

### 파이썬 패키지 설치방법

#### 1. PyPI (Python Package Index)

- 파이썬 패키지 목록 사이트
- URL : <https://pypi.python.org/pypi>

1) 검색하고자 하는 패키지 이름 검색

The screenshot shows the PyPI website interface. At the top, there's a search bar with the text 'nltk' entered and a 'search' button. Below the search bar, the page title is 'PyPI - the Python Package Index'. The main content area includes a sidebar with links like 'PACKAGE INDEX', 'Browse packages', 'List trove classifiers', etc. The main body contains a description of PyPI, a 'Get Packages' section, 'Package Authors' information, and a table of recent updates.

Updated	Package	Description
2017-11-16	<a href="#">Scrapy_mingle 0.1.5</a>	Some useful tools for Scrapy
2017-11-16	<a href="#">captcha-manager 0.0.1</a>	CaptchaManager

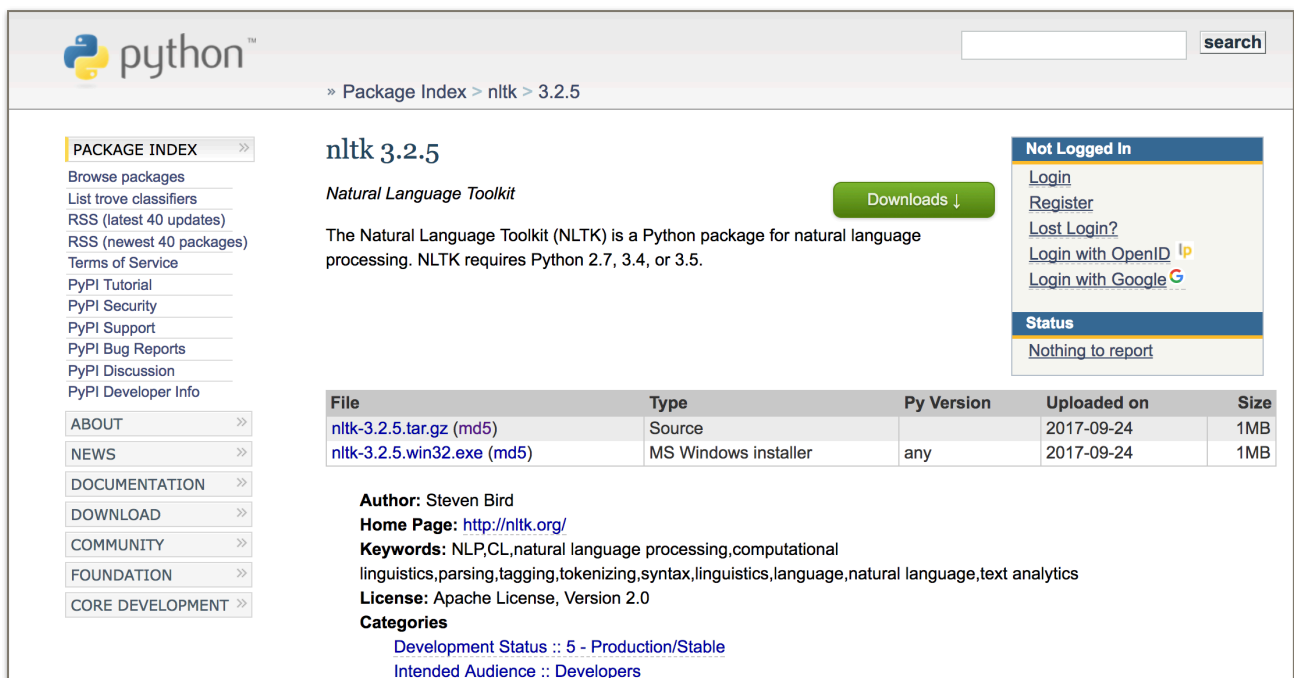
2) 검색결과 확인

- Weight : 이용자들이 많이 검색한 가중치
- Description : 패키지 설명

Package	Weight*	Description
<a href="#">nltk 3.2.5</a>	9	Natural Language Toolkit
<a href="#">nltk_tgrep 1.0.6</a>	9	tgrep2 Searching for NLTK Trees
<a href="#">nltkrest 0.12</a>	9	NLTK as a REST service
<a href="#">estnltk 1.4.1.1</a>	7	Estnltk — open source tools for Estonian natural language processing
<a href="#">rake-nltk 1.0.1</a>	7	Python implementation of the Rapid Automatic Keyword Extraction algorithm using NLTK
<a href="#">estnltk-textclassifier 1.2.2</a>	4	Machine learning software for organizing data into categories
<a href="#">jnlk 0.0.1</a>	4	Japanese Natural Language Processing toolkit
<a href="#">pysummarize 0.6.0</a>	4	Simple multi-language Python and NLTK-based implementation of text summarization
<a href="#">auto_tagify 1.4</a>	3	Auto-tags a selection of text and generates links to the tagged versions of the words
<a href="#">bluestocking 0.1.2</a>	3	An information extraction toolkit built on top of NLTK.
<a href="#">edeposit.amqp_errors 1.0.0</a>	3	eDeposit Plone app that helps with classifying of amqp errors

### 3) 검색결과 확인 > Downloads

- 패키지에 따라 설치방법이 다름



The screenshot shows the PyPI package page for **nltk 3.2.5**. The page layout includes a left sidebar with navigation links, a main content area with package details, and a right sidebar with user login options.

**PACKAGE INDEX** >>

- Browse packages
- List trove classifiers
- RSS (latest 40 updates)
- RSS (newest 40 packages)
- Terms of Service
- PyPI Tutorial
- PyPI Security
- PyPI Support
- PyPI Bug Reports
- PyPI Discussion
- PyPI Developer Info

**ABOUT** >>

**NEWS** >>

**DOCUMENTATION** >>

**DOWNLOAD** >>

**COMMUNITY** >>

**FOUNDATION** >>

**CORE DEVELOPMENT** >>

## nltk 3.2.5

*Natural Language Toolkit*

[Downloads ↓](#)

The Natural Language Toolkit (NLTK) is a Python package for natural language processing. NLTK requires Python 2.7, 3.4, or 3.5.

File	Type	Py Version	Uploaded on	Size
<a href="#">nltk-3.2.5.tar.gz (md5)</a>	Source		2017-09-24	1MB
<a href="#">nltk-3.2.5.win32.exe (md5)</a>	MS Windows installer	any	2017-09-24	1MB

**Author:** Steven Bird  
**Home Page:** <http://nltk.org/>  
**Keywords:** NLP, CL, natural language processing, computational linguistics, parsing, tagging, tokenizing, syntax, linguistics, language, natural language, text analytics  
**License:** Apache License, Version 2.0  
**Categories:**  
[Development Status :: 5 - Production/Stable](#)  
[Intended Audience :: Developers](#)

**Not Logged In**

- [Login](#)
- [Register](#)
- [Lost Login?](#)
- [Login with OpenID](#)
- [Login with Google](#)

**Status**

[Nothing to report](#)

