# ICT 133
# Structured Programming

## Seminar 5

## Topics

- Nested lists
- Dictionary

## List example without nesting

```
month = (31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31)
monthName = ('January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October', 'November', 'December')

for i, name in enumerate(monthName):
    print('{} has {} days'.format(name, month[i]))
```

*Output:*

```
January has 31 days
February has 28 days
March has 31 days
...
```

## List example with nesting

```
month = [['January', 31], ['February', 28], ['March', 31], \
        ['April', 30],  ['May', 31],  ['June', 30], \
        ['July', 31],  ['August', 31],  ['September', 30],\
        ['October', 31],  ['November', 30],  ['December', 31]]

for m in month:
    print('{} has {} days'.format(m[0], m[1]))
```

*Output:*

```
January has 31 days
February has 28 days
March has 31 days
…
```

## More List Operations

| Method | Meaning |
|---|---|
| <list>.sort() | Sort (order) the list. A comparison function may be passed as a parameter. |
| <list>.reverse() | Reverse the list. |
| <list>.index(x) | Returns index of first occurrence of x. |
| <list>.count(x) | Returns the number of occurrences of x in list. |

Source: Python Programming, 3/e 5

## Nested List example

```
Enter name: evelyn
Enter score: 22
Enter name: helen
Enter score: 33
Enter name: george
Enter score: 33
Enter name: alice
Enter score: 22
Enter name:
George scored 33
Helen scored 33
Alice scored 22
Evelyn scored 22
```

Scores in descending order with names in ascending order for names with same score

# Nested List example 2

```
def getScores():
    scores = []
    while True:
        name = input('Enter name: ')
.capitalize()
        if name == '': break
        score = int(input('Enter score: '))
        scores.append([name, score])
    return scores

def byScore(elem):
    return elem[1]
```

```
def main():
    scores = getScores()
    scores.sort()
    scores.sort(key = byScore,
reverse = True)

    for s in scores:
        print('{} scored {}
'.format(s[0], s[1]))

main()
```

| | |
|---|---|
| getScores(): | [['Evelyn', 22], ['Helen', 33], ['*George*', 33], ['Alice', 22]] |
| scores.sort() | [['Alice', 22], ['Evelyn', 22], ['*George*', 33], ['Helen', 33]] |
| scores.sort(key = byScore) | [['*George*', 33], ['Helen', 33], ['Alice', 22], ['Evelyn', 22]] |

## What is Alice’s score?

```
def getScores():
    return [['Evelyn', 22], ['Helen', 33],
['George', 33], ['Alice', 22]]


def searchScore(scores, name):
    score = [elem[1] for elem in scores if
elem[0] == name]
    if score != []:
        return score[0]
    else:
        return 'Not recorded'
```

```
def main():
    scores = getScores()
    while True:
        name = input("Enter name
of student or <ENTER> to end:
").capitalize()
        if name == '': break
        print(searchScore(scores,
name))

main()
```

## Dictionary Basics

- Accessing a value in a collection using a key rather than an index
  - a *key-value pair*

- Python dictionaries are *mapping*s.
  - Names and phone numbers
  - Usernames and passwords

# Dictionary Basics

- Key-value pairs within curly braces, separated by commas.
- Keys and values are joined by ":"

```
month = {'January': 31, 'February': 28, 'March': 31, \
    'April': 30, 'May': 31,  'June': 30, \
    'July': 31, 'August': 31,  'September': 30,\
    'October': 31,  'November': 30,  'December': 31}
```

- Keys can be any immutable type, values can be any type.
- Mappings are inherently unordered.

