

# PROBLEM SOLVING AND PROGRAMMING

## CSE1001

Prof. Tulasi Prasad Sariki

September 16, 2019

## PROBLEM

Write a kids play program that prints the capital of a country given the name of the country.

## PROBLEM

Write a kids play program that prints the capital of a country given the name of the country.

### PAC For Quiz Problem

Input	Processing	Output
A set of question/ answer pairs and a question	Map each question to the corresponding an- swer. Find the answer for the given question	Answer for the question

## PSEUDOCODE

```
READ num_of_countries
FOR i=0 to num_of_countries
    READ name_of_country
    READ capital_of_country
    MAP name_of_country to capital_of_country
END FOR
READ country_asked
GET capital for country_asked
PRINT capital
```

## Already we know

- To read values from user
- Print values
- We have to yet know to
  - Map a pair of values

## Already we know

- To read values from user
- Print values
- We have to yet know to
  - Map a pair of values

**Python provides Dictionaries to Map a pair of values**

## Introduction to Dictionary

- Pair of items
- Each pair has key and value
- Keys should be unique
- Key and value are separated by Colon(:)
- Each pair is separated by comma(,)

### Example:

```
dict = {'Alice' : 1234, 'Bob' : 1235}
```

## Properties of Dictionary

- Unordered mutable collections;
- Items are stored and fetched by key,
- Accessed by key, not offset position
- Unordered collections of arbitrary objects
- Variable-length, heterogeneous, and arbitrarily nestable



## Creating a Dictionary

- Creating an **EMPTY dictionary**

```
dictname = {}
```

- Example:

- Dict1 = {}
- MyDict = {}
- Books = {}

- Creating a **dictionary with items**

```
dictname = {key1:val1, key2:val2, ...}
```

- Example:

- MyDict = { 1 : 'Chocolate', 2 : 'Icecream' }
- MyCourse = { 'MS' : 'Python', 'IT' : 'C', 'CSE' : 'C++', 'MCA' : 'Java' }
- MyCircle = { 'Dad' : 9486028245, 'Mom': 9486301601 }

## Accessing Values

- Using keys within square brackets
- ```
>>> print MyDict[1]
'Chocolate'
```
- ```
>>> print MyCourse['CSE']
'C++'
```

## Accessing Values

- Using keys within square brackets
- ```
>>> print MyDict[1]
'Chocolate'
```
- ```
>>> print MyCourse['CSE']
'C++'
```

## Updating Elements

- **Update** by adding a new item (key-value) pair
  - **Modify** an existing entry
- ```
>>> MyDict[1] = 'Pizza'
>>> MyCourse['MCA'] = 'UML'
```

## Accessing Values

- Using keys within square brackets
- ```
>>> print MyDict[1]
'Chocholate'
```
- ```
>>> print MyCourse['CSE']
'C++'
```

## Updating Elements

- **Update** by adding a new item (key-value) pair
  - **Modify** an existing entry
- ```
>>> MyDict[1] = 'Pizza'
>>> MyCourse['MCA'] = 'UML'
```

## Deleting Elements

- **Remove an element** in a dictionary using the key
  - **Remove all the elements**
  - **Delete the dictionary**
- ```
>>> del MyCourse['IT']
>>> MyCourse.clear()
>>> del MyCourse
```

## Basic Operations

- >>> D = {'spam': 2, 'ham': 1, 'eggs': 3 }
- >>> len(D)                   # Number of entries in dictionary  
3
- >>> 'ham' in D               # Key membership test  
True
- >>> list(D.keys())           # Create a new list of D's keys  
['eggs', 'spam', 'ham']
- >>> list(D.values())  
[3, 2, 1]
- >>> list(D.items())  
[(*'eggs'*, 3), (*'spam'*, 2), (*'ham'*, 1)]
- >>> D.get('spam')           # key is Present  
2
- >>> print(D.get('toast')) → Return Nothing   # key is missing

## Update Method

- `>>> D → {'eggs': 3, 'spam': 2, 'ham': 1}`
- `>>> D2 = {'toast':4, 'muffin':5}`
- `>>> D.update(D2)`
- `>>> D`  
`{'eggs': 3, 'muffin': 5, 'toast': 4, 'spam': 2, 'ham': 1} # Unordered`

## Update Method

- >>> D → {'eggs': 3, 'spam': 2, 'ham': 1}
- >>> D2 = {'toast':4, 'muffin':5}
- >>> D.update(D2)
- >>> D  
{'eggs': 3, 'muffin': 5, 'toast': 4, 'spam': 2, 'ham': 1} # Unordered

