

Jin-Soo Kim
(jinsoo.kim@snu.ac.kr)

Systems Software &
Architecture Lab.
Seoul National University

Dec 16 – 20, 2019

Introduction to Python



교재

- 데이터 과학을 위한 파이썬 프로그래밍
- 최성철 저
- 한빛 아카데미, 2019.



일정

		12/16 (Mon)	12/17 (Tue)	12/18 (Wed)	12/19 (Thu)	12/20 (Fri)	
오전	8	Introduction (Ch. 1)	What is Programming?	Functions (Ch. 5)	Dictionary (Ch. 7.4)	[실습]	
	9	Basic data types (Ch. 2, 3.1-3.3)	Lists (Ch. 3.4-3.5, 7.1-7.2)	File I/O (Ch. 12.2)			
	10	[실습]		Strings (Ch. 6)	Tuple, Set (Ch. 7.3)	Project	
	11		Control structures (Ch. 4)				
오후	1	[실습]	[실습]	[실습]	List comprehension (Ch. 8)		Project 발표 Wrap-up
	2						
	3					Lambda (Ch. 9.1)	
	4					Map/reduce (Ch. 9.2)	

About Me

- 김진수 (Jin-Soo Kim)
 - Professor @ Dept. of Computer Science & Engineering, SNU
 - Systems Software & Architecture Laboratory
 - Operating systems, Storage systems, Parallel and Distributed computing, Embedded systems, ...
- E-mail: jinsoo.kim@snu.ac.kr
- <http://csl.snu.ac.kr>
- Tel: 02-880-7302
- Office: SNU Engineering Building #301-520
- <https://github.com/snu-csl/pyrisc>: Educational RISC-V toolset in Python