## Dictionary Basics

- Indexing notation `<dictionary>[<key>]`
  - returns the object associated with the key.
  - `month["May"]` evaluates to `31`
- Dictionaries are mutable.
  - `month["February"] = 29`

```
month ->  {'January': 31, 'February': 29, 'March': 31, \
        'April': 30, 'May': 31,  'June': 30, \
        'July': 31, 'August': 31,  'September': 30,\
        'October': 31,  'November': 30,  'December': 31}
```

## Dictionary Operations

- Common method to build dictionaries
  - start with an empty collection
  - add the key-value pairs one at a time.

```
passwd = {}
for line in open('passwords.txt', 'r'):
    user, pass = line.split()
    passwd[user] = pass
```

# Dictionary Operations

| Method | Meaning |
|---|---|
| <dict>.keys() | Returns a sequence of keys. |
| <dict>.values() | Returns a sequence of values. |
| <dict>.items() | Returns a sequence of tuples (key, value) representing the key-value pairs. |
| <key> in <dict> | Returns true if dictionary contains the specified key, false if it doesn't. |
| for <var> in <dict>: | Loop over the keys. |
| <dict>.get(<key>, <default>) | If dictionary has key returns its value; otherwise returns default. |
| del <dict>[<key>] | Deletes the specified entry. |
| <dict>.clear() | Deletes all entries. |

## Dictionary example

```
month = {'January': 31, 'February': 28, 'March': 31, |
        'April': 30, 'May': 31,  'June': 30, |
        'July': 31, 'August': 31,  'September': 30,|
        'October': 31,  'November': 30,  'December': 31}

for k, v in month.items():
    print('{} has {} days'.format(k, v))
```

*Output:*
```
January has 31 days
February has 28 days
March has 31 days
…
```

## What is Alice’s score?

```
def getScores():
    return {'Evelyn': 22, 'Helen': 33,
'George': 33, 'Alice': 22}


def searchScore(scores, name):
    return scores.get(name, 'Not
recorded')
```

```
def main():
    scores = getScores()
    while True:
        name = input("Enter name
of student or <ENTER> to end:
").capitalize()
        if name == '': break
        print(searchScore(scores,
name))

main()
```

## Sorting and searching

```
def getScores():
    scores = {}
    while True:
        name = input('Enter name: ')
.capitalize()
        if name == '': break
        score = int(input('Enter score: '))
        scores[name] =  score
    return scores

def byScore(elem):
    return elem[1]

def searchScore(scores, name):
    return scores.get(name, 'Not recorded')
```

```
def main():
    scores = getScores()
    scoreList = list(scores.items())
    scoreList.sort()
    scoreList.sort(key = byScore, reverse = True)

    for s in scoreList:
        print('{} scored {} '.format(s[0], s[1]))

    while True:
        name = input("Enter name of student or
<ENTER> to end: ").capitalize()
        if name == '': break
        print(searchScore(scores, name))
```

# Sorting and searching

```
def getScores():
    scores = {}
    while True:
        name = input('Enter name: ')
.capitalize()
        if name == '': break
        score = int(input('Enter score: '))
        scores[name] =  score
    return scores

def byScore(elem):
    return elem[1]

def searchScore(scores, name):
    return scores.get(name, 'Not recorded')
```

```
def main():
    scores = getScores()
    scoreList = list(scores.items())
    scoreList.sort()
    scoreList.sort(key = byScore, reverse = True)

    for s in scoreList:
        print('{} scored {} '.format(s[0], s[1]))

    while True:
        name = input("Enter name of student or
<ENTER> to end: ").capitalize()
        if name == '': break
        print(searchScore(scores, name))
```