## 1.1 Numpy 설치 예시

### 1) 패키지 사이트 접속

- <https://pypi.python.org/pypi/numpy/1.13.3>

### 2) 다운로드

- 클릭하여 설치

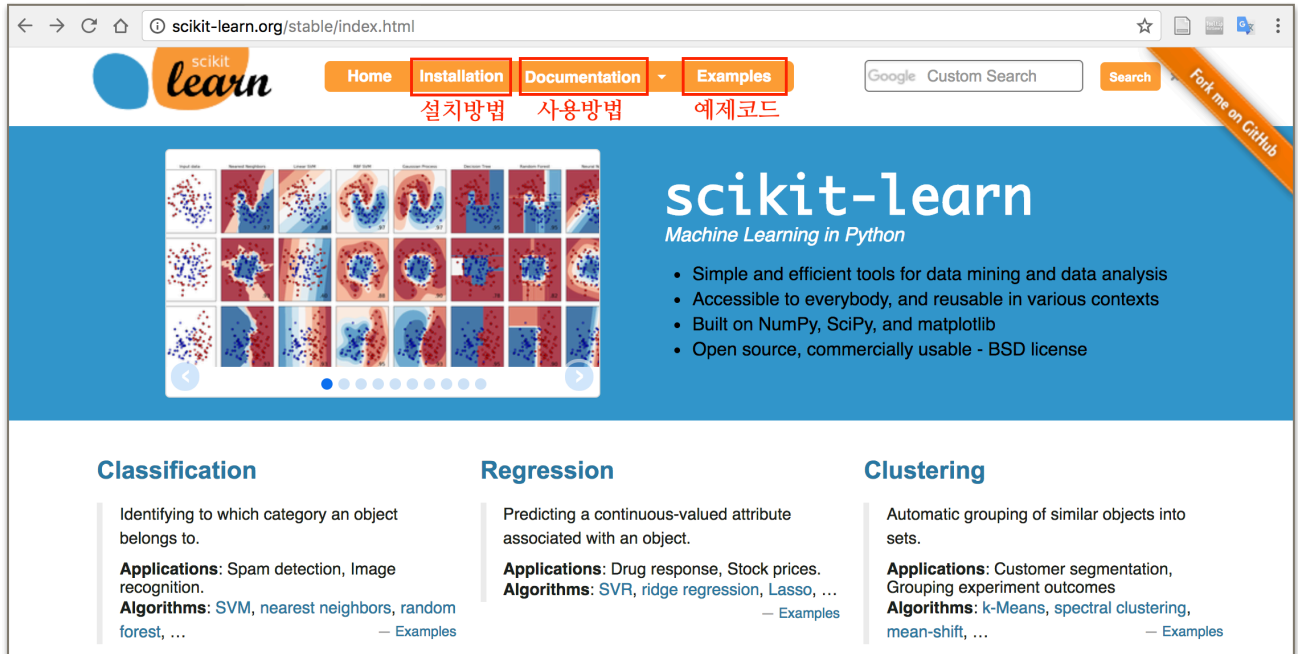
## 2. 패키지별 홈페이지 이용

유명한 패키지나 라이브러리는 별도의 홈페이지에서 설치 및 사용법 문서(Documentation) 제공