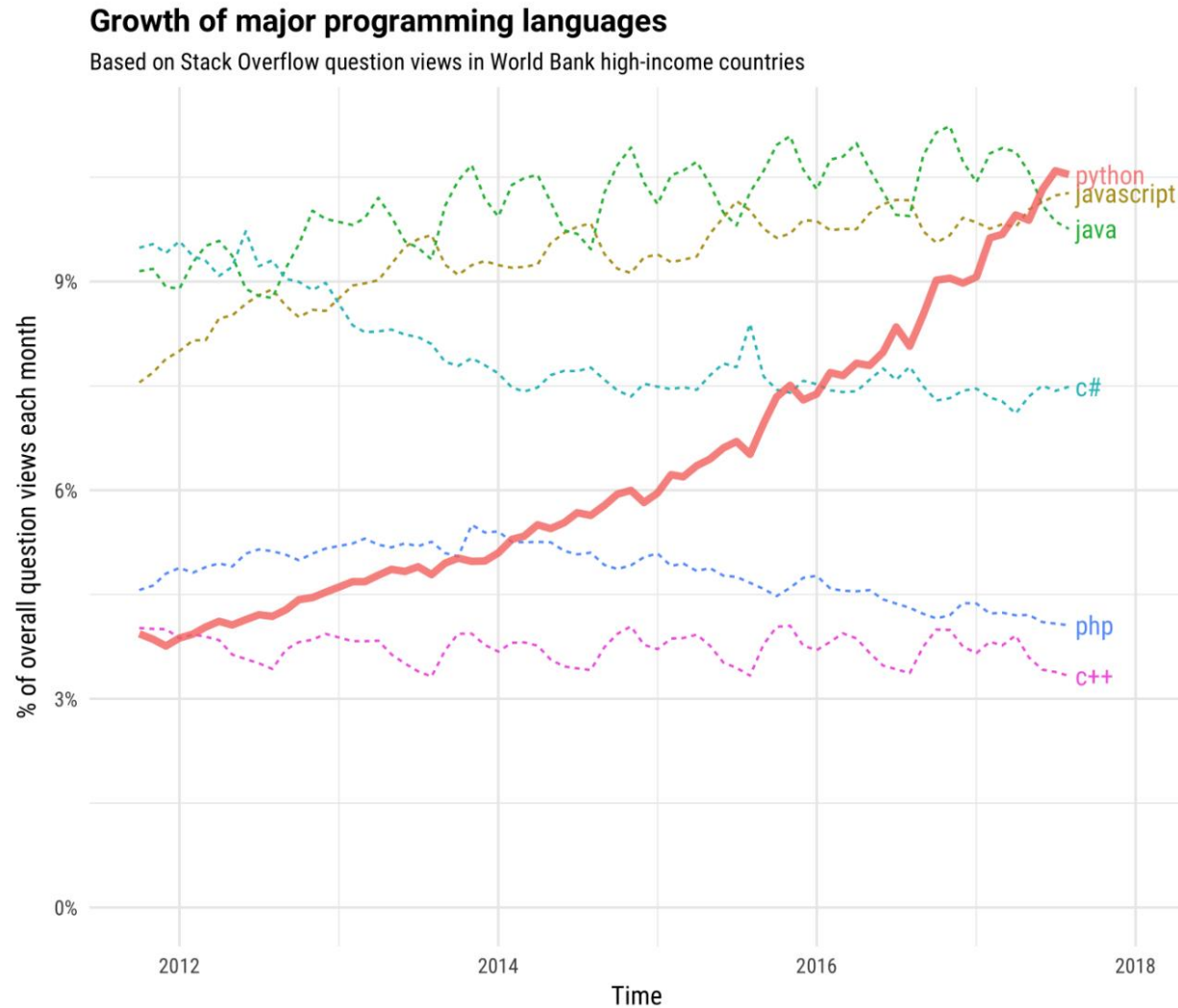
Python

Why Python?

- 1st place among the “Top Programming Languages” (IEEE Spectrum, 2019)
- “Fastest growing major programming language” (stackoverflow.com, 2019)
- The 2nd most popular programming language in GitHub (Nov. 2019)
- “The language of AI”

Rank	Language	Type	Score
1	Python	  	100.0
2	Java	  	96.3
3	C	  	94.4
4	C++	  	87.5
5	R		81.5
6	JavaScript		79.4
7	C#	   	74.5
8	Matlab		70.6
9	Swift	 	69.1
10	Go	 	68.0

Growth of Major Programming Languages



The Birth of Python

- Developed by Guido van Rossum in 1990

Over six years ago, in December 1989, I was looking for a “hobby” programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python’s Flying Circus).



Sounds Familiar?

- Unix developed by Ken Thompson in 1969

... It was the summer of '69. In fact, my wife went on vacation to my family's place in California.... I allocated a week each to the operating system, the shell, the editor, and the assembler, to reproduce itself, and during the month she was gone, it was totally rewritten in a form that looked like an operating system, with tools that were sort of known, you know, assembler, editor, and shell Yeh, essentially one person for a month.



Python Philosophy

- Beautiful is better than ugly
 - Explicit is better than implicit
 - Simple is better than complex
 - Complex is better than complicated
 - Readability counts
-
- “There is more than one way to do it” (Perl)
 - “There should be one – and preferably only one – obvious way to do it” (Python)

Python Goals

- “Computer programming for Everybody”
 - DARPA funding proposal
- An easy and intuitive language just as powerful as major competitors
- Open source, so anyone can contribute to its development
- Code that is as understandable as plain English
- Suitability for everyday tasks, allowing for short development times

Program like Plain English?

```
name = input('Enter file:')
f = open(name)

counts = dict()
for line in f:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

lst = list()
for key, val in counts.items():
    t = (val, key)
    list.append(t)

lst = sorted(lst, reverse=True)

for val, key in lst[:5]:
    print(key, val)
```

Compare with this:

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MAX_CHARS 26
#define MAX_WORD_SIZE 30

struct TrieNode
{
    bool isEnd;
    unsigned frequency;
    int indexMinHeap;
    TrieNode* child[MAX_CHARS];
};

struct MinHeapNode
{
    TrieNode* root;
    unsigned frequency;
    char* word;
};

struct MinHeap
{
    unsigned capacity;
    int count;
    MinHeapNode* array;
};

