Bagels Deduction Game

Making Game with Python (1)

Zhihong (John) Zeng & Andrew Zeng

Agenda

- Introduction
- Flowchart
- function definition and test
    - Introduction
    - get_random_number
    - get_player_input
    - get_clue
- Main function

Demo

Introduction

```
NUM_DIGIT = 3
MAX_GUESS = 10

def introduction():
    intro = '''
    I am thinking of a {}-digit number. Try to guess what it is.
    The clues I give are...
    When I say:    That means:
    Bagels       None of the digits is correct.
    Pico         One digit is correct but in the wrong position.
    Fermi        One digit is correct and in the right position.
    I have thought up a number. You have {} guesses to get it.
    '''.format(NUM_DIGIT, MAX_GUESS)
    print(intro)
```

Flowchart

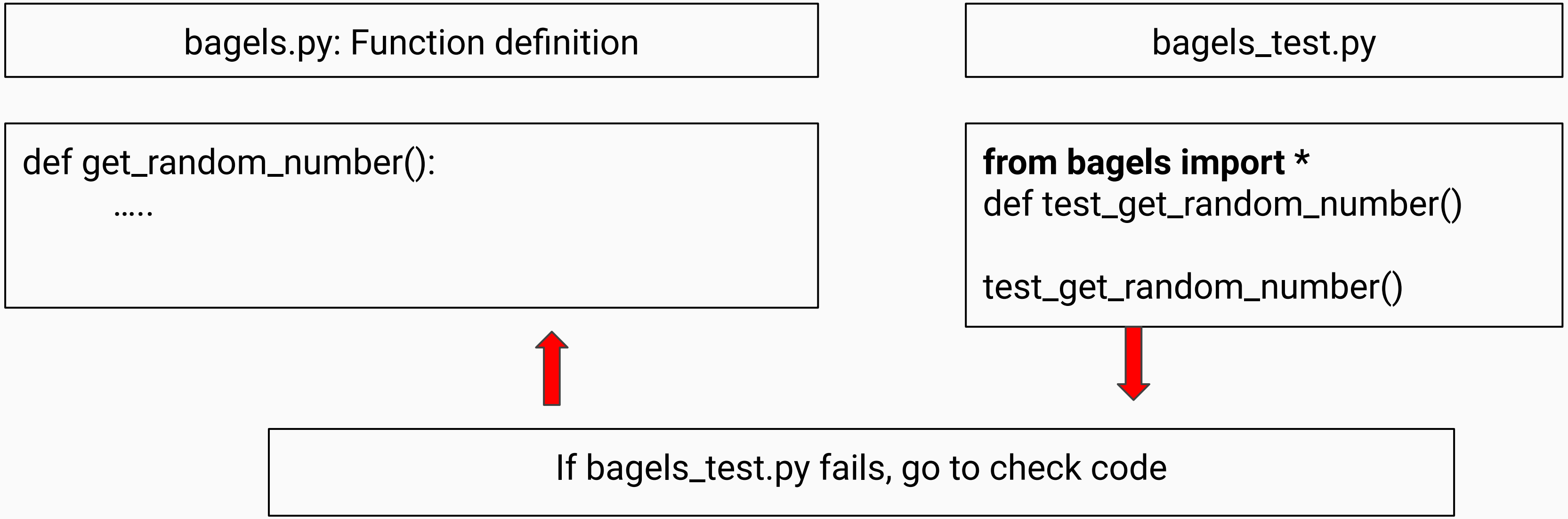
Intro

get_random_number

```
For loop with max guess:
    print(message)
    guess = get_player_input()
    clue = get_clue(guess)
    If clue == ‘Won’
        print(congrats)
        return
    else:
        print(clue)
```

print(‘Ran out of guesses’)

Debug: function definition and test



bagels.py: Function definition

```
def get_random_number():
        .....
```

bagels_test.py

```
from bagels import *
def test_get_random_number()

test_get_random_number()
```

If bagels_test.py fails, go to check code

Create random number

```
def get_random_number(size):
    nums = list(range(1, 10))
    random.shuffle(nums)
    ans = ''
    for x in nums[:size]:
        ans += str(x)
    return ans
```

```
def test_get_random_number():
    print('--------test get_random_number')
    print(get_random_number(2))
    print(get_random_number(2))
    print(get_random_number(3))
    print(get_random_number(3))
    print(get_random_number(3))
    print('-------done------\n')

test_get_random_number()
```

get_player_input

```
def get_player_input(size):
    ans = ''
    while len(ans) != size or not ans.isdigit():
        ans = input('Make a guess ({} digits): \n'.format(size))
    return ans
```

```
def test_get_player_input():
    print('--------test get_player_input')
    ans = get_player_input(3)
    print('Your guess is {}'.format(ans))
    print('--------done-----\n')

test_get_player_input()
```

Get clue

```
def get_clue(guess, secrete_number):
    if guess == secrete_number:
        return 'Won'

    clue = []
    for i, x in enumerate(guess):
        if x == secrete_number[i]:
            clue.append('Fermi')
        elif x in secrete_number:
            clue.append('Pico')

    if not clue:
        return 'Bagels'
    else:
        # clue.sort()
        return ' '.join(clue)
```

```
def test_get_clue():
    print('--------test get_clue------')
    data = {
        ('123', '123'): 'Won',
        ('123', '213'): 'Pico Pico Fermi',
        ('123', '230'): 'Pico Pico',
        ('123', '456'): 'Bagels'
    }
    for (guess, secrete), value in data.items():
        ans = get_clue(guess, secrete)
        if ans != value:
            print('Failure: get_clue({}, {}) is expected to be {}, \
                        but got {}'.format(guess, secrete, value, ans))
            return
    print('--------Success------\n')

def test_get_clue()
```

Main function

```
def bagels_game():
    introduction()
    secrete = get_random_number(NUM_DIGIT)

    for i in range(MAX_GUESS):
        print('\n#{}:'.format(i+1))
        ans = get_player_input(NUM_DIGIT)
        clue = get_clue(ans, secrete)
        if clue == 'Won':
            print('Congrats! You got it.')
            return
        else:
            print(clue)

    print('You ran out of guesses. The answer was {}'.format(secrete))
```

program entry

```
if __name__ == '__main__':
    bagels_game()
```

Dictionary

- Dictionary is a collection which is to store data with key-value pairs
- Syntax: A= {key1: value1, key: value2}
- Access
  - Value = A[key]   # compare: access list value
- Add element
  - A[new_key] = new_value
- Check the existence of a key
  - Key in A
- Loop:
  - For k, v in A.items():
    - print(k, v)

```
#exercise

A = {'a': 1, 'b': 2}

print(A['b'])

A['c'] = 3

print(A)

print('a' in A)

For k, v in A.items():

        print(k, v)
```

Dictionary

- Get keys
  - A.keys()   # compare: access list value
- Get values
  - A.values()
- Deletion
  - A.pop(key)

```
# exercise continue:

print(A.keys())

print(A.values())

print(A.pop('a'))  # return the related value

print(A)



for k in A:

        print(k)
```