Python编程新思维及实战

嵩 天

实例1：蒙特卡罗猜测与计时

嵩 天

Python编程新思维及实战

"蒙特卡罗猜测与计时"需求分析

Python123

需求分析

蒙特卡罗猜测：计算机匹配正则表达式

- 输入：一个正则表达式，由程序员给出
- 程序：随机产生字符串，匹配正则表达式
- 计时：统计时间及猜测次数

需求分析

蒙特卡罗猜测：计算机匹配正则表达式

- 正则表达式： r'[1-2][^2-8][D-F]0+[A-F]’)
- 随机字符串：32个字符长度，字母表是0-9,A-Z，十六进制字符
- 输出：匹配次数、匹配字符串、程序关键部分所用时间(5位小数)

需求分析

需求的真实应用

- 正则表达式代表病毒片段： r'[1-2][^2-8][D-F]0+[A-F]’)
- 任何文件都可以表示为十六进制字符的组合形式
- 匹配：病毒引擎的扫描过程

Python编程新思维及实战

"蒙特卡罗猜测与计时"实例编写

Python123

代码纵览

```
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

引入3个标准库

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

定义能够生成32个随机字符的函数

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

设置一个全局字母表约定随机产生的字符范围

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

随机产生32个字符，并将产生后的字符串返回

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

编译正则表达式

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

记录并统计
猜测次数

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

对核心代码部分设置计时功能

代码详解

```python
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

进行正则表达式查找

根据匹配结果决定程序是否继续循环执行

代码详解

```
import time, random, re

def genStr():
    global sigma
    s = ""
    for i in range(32):
        s += sigma[random.randint(0, 15)]
    return s

sigma = "0123456789ABCDEF"
regex = re.compile(r'[1-2][^2-8][D-F]0+[A-F]')
count = 0
start = time.perf_counter()
match = regex.search(genStr())
while not match:
    count += 1
    match = regex.search(genStr())
print("程序匹配: 猜测{}次, {}->{}".format(count, match.string, match.group(0)))
end = time.perf_counter()
print("程序用时:{:.5f}秒".format(end-start))
```

输出匹配字符串

代码详解

程序匹配: 猜测219次，776351EAB26BB60910B8F9194C2CE0C5->2CE0C

程序用时:0.01107秒

程序匹配: 猜测68次，4A76219D0C7935E73DEB462C22EF32D8->19D0C

程序用时:0.00317秒

程序匹配: 猜测198次，FD33470040746AB923C4621F0BEF33E2->21F0B

程序用时:0.00972秒


Python 123
Thank you