TrieNode* newTrieNode()
{
    TrieNode* trieNode = new TrieNode;

    trieNode->isEnd = 0;
    trieNode->frequency = 0;
    trieNode->indexMinHeap = -1;
    for( int i = 0; i < MAX_CHARS; ++i )
        trieNode->child[i] = NULL;

    return trieNode;
}

MinHeap* createMinHeap( int capacity )
{
    MinHeap* minHeap = new MinHeap;

    minHeap->capacity = capacity;
    minHeap->count = 0;

    minHeap->array = new MinHeapNode [ minHeap->capacity ];

    return minHeap;
}

void swapMinHeapNodes ( MinHeapNode* a, MinHeapNode* b )
{
    MinHeapNode temp = *a;
    *a = *b;
    *b = temp;
}

void minHeapify( MinHeap* minHeap, int idx )
{
    int left, right, smallest;

    left = 2 * idx + 1;
    right = 2 * idx + 2;
    smallest = idx;
```

```
if ( left < minHeap->count &&
    minHeap->array[ left ]. frequency <
    minHeap->array[ smallest ]. frequency
    )
    smallest = left;

if ( right < minHeap->count &&
    minHeap->array[ right ]. frequency <
    minHeap->array[ smallest ]. frequency
    )
    smallest = right;

if( smallest != idx )
{
    minHeap->array[ smallest ]. root->indexMinHeap = idx;
    minHeap->array[ idx ]. root->indexMinHeap = smallest;

    swapMinHeapNodes ( &minHeap->array[ smallest ], &minHeap->array[ idx ] );

    minHeapify( minHeap, smallest );
}
}

void buildMinHeap( MinHeap* minHeap )
{
    int n, i;
    n = minHeap->count - 1;

    for( i = ( n - 1 ) / 2; i >= 0; --i )
        minHeapify( minHeap, i );
}

void insertInMinHeap( MinHeap* minHeap, TrieNode** root, const char* word )
{
    if ( (*root)->indexMinHeap != -1 )
    {
        ++( minHeap->array[ (*root)->indexMinHeap ]. frequency );

        minHeapify( minHeap, (*root)->indexMinHeap );
    }

    else if( minHeap->count < minHeap->capacity )
    {
        int count = minHeap->count;
        minHeap->array[ count ]. frequency = (*root)->frequency;
        minHeap->array[ count ]. word = new char [strlen( word ) + 1];
        strcpy( minHeap->array[ count ]. word, word );

        minHeap->array[ count ]. root = *root;
        (*root)->indexMinHeap = minHeap->count;

        ++( minHeap->count );
        buildMinHeap( minHeap );
    }

    else if ( (*root)->frequency > minHeap->array[0]. frequency )
    {
        minHeap->array[ 0 ]. root->indexMinHeap = -1;
        minHeap->array[ 0 ]. root = *root;
        minHeap->array[ 0 ]. root->indexMinHeap = 0;
        minHeap->array[ 0 ]. frequency = (*root)->frequency;

        delete [] minHeap->array[ 0 ]. word;
        minHeap->array[ 0 ]. word = new char [strlen( word ) + 1];
        strcpy( minHeap->array[ 0 ]. word, word );

        minHeapify ( minHeap, 0 );
    }
}
```

```
}

void insertUtil ( TrieNode** root, MinHeap* minHeap,
                const char* word, const char* dupWord )
{
    if ( *root == NULL )
        *root = newTrieNode();

    if ( *word != '\0' )
        insertUtil ( &((*root)->child[ tolower( *word ) - 97 ]),
                    minHeap, word + 1, dupWord );
    else
    {
        if ( (*root)->isEnd )
            ++( (*root)->frequency );
        else
        {
            (*root)->isEnd = 1;
            (*root)->frequency = 1;
        }

        insertInMinHeap( minHeap, root, dupWord );
    }
}

void insertTrieAndHeap(const char *word, TrieNode** root, MinHeap* minHeap)
{
    insertUtil( root, minHeap, word, word );
}

void displayMinHeap( MinHeap* minHeap )
{
    int i;

    for( i = 0; i < minHeap->count; ++i )
    {
        printf( "%s : %d\n", minHeap->array[i].word,
                minHeap->array[i].frequency );
    }
}

void printKMostFreq( FILE* fp, int k )
{
    MinHeap* minHeap = createMinHeap( k );

