

Bagels Deduction Game

Making Game with Python (1)

Zhihong (John) Zeng & Andrew Zeng



Agenda

- List operations
- Sets
- Dictionaries
- Demo
- function definition and test
 - Introduction
 - `get_random_number`
 - `get_player_input`
 - `get_clue`
- Main function

Creating a list

- Writing out all the elements: `x = [1, 2, 3, 4, etc]`
- Concatenation:
`x = [1, 2] + [3, 4]`, creates list `[1, 2, 3, 4]`
`x = [1] * 10`, creates list `[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]`
- Split: `x = "a b c d e".split()`, creates list `["a", "b", "c", "d", "e"]`
- List: `x = list("abcd")`, creates list `["a", "b", "c", "d"]`
- Loop: `x = [i for i in range(5)]`, creates list `[0, 1, 2, 3, 4]`

Other list operations

- Join

- Creates a string with every item in the list merged together
- Ex: `a = ["a", "b", "c"]`
`print("".join(a))`

- Sublist

- Allows you to access parts of the list
- Ex: `a = [1, 2, 3, 4, 5]`
`print(a[:2])`
`print(a[3:])`
`print(a[1:3])`
`print(a[:-1])`

Sets

- A collection of easily accessible items

Syntax: {1, 2, 3 ...}

- Creating a set

```
x = set()
```

```
x = {1, 2, 3}
```

- Adding to a set

```
x.add(4)
```

- Deleting from a set

```
x.delete(1)
```

- Get set size

```
len(x)
```

Lists vs sets

	length = 5				
	'p'	'r'	'o'	'b'	'e'
index	0	1	2	3	4
negative index	-5	-4	-3	-2	-1

- Order vs accessibility
- Finding stuff and iterating
- Duplicates

"p"	"r"	"o"	"b"	"e"
15	17	14	1	4

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)`
 - `del A['a']`

exercise continue:

```
print(A.keys())
```

```
print(A.values())
```

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

```
print(A)
```

```
for k in A:
```

```
    print(k)
```


Demo

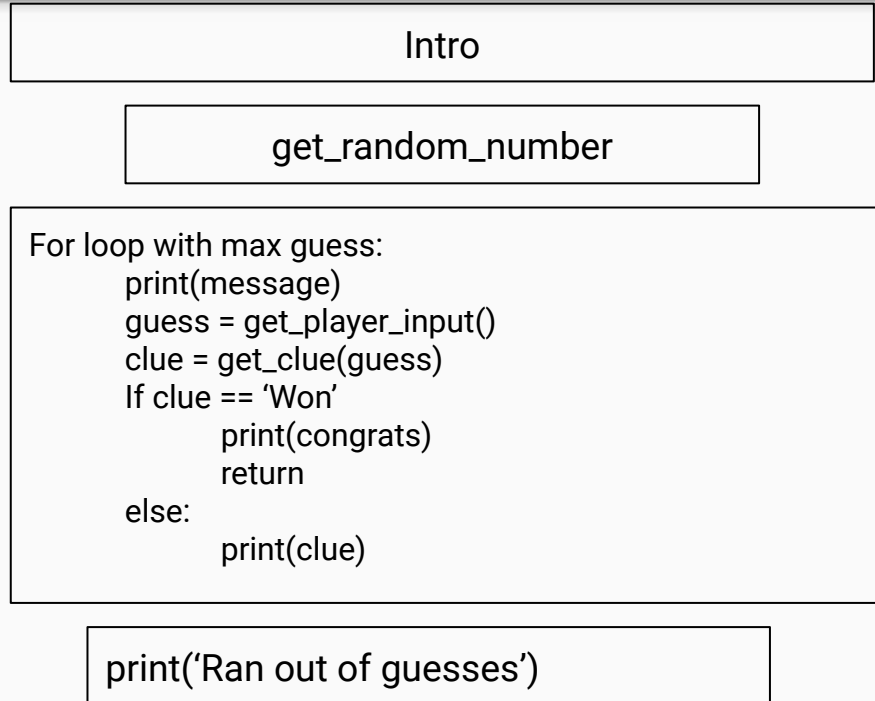


Introduction

```
NUM_DIGIT = 3  
MAX_GUESS = 10
```

```
def introduction():  
    intro = f"  
    I am thinking of a {NUM_DIGIT}-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 {MAX_GUESS} guesses to get it."  
  
    print(intro)
```

Flowchart



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, secret, secretDict, secretSet):  
    if guess == secret:  
        return "won"  
  
    clue = []  
    for i in range(len(guess)):  
        if guess[i] == secretDict[i]:  
            clue.append("fermi")  
        elif guess[i] in secretSet:  
            clue.append("Pico")  
  
    if not clue:  
        return "bagels"  
    else:  
        return "".join(clue)
```

```
def secret_set(secret):  
    secret_set = set()  
    for i in secret:  
        secret_set.add(i)  
  
    return secret_set  
  
def secret_dict(secret):  
    ind = {}  
    for i in range(len(secret)):  
        ind[i] = secret[i]  
  
    return ind
```

Main function

```
def bagels():
    introduction()

    secret = get_random_num(num_digit)
    secretSet = secret_set(secret)
    secretDict = secret_dict(secret)

    for i in range(max_guess):
        print(f"Guess {i + 1}")
        ans = get_input(num_digit)
        clue = get_clue(ans, secret, secretDict, secretSet)

        if clue == "won":
            print("You guessed the number!")
            return
        else:
            print(clue)

    print(f"You ran out of guesses, the number was {secret}")
```


program entry

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