# Multi-player dice game

```
Enter name: alan
Enter name: ben
Enter name: cindy
Enter name:
```

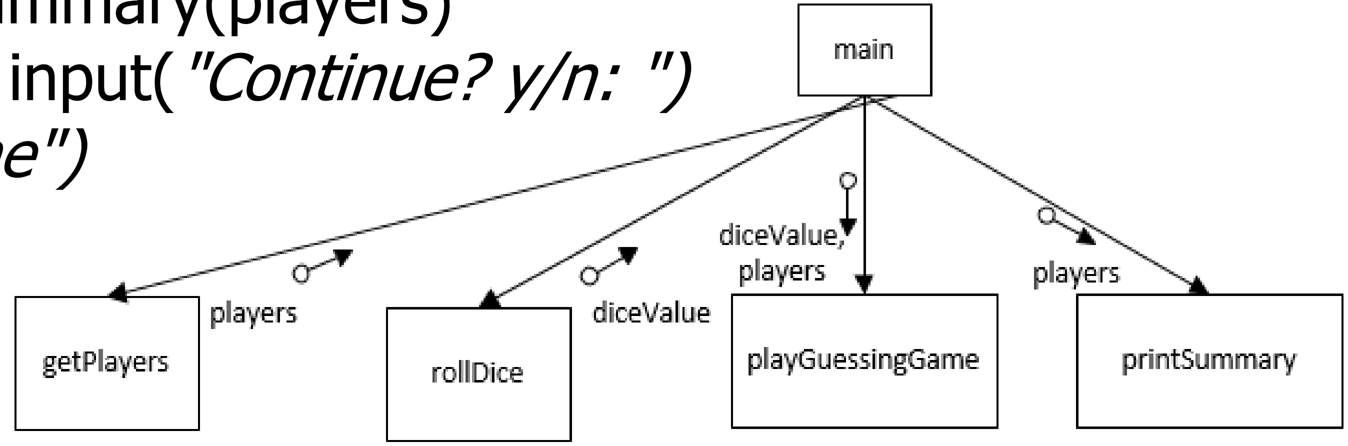
```
Try 1. Alan, enter guess: 1
Try 1. Ben, enter guess: 1
Try 1. Cindy, enter guess: 2
Alan,incorrect
Ben,incorrect
Cindy,incorrect
Try 2. Alan, enter guess: 3
Try 2. Ben, enter guess: 4
Try 2. Cindy, enter guess: 5
Alan, you got it!
Ben,incorrect
Cindy,incorrect
```

```
Alan won 1 game
Ben won 0 game
Cindy won 0 game
Continue? y/n: y
```

```
Try 1. Alan, enter guess: 4
Try 1. Ben, enter guess: 5
Try 1. Cindy, enter guess: 6
Alan,incorrect
Ben, you got it!
Cindy,incorrect
Alan won 1 game
Ben won 1 game
Cindy won 0 game
Continue? y/n: n
```

# Multi-player dice game

```
def main():
    players = getPlayers()
    playAgain = 'y'
    while playAgain[0].lower() in 'yY':
        diceValue = rollDice()
        playGuessingGame(players, diceValue)
        printScoreSummary(players)
        playAgain = input( "Continue? y/n:  ")
    print( "End game")
```



# Multi-player dice game

```
def getPlayers():
    players = {}
    while True:
        name = input('Enter name: ') .capitalize()
        if name == '': break
        players[name] = {'won': 0, 'guess': 0}
    return players
```

{ 'Alan': {'won': 0, 'guess': 0}, 'Ben': {'won': 0, 'guess': 0}, 'Cindy': {'won': 0, 'guess': 0} }