    TrieNode* root = NULL;

    char buffer[MAX_WORD_SIZE];

    while( fscanf( fp, "%s", buffer ) != EOF )
        insertTrieAndHeap(buffer, &root, minHeap);

    displayMinHeap( minHeap );
}

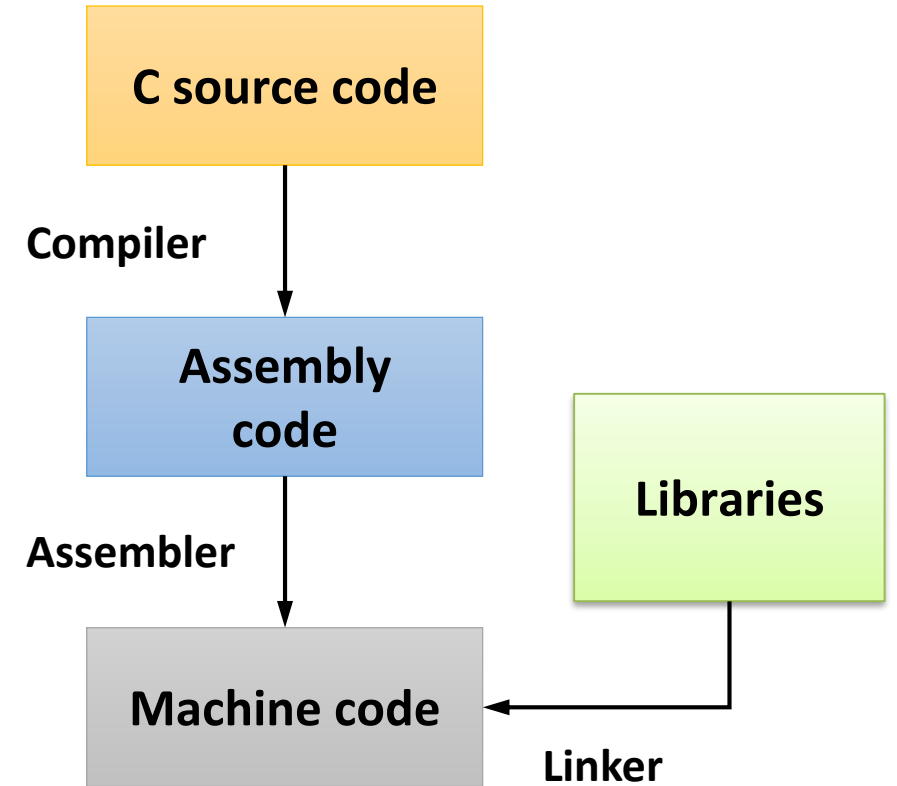
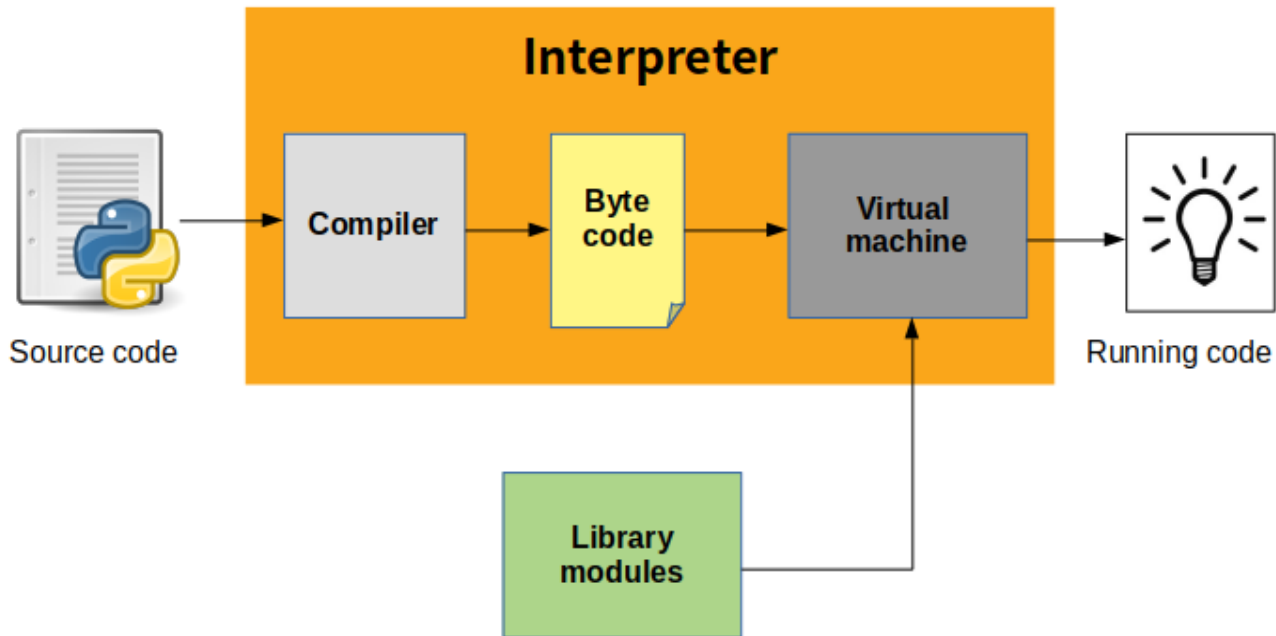
int main()
{
    int k = 5;
    FILE *fp = fopen ( "test.txt", "r" );
    if ( fp == NULL )
        printf ( "File doesn't exist " );
    else
        printKMostFreq ( fp, k );

