Dragon Realm

Making Game with Python (1)

Zhihong (John) Zeng & Andrew Zeng

Today

- Review and test
- import time module
- Escape character and multiline string
- User-defined function
- While loop
- Boolean operators: and, or , not
- Dragon Realm

Review

- Function and module
- Import random module
- Flow control (if else and for loop)

Test

```
# find the bugs

number = 3
guess = input(‘input your guess ’)

if guess == number:
print(‘Your guess is correct’)

elif guess < number
        print(‘Your guess is too small’)
                print(‘done’)

else:
        print(‘Your guess is too large’)
```

Test

```
# find the bugs

number = 3
guess = input('input your guess ')

if guess == number:
print('Your guess is correct')

elif guess < number
        print('Your guess is too small')
                print('done')

else:
        print('Your guess is too large')
```

```
# find the bugs

number = 3
guess = int(input('input your guess '))

if guess == number :
    print('Your guess is correct')

elif guess < number :
        print('Your guess is too small')
        print('done')

else:
        print('Your guess is too large')
```

Test

```
# what will be printed

for x in range(5):
        print(x, end=’  ‘)

for x in [3, 2, 1, 0, -1]:
        print(x)
        if x<0:
                print(‘negative number’)
```

Test

```
# what will be printed

for x in range(5):
        print(x, end=’ ‘)

for x in [3, 2, 1, 0, -1]:
        print(x)
        if x<0:
                print(‘negative number’)
```

```
# what will be printed

0 1 2 3 4


3
2
1
0
-1
Negative number
```

Escape Characters

‘\’(backslash): “escape” character. It is used to print certain special characters:

Escape character	What is actually printed	Examples
\’	Single quote (‘)	print(‘It\’s a test’)
\”	Double quote(”)	print(“He said: \“sure\” ”)
\n	Newline	print(‘left\nright)
\t	Tab	print(‘left\tright’)
\\	Backslash(\)	print(‘Backslash \\’)

Multiline string

- Using escape character ‘\n’
  - A = ‘This is \na test’
  - print(A)
- Using ‘ ’ ’... ’ ’ ’ or “ ” ” ….” “ “
  - A = ‘ ’ ’This is
  - a test’ ‘ ‘
  - print(A)

Import time module

- import time
- time.asctime():
  - string of local time: e.g., 'Sat Sep 28 17:22:20 2019'
- time.sleep(second):
  - suspend execution of the program for certain seconds
- time.time() :
  - the time in seconds since the epoch (1/1/1970, 00:00:00)

Import time module

- How to calculate the duration:
  - import time
  - start = time.time()
  - (do something)
  - duration = time.time() - start
  - print(duration)

User-defined functions

- Advantage of user-defined functions
  - Written once, used multiple time
  - Helpful to organize and maintain code
- Syntax 1:

```
def function_name(arg1, arg2, ...):
        statement1
        Statement2
        .....

# calling the function
function_name(var1, var2, …)
```

Exercise:

```
def my_function(name, school):
        print(f‘my name is {name}’)
        print(‘my school is {school}’)


my_function(‘Amy’, ‘Gates’)
```

User-defined functions (cont)

- Syntax 2:

```
def my_function(arg1, arg2, ...):
        statement1
        statement2
        ……
        return value

ans = my_function(val1, val2, ...)
print(ans)
```

```
def my_function(name, school):
        print('my name is ', name)
        print('my school is ', school)
        return 'Done'

ans = my_function('Amy', 'Gates')
print(ans)
```

User-defined functions (cont)

- Syntax 3:

```
def my_function(arg1, arg2=default, ...):
        statement1
        statement2
        ……
        return value

# val2 is optional
ans = my_function(val1, val2, ...)
print(ans)
```

```
def my_function(name, school='Gates'):
        print('my name is ', name)
        print('my school is ', school)
        return 'Done'

ans = my_function('Amy')
print(ans)

my_function('Amy', 'Conant')
```

While statement

- Difference between while and for loop
  - For loop: loops a specific number of times
  - While loop: loop repeats as long as a certain condition is True
  - “For loop” can always be replaced with “while loop”, but not always otherwise
- Syntax:

```
statement1
while condition==True:
        statement2
        statement3
        ……

statement4
```

While loop:

```
counter = 0
while counter < 5:
        print(counter)
        counter = counter +1

print('Done')
```

For loop:

```
for counter in range(5):
        print(counter)

print('Done')
```

Boolean operators

- Boolean operators evaluate statement and return True or False
- True or False:
  - Cats have whiskers and dogs have tails
  - Cats have whiskers and dogs have wings

Boolean operator: and

- If values on both sides of keyword “and” are true, the statement is True
- If either side are false, the statement is False
- Exercise:

```
A = 7
A > 5
A < 10

A > 5 and A < 10

A > 10
A > 5 and A >10
```

Boolean operator: or

- If either side of keyword “or” is true, the statement is True
- If bother side are false, the statement is False
- Exercise:

A = 7

A > 5 or A < 10

A > 5 or A >10

A < 5 or A > 10

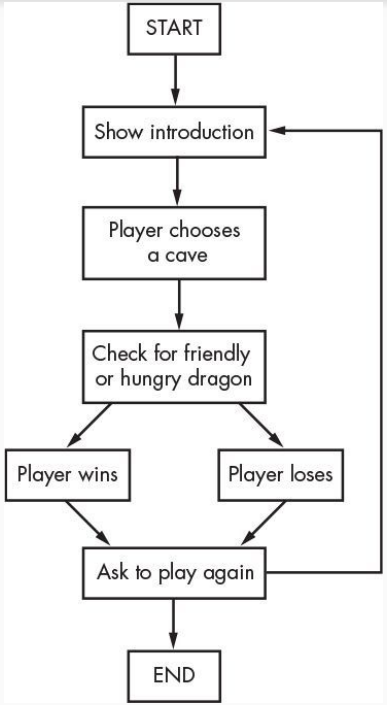
Boolean operator: not

- Return the opposite boolean value of the statement
- Exercise:

```
A = 7

not A > 5

not A < 5

not (A > 5 or A < 10)  # combination
```

Dragon Realm

Dragon Realm: flow chart



```
import random
import time

def displayIntro():
    print('''
            you see two caves.
            In one cave, the dragon is friendly.
            In the other cave, the other dragon is greedy and hungry.
            ''')

def chooseCave():
    cave = ''
    while cave not in ['1', '2']:
        cave = input('Which cave will you go into? (1 or 2) ')
    return cave

def checkCave(chosenCave):
    print('A large dragon jumps out in front of you! He opens his jaws and...\n')
    time.sleep(2)
    number = random.randint(1, 2)
    if chosenCave == str(number):
        print('Gives you his treasure!')
    else:
        print('Gobbles you down in one bite!')

playAgain = 'yes'
while playAgain.startswith('y'):
    displayIntro()
    caveNumber = chooseCave()
    checkCave(caveNumber)
    print()
    playAgain = input('Do you want to play again? (yes or no)')
```