  
sum	
 28935 committed Sep 10, 2024	 a546f71  
html	
 28935 committed Sep 10, 2024	 08c8c4a  
debug	
 28935 committed Sep 10, 2024	 bc27c72  
Commits on Aug 29, 2024	
ignore	
 28935 committed Aug 29, 2024	 c7c287a  
add 11111111111111111111111111111111 to password	
 28935 committed Aug 29, 2024	 00a8d36  
Please enter the commit message for your changes. Lines starting	
 28935 committed Aug 29, 2024	 08a9a80  

图 28: commit 记录