    return 0;
}
```

Python Features

- Multi-paradigm programming language
 - Structured
 - Object-oriented
 - Functional
 - ...
- Highly extensible
 - Modules can be written in other languages such as C, C++, ...
- Interpreted
- “Pythonic”

Interpreted vs. Compiled



Python Versions

- Python 1.0 (1990)
- Python 2.0 (2000)
- Python 3.0 (2008) – Not backward compatible to 2.0

- The latest version: 3.8.0

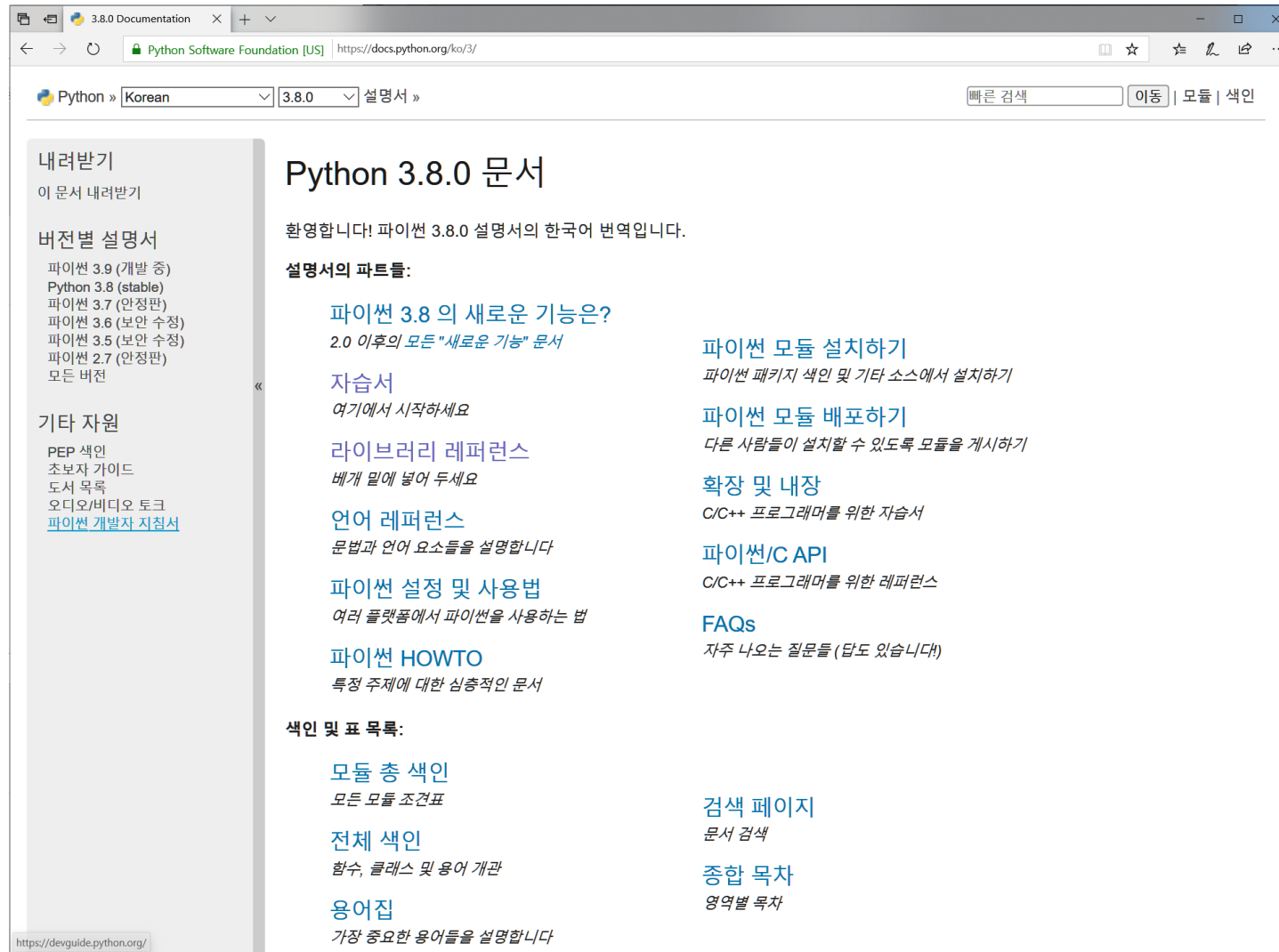
- Official homepage: <https://www.python.org>
- Tutorial: <https://docs.python.org/ko/3/tutorial>

Python Applications

- Machine Learning (TensorFlow, PyTorch, etc.)
- GUI Applications (Kivy, Tkinter, PyQt, etc.)
- Web frameworks (Django used by YouTube, Instagram, Dropbox)
- Image processing (OpenCV, Pillow, etc.)
- Web scraping (Scrapy, BeautifulSoup, etc.)
- Text processing (NLTK, KoNLPy, Word2vec, etc.)
- Test frameworks
- Multimedia
- Scientific computing and many more ...



https://docs.python.org



https://stackoverflow.com

A screenshot of a Google search result for the query "how to copy list in python". The search results show approximately 154,000,000 results in 0.48 seconds. A featured snippet is displayed, titled "To actually copy the list, you have various possibilities:". It lists three methods: 1. Using the builtin `list.copy()` method (available since Python 3.3): `new_list = old_list. ...`; 2. Slicing: `new_list = old_list[:]`; 3. Using the builtin `list()` function: `new_list = list(old_list)`. Below the snippet, there are links to "How to clone or copy a list? - Stack Overflow" and "Python List copy() - Programiz".

