第12周作业

张治卓517111910078

1 Python Fundamentals

1.1 Python解释器

> 在Python环境中输入下面的语句，会输出什么?
>
> ```
> (a) 3 + 1
> (b) 3 * 3
> (c) 2 ** 38
> (d) "Hello, world!"
> ```

(a)4
(b)9
(c)274877906944
(d)’ Hello, world!’

1.2 Python脚本文件

> 将上述脚本保存为test1.py，然后运行该脚本，能否正常运行。如果不能，为什么？你能否改正这个
> 问题?

不能。因为脚本里输出需要使用print()函数。修改后的脚本为

```
print(3 + 1)
print(3 * 3)
print(2 ** 38)
print("Hello, world!")
```

1.3 字符串

> 解释下列语句的输出结果:
>
> ```
> (a) 'py' + 'thon'
> (b) 'py' * 3 + 'thon'
> (c) 'py' - 'py'
> (d) '3' + 3
> (e) 3 * '3'
> (f) a
> ```

```
(g) a = 3
(h) a
```

(a)’ python’
字符串+等于连接合并
(b)’ pypypython’
字符串*int等于重复int次
(c) `Traceback (most recent call last):`
`File "<pyshell#11>", line 1, in <module>`
`'py' - 'py'`
`TypeError: unsupported operand type(s) for -: 'str' and 'str'`
字符串不支持-运算
(d) `Traceback (most recent call last):`
`File "<pyshell#12>", line 1, in <module>`
`'3' + 3`
`TypeError: can only concatenate str (not "int") to str`
不支持字符串和整型+运算
(e)’ 333’
字符串前后与整型*运算都代表重复
(f) `Traceback (most recent call last):`
`File "<pyshell#14>", line 1, in <module>`
`a`
`NameError: name 'a' is not defined`
a变量并没有定义，因此不能输出
(g)（无输出）
对变量a赋值3，不产生输出
(h)3
因为前一步已经对a赋值3，因此可以输出a的值3

1.4 布尔类型

> 解释下列语句的输出结果：
>
> ```
> (a) 1 == 1
> (b) 1 == True
> (c) 0 == True
> (d) 0 == False
> (e) 3 == 1 * 3
> (f) (3 == 1) * 3
> (g) (3 == 3) * 4 + 3 == 1
> (h) 3**5 >= 4**4
> ```

(a) True
1与1相等，返回True

(b) True
因为True就是1，因此相等返回True
(c) False
因为True是1不等于0，因此返回False
(d) True
因为False就是0，因此返回True
(e) True
因为1*3等于3，所以返回True，且*优先于==
(f) 0
首先3==1返回False即0，后计算0*3返回0
(g) False
首先3==3返回True即1，然后1*4+3等于7不等于1，因此返回False
(h) False
等号前面等于243，后面等于256，因此前面不是大于等于后面，返回False

1.5 整数

> 解释下列语句的输出结果：
>
> ```
> (a) 5 / 3
> (b) 5 % 3
> (c) 5.0 / 3
> (d) 5 / 3.0
> (e) 5.2 % 3
> (f) -9 % -5
> (g) 9 % -5
> (h) 2001 ** 200
> ```

(a) 1.6666666666666667
输出5/3的浮点数
(b) 2
输出5/3的余数
(c) 1.6666666666666667
输出5/3的浮点数
(d) 1.6666666666666667
输出5/3的浮点数
(e) 2.2
输出5.2/3的余数，即5.2%3=1…2.2
(f) -4
输出-9%-5的余数，即-9%-5=1…-4
(g) -1
输出9%-5的余数，即9%-5=-2…-1
Python倾向于让商更小，因此是-2…-1，而不是-1…4
(h)177589681048312191435093479787150106345284342884379442232320253088728153650

4521062992112989811320174987523429734050780420176145359603401626418950118692
4066128377025843892373608427790859511135990682732202975330824797118808624727
3516081831941545572087304944401104296356500574318336742864624635087552763028
9615433647578276861396433276410813253392557034222034069897376138054129497013
9762186212823359128790706292900765512137078550033912252338262922477518858757
1148400125765147247423881245950613015022229348060740326886911708809968819674
2644294782826105785287103236687917999612221638587027302050607924010391072876
6397733398071775041745854959302025036249707279600400001
输出2001的200次方

1.6 浮点数

> 解释下列语句的输出：
>
> ```
> (a) 2000.3 ** 200 (与前面的结果进行比较)
> (b) 1.0 + 1.0 - 1.0
> (c) 1.0 + 1.0e20 - 1.0e20
> ```

(a) `Traceback (most recent call last):`
`File "<pyshell#40>", line 1, in <module>`
`2000.3 ** 200`
`OverflowError: (34, 'Result too large')`
结果太大，无法输出（应该是浮点溢出）
(b)1.0
正常的浮点数运算
(c)0.0
后面的1.0e20是科学计数法，因为远大于1.0，所以第一步结果还是1.0e20，最后相减变为0

