

Intro to Computer Science

Local Laboratory

*** Udacity – Intro to Computer Science**

Introduction

Unit 4: Responding to Queries

compose, 55	list-map, 86	stack, 24
fibonacci, 127, 131	list-product, 84	state, 45
filtering, 87	list-reverse, 89	String, 170
flattening lists, 90	list-search, 168	string, 19, 217
format, 94	list-sum, 84	subclass, 201
Fortran, 26	logarithm, 5	substitution, 181
frame, 182	logarithmic growth, 160	superclass, 201
function, 41		surface forms, 19
functional programming, 188	machine code, 38	Survivor, 71
	magnetic-core memory, 208	syllogisms, 237
games, 71	map, 86	Symbol, 94
global environment, 182	mcons, 186	symbol, 198
Goldbach's Conjecture, 242	means of abstraction, 21, 37	
grammar, 26	means of combination, 20	
growth rates, 139	measuring input size, 136	
	messages, 197	
halting problem, 241	methods, 200	tagged list, 94
higher-order procedure, 46, 55	Methods (Python), 219	tail recursive, 62
	MIT, ix, 207, 208	thunk, 230, 230
immutable, 186	mlist, 187	token, 213
imperative programming, 188, 188	mlist-append, 190	tokenizer, 213
inc, 55	modulo, 61	tracing, 69
incompleteness, 240	morpheme, 20	transitive, 153
indexed search, 169	mutable lists, 187	tree, 161
information, 3	mutable pair, 186	truth table, 109
information processes, 2	mutators, 179	truthiness, 21
interruption, 196		Turing Machine, 1
ints, 202	name, 44	types, 75
local variables, 199	natural language, 19	
lexer, 38, 211	natural languages, 36	universal comput
		universal program

Quiz: Data Structures

아래 중 어떤 자료구조가 우리의 검색엔진을 위한 인덱스를 표현하기에 가장 좋은 방법일지 선택하시오.

- ☐ [<keyword_1>, <url_1_1>, <url_1_2>, <keyword_2>, ...]
- ☐ [[<keyword_1>, <url_1_1>, <url_1_2>],
[<keyword_2>, <url_2_1>], ...
]
- ☐ [[<url1_1>, [<keyword_1>, <keyword_23>, ...]],
[<url2_1>, [<keyword_2>, ...]],
...]
- ☐ [[<keyword_1>, [<url_1_1>, <url_1_2>]],
[<keyword_2>, [<url_2_1>]], ...]

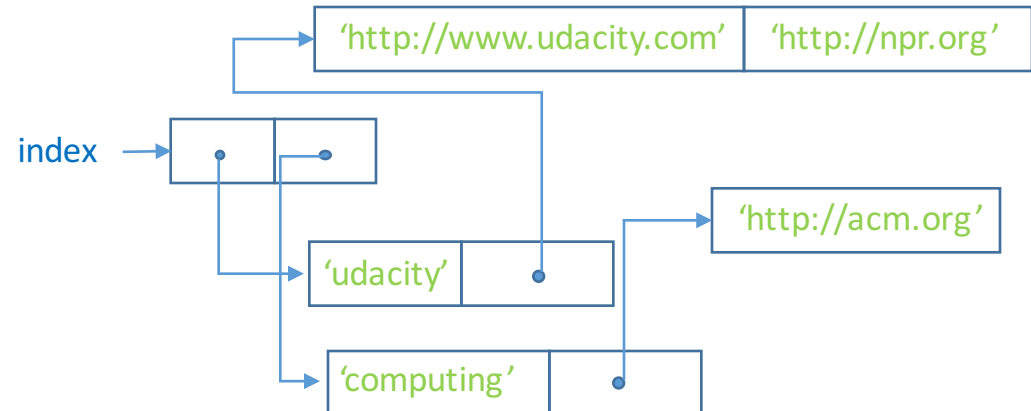
Quiz: Add to Index

아래 3개의 입력을 받는
`add_to_index`라는 프로시저를 정의하시오.