### 2-1. Scikit-learn 설치 예시

1) Scikit-learn 홈페이지 접속

- <http://scikit-learn.org/stable/index.html>



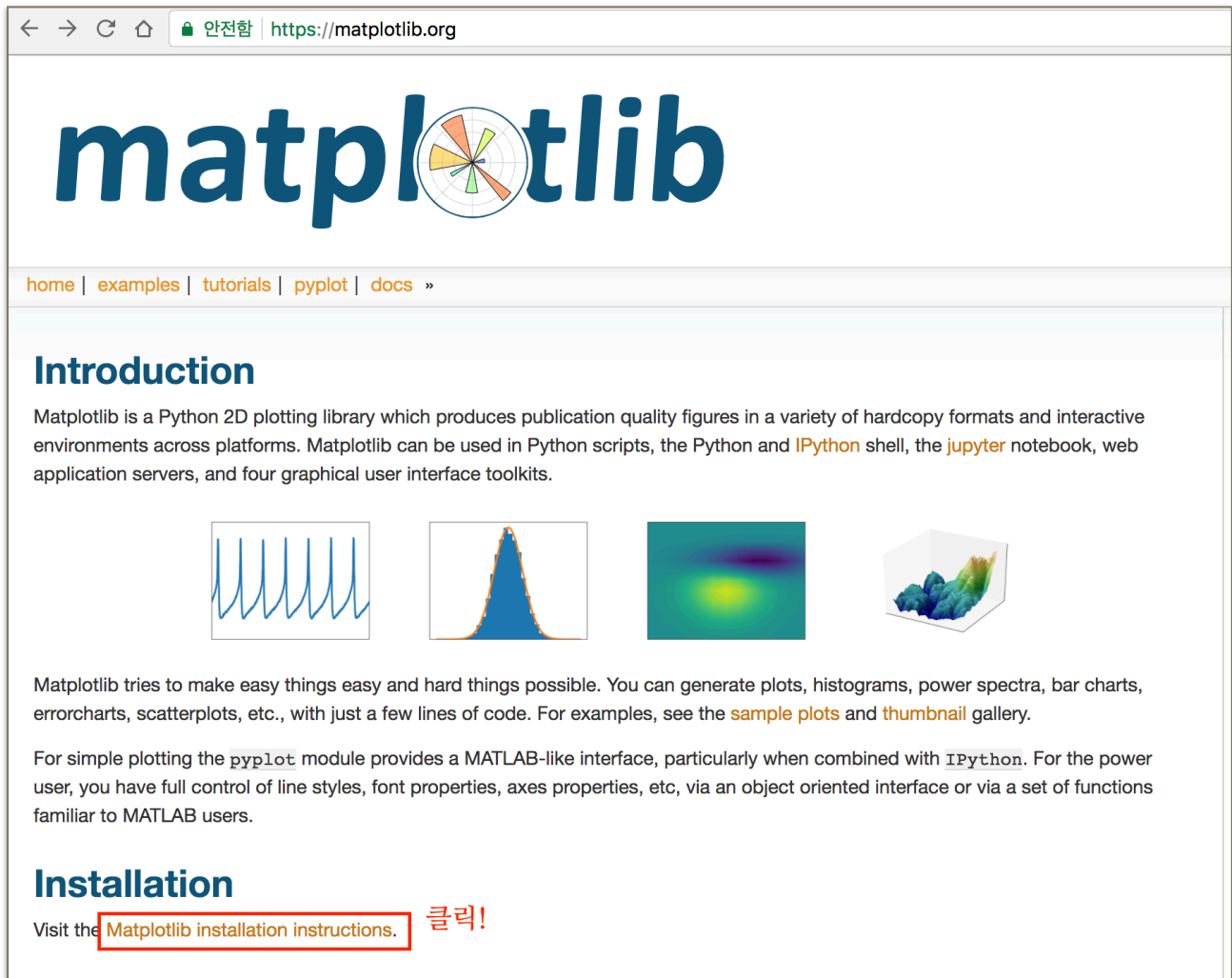
2) 설치 명령어 실행

```
$ pip install -U scikit-learn
```