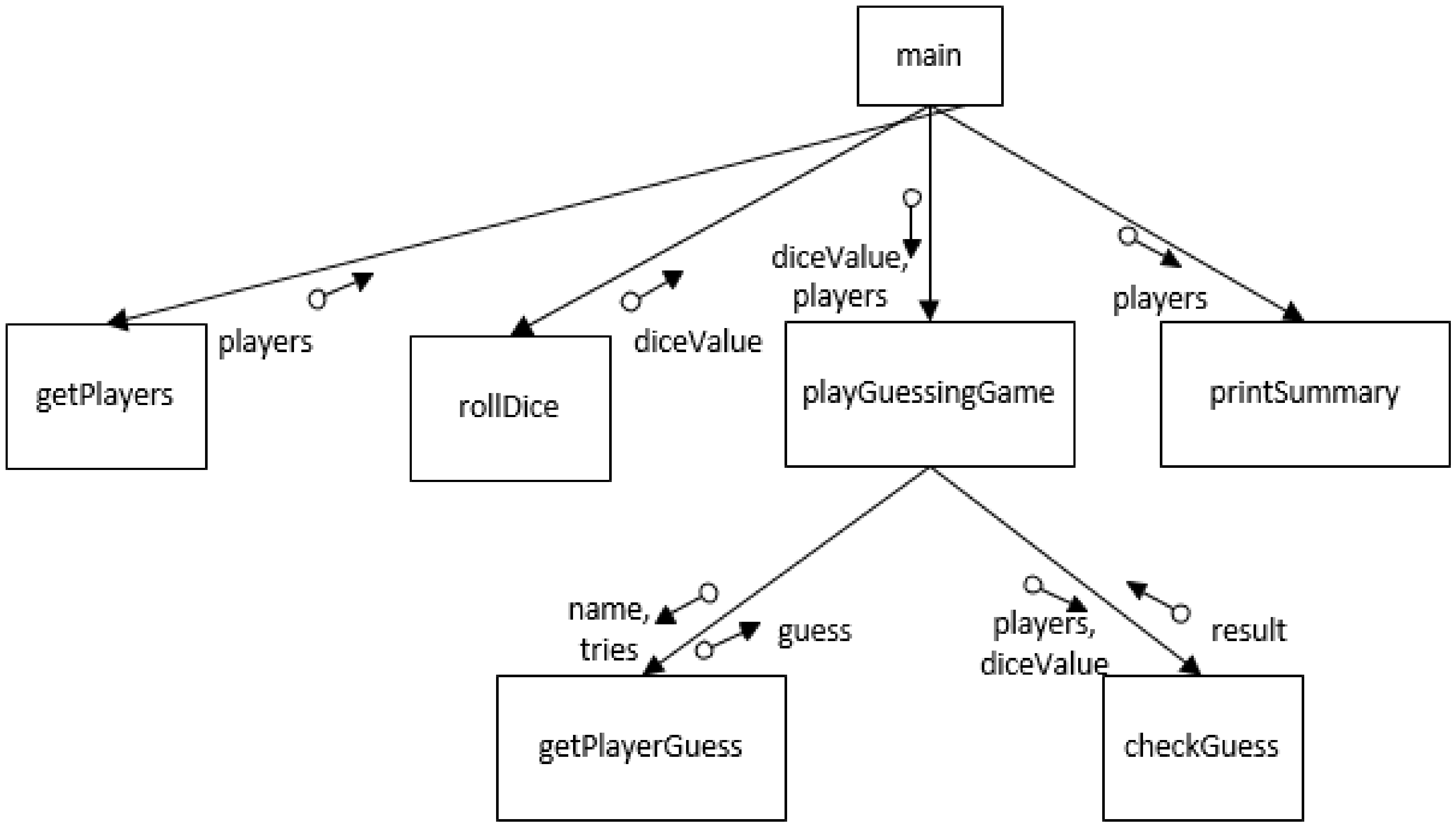
```
from random import randint
def rollDice():
    return randint(1, 6)
```

# Multi-player dice game

```
def playGuessingGame(players, diceValue):
    for tries in range(1,4):
        for k, v in players.items():
            v['guess'] = getPlayerGuess(k, tries)
        if checkGuess(players, diceValue):
            break
    else:
        print("Sorry, value is {}".format(diceValue))

def printScoreSummary(players):
    for k, v in players.items():
        print("{} won {} game{}".format(k, v['won'], '' if
v['won'] <2 else 's'))
```

# Multi-player dice game



# Multi-player dice game

```
def getPlayerGuess(name, tries):
    return int(input( "Try {}. {}, enter guess: ".format(tries,
name)))

def checkGuess(players, diceValue):
    correct = False
    for k, v in players.items():
        if diceValue == v['guess']:
            print("{}, you got it!".format(k))
            v['won'] += 1
            correct = True
        else: print("{},incorrect".format(k))
    return correct
```