- an index [[<keyword>, [<url>, ...]], ...]
- a keyword string
- a url string


만약 keyword가 이미 index에 있다면, keyword와 연관된 url list에 url을 추가하시오.
만약 keyword가 index에 없다면 index에
다음과 같이 entry를 추가하시오: [keyword, [url]]

```
index = []  
add_to_index(index, 'udacity', 'http://udacity.com')  
add_to_index(index, 'computing', 'http://acm.org')  
add_to_index('udacity', 'http://npr.org')
```



Add to Index

```
def add_to_index(index, keyword, url):  
    for entry in index:  
        if entry[0] == keyword:  
            entry[1].append(url)  
        return  
    index.append([keyword, [url]])
```



Quiz: Lookup

다음 2개의 입력을 가지는
`lookup` 이라는 프로시저를 정의하시오.


- an index
- keyword

출력(리턴값)은 keyword와 연관된 url들의 list여야 한다.
만약 keyword가 index에 존재하지 않는다면, 출력값은 빈 list여야 한다.

`lookup(index, 'udacity') -> ['http://udacity.com', 'http://npr.org']`

Lookup

```
def lookup (index, keyword):  
    for entry in index:  
        if entry[0] == keyword:  
            return entry[1]  
    return []
```



Building the Web Index

`<string>.split()`
└───────────▶ `[<word>, <word>, ...]`

code

```
quote = "in Washington, it's dog eat dog. In academia, it's exactly the opposite. --- Robert Reich"  
print(quote.split())
```

result

```
['In', 'Washington,', "it's", 'dog', 'eat', 'dog.', 'In', 'academia,',  
"it's", 'exactly', 'the', 'opposite.', '---', 'Robert', 'Reich']
```


Quiz: Add Page to Index

다음 3개의 입력을 받는

`add_page_to_index` 라는 프로시저를 정의하시오.

- index
- url (string)
- content (string)

이 프로시저는 단어와 관련 있는 url list에 url을 추가하여,
page에서 찾은 모든 단어에 대한 index를 update한다.

code

```
index = []  
add_page_to_index(index, 'www.fake.com', 'This is a test')  
print(index)  
print(index[0])
```

result

```
[[ 'This', [ 'www.fake.com' ] ], [ 'is', [ 'www.fake.com' ] ], [ 'a', [ 'www.fake.com' ] ], [ 'test', [ 'www.fake.com' ] ]]  
[ 'This', [ 'www.fake.com' ] ]
```

Quiz: Add Page to Index

```
def add_page_to_index(index, url, content):  
    words = content.split()  
    for word in words:  
        add_to_index(index, word, url)
```

Beautiful Index

```
from bs4 import BeautifulSoup
```

```
def add_page_to_index(index, url, html ):  
    bs = BeautifulSoup(html, 'html.parser')  
    content = html.get_text()  
    words = content.split()  
    for word in words:  
        add_to_index(index, word, url)
```

Quiz: Finishing the Web Crawler

```
def crawl_web(seed):  
    tocrawl = [seed]  
    crawled = []  
    index = []  
    while tocrawl:  
        page = tocrawl.pop()  
        if page not in crawled:  
            html = get_page(page)  
            add_page_to_index(index, page, html)  
            union(tocrawl, get_all_links(html))  
            crawled.append(page)  
    return index
```

Dictionaries

String

'hello'

sequence of
characters

immutable

s[i]

i th character in s

~~s[i] = 'x'~~

List

['alpha', 23]

list of

elements

mutable

p[i]

i th element of p

p[i] = u

replace value of

i th element with u

Dictionary

{ 'hydrogen': 1,
 'helium': 2 }

set of <key, value> pairs

mutable

d[k] ← key

value associated with k in d

d[k] = v

update k -> v

Using Dictionaries

code

```
elements = {'hydrogen': 1, 'helium': 2, 'carbon': 6}
print(elements)
print(elements['hydrogen'])
print(elements['carbon'])
print(elements['lithium'])
```

result

```
{'hydrogen': 1, 'helium': 2, 'carbon': 6}
1
6
KeyError: 'lithium'
```

code

```
elements['lithium'] = 3
elements['nitrogen'] = 8

print(elements['nitrogen'])
elements['nitrogen'] = 7
print(elements['nitrogen'])
```

result

```
8
7
```

Quiz: Population

세계에서 가장 큰 도시들에 대한 정보를 제공하는 `population` 이라는 Dictionary를 정의하시오.
key는 도시의 이름(문자열)으로 하고,
연관된 value는 백만단위(millions)의 인구수로 하시오.