## Pop Method

Delete and return value for a given key

- >>> D = {'eggs': 3, 'muffin': 5, 'toast': 4, 'spam': 2, 'ham': 1}
- >>> D.pop('muffin')  
5
- >>> D.pop('toast')  
4
- >>> D → {'eggs': 3, 'spam': 2, 'ham': 1}

## List Vs Dictionary

- `>>> L = []`
- `>>> L[99] = 'spam'`  
Traceback (most recent call last): File "*< stdin >*", line 1, in ?  
**IndexError: list assignment index out of range**
- `>>> D = {}`
- `>>> D[99] = 'spam'`
- `>>> D[99]`  
'spam'
- `>>> D`  
{99: 'spam'}



## Nesting in Dictionaries

- >>> jobs = []
- >>> jobs.append('developer')
- >>> jobs.append('manager')
- >>> rec = {}
- >>> rec['name'] = 'Bob'
- >>> rec['age'] = 40.5
- >>> rec['job'] = jobs
- >>> rec
- {'name': 'Bob', 'age': 40.5, 'job': ['developer', 'manager']}
- >>> rec['name']  
'Bob'
- >>> rec['job']  
['developer', 'manager']
- >>> rec['job'][1]  
'manager'

## Other Ways to Make Dictionaries

- `D = {'name': 'Bob', 'age': 40}`
- `D = {}`      **# Assign by keys dynamically**
- `D['name'] = 'Bob'`
- `D['age'] = 40`
- **# Creating a dictionary by assignment**
- `dict(name='Bob', age=40)`
- **# Creating dictionary with tuples form**
- `dict([('name', 'Bob'), ('age', 40)])`

## Comprehensions in Dictionaries

- `>>> D = {k: v for (k, v) in zip(['a', 'b', 'c'], [1, 2, 3])}`
- `>>> D`  
`{'b': 2, 'c': 3, 'a': 1}`
- `>>> D = {x: x ** 2 for x in [1, 2, 3, 4]}`
- # Or: `range(1, 5)`
- `>>> D`  
`{1: 1, 2: 4, 3: 9, 4: 16}`
- `>>> D = {c: c * 4 for c in 'SPAM'}`
- `>>> D`  
`{'S': 'SSSS', 'P': 'PPPP', 'A': 'AAAA', 'M': 'MMMM'}`
- `>>> D = {c.lower(): c + '!' for c in ['SPAM', 'EGGS', 'HAM']}`
- `>>> D`  
`{'eggs': 'EGGS!', 'spam': 'SPAM!', 'ham': 'HAM!'}`

## Initializing Dictionaries

- **# Initialize dict from keys**
- `>>> D = dict.fromkeys(['a', 'b', 'c'], 0)`
- `>>> D`  
`{'b': 0, 'c': 0, 'a': 0}`
- **# Same, but with a comprehension**
- `>>> D = {k:0 for k in ['a', 'b', 'c']}`
- `>>> D`  
`{'b': 0, 'c': 0, 'a': 0}`
- **# Comprehension**
- `>>> D = {k: None for k in 'spam'}`
- `>>> D`  
`{'s': None, 'p': None, 'a': None, 'm': None}`

## Dictionary methods

- `< dict >.items()`
  - displays the items in the dictionary (pair of keys and values)
- `< dict >.keys()`
  - display the keys in the dictionary
- `< dict >.values()`
  - displays the values in the dictionary
- `< dict >.pop(key)`
  - removes item with key value from the dictionary
- `< dict2 > = < dict1 >.copy()`
  - copies the items from dict1 to dict2
- `< dict >.clear()`
  - removes all the items from the dictionary

### Other methods

- `str(dict)` - produces printable string representation of a dictionary
- `len(dict)` - returns the number of items in the dictionary

- Dictionaries can replace elif ladder/switch-case
- `print ({1:'one',2:'two',3:'three',4:'four',5:'five'}) [choice]`
- if `choice = 3` then the code prints three

## EXERCISE 1:

Write a program to maintain a telephone directory of the employees of an organization. If the employee has more than one number store all the numbers. Write a program to print the mobile numbers given full or part of the name of the employee. Eg: Given name of the employee as 'John' the program must print phone numbers of 'John Paul' and 'Michel John'.

## EXERCISE 2:

Write a program to store the name of the players against each of a 20-20 cricket team. The program should print the name of the players given the team name.



*Thank you*