A screenshot of a Stack Overflow question titled "How to clone or copy a list?". The question was asked 9 years, 8 months ago and is active. It has 2301 views and 16 answers. The question text is: "What are the options to clone or copy a list in Python? While using `new_list = my_list`, any modifications to `new_list` changes `my_list` everytime. Why is this?". The question has tags for `python`, `list`, `copy`, and `clone`. The user who asked the question is Satya, with a reputation of 513. The question was edited on Feb 22 at 12:43. The question was asked on Apr 10 '10 at 8:49. The question has 558 votes. The question is marked as "First 25 Users Free". The question has 16 answers. The top answer is by user "aF" with 50.8k reputation, 39 votes, and 117 answers. The answer text is: "With `new_list = my_list`, you don't actually have two lists. The assignment just copies the reference to the list, not the actual list, so both `new_list` and `my_list` refer to the same list after the assignment." Below the answer, there is a link to "To actually copy the list, you have various possibilities:". At the bottom of the page, there is a footer with a cookie policy notice.

https://hashcode.co.kr

The image displays two side-by-side browser windows from the website hashcode.co.kr.

The left window shows a search results page for the query "python list". The page has a blue header with the "#ashcode" logo and navigation links: "가입하기", "로그인", and "의견 보내기". Below the header is a search bar containing "python list". The results section, titled "2488개의 검색 결과", lists several questions related to Python lists, each with a vote count, a "좋아요" (like) button, an "답변" (answer) button, and tags for "python" and "list".

The right window shows a code runner interface. The top bar includes a dropdown menu for "python3", buttons for "코드 초기화" (reset code) and "실행" (run), and a "RESULT" section. The code editor contains the following code:

```
1 print('hello world')
2
3
```

The output section shows the result of the code execution:

```
hello world
```

Below the output, it indicates the execution time: "실행시간: 16.46ms".

Welcome to the World of Spam!