Shanghai	17.8
Istanbul	13.3
Karachi	13.0
Mumbai	12.5

code

```
population = {}  
population['Shanghai'] = 17.8  
population['Istanbul'] = 13.3  
population['Karachi'] = 13.0  
population['Mumbai'] = 12.5  
population['Charlottesville'] = 0.043  
  
print(population['Shanghai'])  
print(population['Charlottesville'])
```

result

```
17.8  
0.043
```

A Noble Gas

code

```
elements = {}  
elements['H'] = {'name': 'Hydrogen', 'number': 1, 'weight': 1.00794}  
elements['He'] = {'name': 'Helium', 'number': 2, 'weight': 4.002602,  
                  'noble gas': True}  
print(elements['H'])  
print(elements['H']['name'])  
print(elements['H']['weight'])  
print(elements['He']['noble gas'])  
print(elements['H']['noble gas'])
```

result

```
{'name': 'Hydrogen', 'number': 1, 'weight': 1.00794}  
Hydrogen  
1.00794  
True  
KeyError: 'noble gas'
```


Quiz: Modifying the Search Engine

list index를 Dictionary index로 변경할 때,
아래 우리가 만든 검색엔진의 프로시저 중에 변경되어야 하는 함수를 고르시오.

☐ get_all_links

☐ add_to_index

☐ crawl_web

☐ lookup

☐ add_page_to_index

```
def get_all_links(page):  
    links = []  
    while True:  
        url, endpos = get_next_target(page)  
        if url:  
            links.append(url) page = page[endpos:]  
        else:  
            break  
    return links
```

```
def crawl_web(seed):  
    tocrawl = [seed]  
    crawled = []  
    index = {}  
    while tocrawl:  
        page = tocrawl.pop()  
        if page not in crawled:  
            content = get_page(page)  
            add_page_to_index(index, page, content)  
            union(tocrawl, get_all_links(content))  
            crawled.append(page)  
    return index
```

```
def add_page_to_index(index, url, content):  
    words = content.split()  
    for word in words:  
        add_to_index(index, word, url)
```

```
def add_to_index(index, keyword, url):  
    for entry in index:  
        if entry[0] == keyword: entry[1].append(url)  
        return  
    # not found, add new keyword to index  
    index.append([keyword, [url]])
```

```
def lookup(index, keyword):  
    for entry in index:  
        if entry[0] == keyword:  
            return entry[1]  
    return None
```

Modifying the Search Engine

```
def crawl_web(seed):
    tocrawl = [seed]
    crawled = []
index = [] index = {}
    while tocrawl:
        page = tocrawl.pop()
        if page not in crawled:
            content = get_page(page)
            add_page_to_index(index, page,
content)
            union(tocrawl, get_all_links(content))
            crawled.append(page)
    return index
```

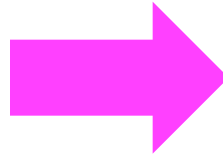
```
def add_page_to_index(index, url, content):
    words = content.split()
    for word in words:
        add_to_index(index, word, url)
```

```
def add_to_index(index, keyword, url):
    for entry in index:
        if entry[0] == keyword:
            entry[1].append(url)
    return
# not found, add new keyword to index
index.append([keyword, [url]])
```

```
def add_to_index(index, keyword, url):
    if keyword in index:
        index[keyword].append(url)
    else:
        # not found, add new keyword to index
        index[keyword] = [url]
```

Quiz: Changing Lookup

```
def lookup(index, keyword):  
    for entry in index:  
        if entry[0] == keyword:  
            return entry[1]  
    return None
```



```
def lookup(index, keyword):  
    if keyword in index:  
        return index[keyword]  
    else:  
        return None
```