1.7 变量

> 写一个脚本，用变量名name存储姓名的字符串。假设你的姓名是Lifu Zhang，要求输出：
> Hello, Lifu
> Zhang! 。

```
name = "Zhang Zhizhuo"
print("Hello, %s!" % name)
```

1.8 类型强制转换

> Python要实现类型的强制转换很简单，一般的函数是是目标类型(x)，如int(x)可以将x 转换为整型变
> 量。
> 解释下面的输出：

```
(a) float(123)
(b) float('123')
(c) float('123.23')
(d) int(123.23)
(e) int('123.23')
(f) int(float('123.23'))
(g) str(12)
(h) str(12.2)
(i) bool('')
(j) bool('a')
(k) bool(0)
(l) bool(0.1)
(m) bool([])
(n) bool([[]])
(o) bool(())
(p) bool((()))
```

(a) 123.0
将整形转换为浮点型
(b) 123.0
将字符串对应10进制数字转换为浮点型
(c) 123.23
将字符串对应浮点数字转换为浮点型
(d) 123
将浮点数取整变为整型
(e) `Traceback (most recent call last):`
`File "<pyshell#55>", line 1, in <module>`
`int('123.23')`
`ValueError: invalid literal for int() with base 10: '123.23'`
字符串对应的数字不是10进制无法转换
(f) 123
先将字符串对应的浮点数字转换为浮点型，再转换为整型，避免了上一问的错误
(g) ‘12’
将整型直接转换为字符串
(h) ‘12.2’
将浮点型直接转换为字符串
(i) False
空串被判定为False
(j) True
非空串被判定为True
(k) False
0被判定为False
(l) True

非0数被判定为True
(m) False
空串被判定为False
(n) True
因为[[]]是非空串，所以判定为True
(o) False
空串被判定为False
(p) False
(())相当于()，空串被判定为False

1.9 is与==的区别

> 说说is和==的联系与区别，说明什么情况下两者相同，什么情况下两者不同。
>
> ```
> (a) x=123; y=123L
> (b) x == y
> (c) x is y
> (d) id(x) == id(y)
> (e)import sys; sys.getrefcount(x)
> ```

在比较数值时相同，比较变量时实际上比较的是id所以可能不同
因此相同的是(b)
不同的是(c)(d)
结果为：
(a)无输出
(b)True
(c)False
(d)False
(e)6

2 Flow control

2.1 考拉兹猜想（Collatz sequence）

```
x = 103
print(x, end=' ')
while x != 1:
    if x % 2 == 0:
        x = x // 2
    else:
        x = 3 * x + 1
    print(x, end=' ')
```

结果:
103 310 155 466 233 700 350 175 526 263 790 395 1186 593 1780 890 445 1336 668 334 167 502 251 754 377 1132 566 283 850 425 1276 638 319 958 479 1438 719 2158 1079 3238 1619 4858 2429 7288 3644 1822 911 2734 1367 4102 2051 6154 3077 9232 4616 2308 1154 577 1732 866 433 1300 650 325 976 488 244 122 61 184 92 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1

3 Functions

3.1 寻找质数的函数

```
# 判断n是否为质数
def is_prime(n):
    if n == 1 or n <= 0:
        return False
    elif n == 2 or n == 3:
        return True
    elif n % 6 == 1 or n % 6 == 5:
        for i in range(2, n):
            if n % i == 0:
                return False
        return True
    else:
        return False


# 输出所有小于n的质数
def less(n):
    for i in range(0, n):
        if is_prime(i):
            print(i, end=' ')


# 输出前n个质数
def head(n):
    count = 0
    i = 0
    while count < n:
        if is_prime(i):
            print(i, end=' ')
            count = count + 1
        i = i + 1
```

3.2 解方程

(1)

```
def fun(n):
    return n * n - 2 * n - 3


def root(f, a, b):
    x = (a + b) / 2.0
    if f(a) == 0:
        return a
    elif f(b) == 0:
        return b
    elif f(x) == 0:
        return x
    if abs(b-a) < 0.001:
        return x
    elif f(a) * f(b) < 0:
        if f(a) * f(x) < 0:
            return root(f, a, x)
        else:
            return root(f, x, b)


print("the root is %0.2f" % root(fun, -1.3, 1.2))
```

(2)

```
def fun(n):
    return n * n - 2 * n - 3


def root(f, a, b):
    x = (a + b) / 2.0
    if a > b:
        a, b = b, a
    if f(a) == 0:
        return "the root is %0.2f" % a
    elif f(b) == 0:
        return "the root is %0.2f" % b
    elif f(x) == 0:
        return "the root is %0.2f" % x
    if abs(b-a) < 0.001:
```

```
        return "the root is %0.2f" % x
    elif f(a) * f(b) < 0:
        if f(a) * f(x) < 0:
            return root(f, a, x)
        else:
            return root(f, x, b)
    elif f(a) * f(b) > 0:
        return "f(a) and f(b) are both over or below zero"


print(root(fun, 1.3, 1.3001))
```