## 2-2. Matplotlib 설치 예시

1) 홈페이지 접속

- <https://matplotlib.org/>
- 하단 [Matplotlib Installation instructions](#) 클릭



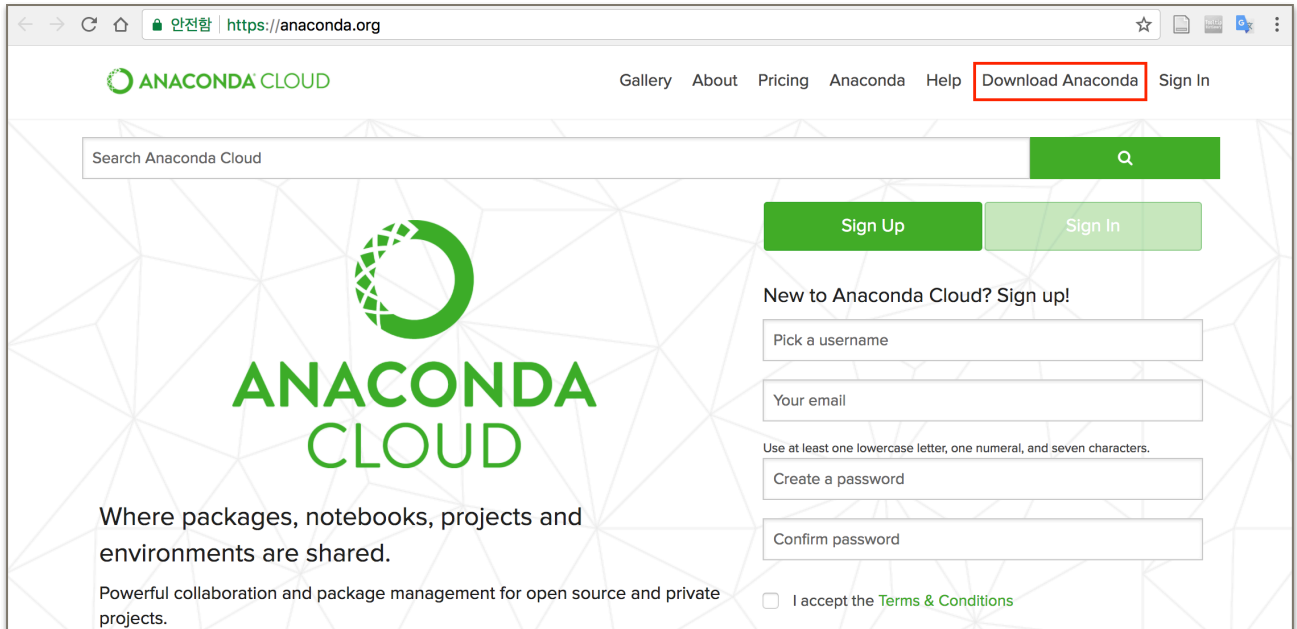
2) 설치 명령어 실행

```
$ pip install matplotlib
```

## 2-3. Anaconda 설치 예시

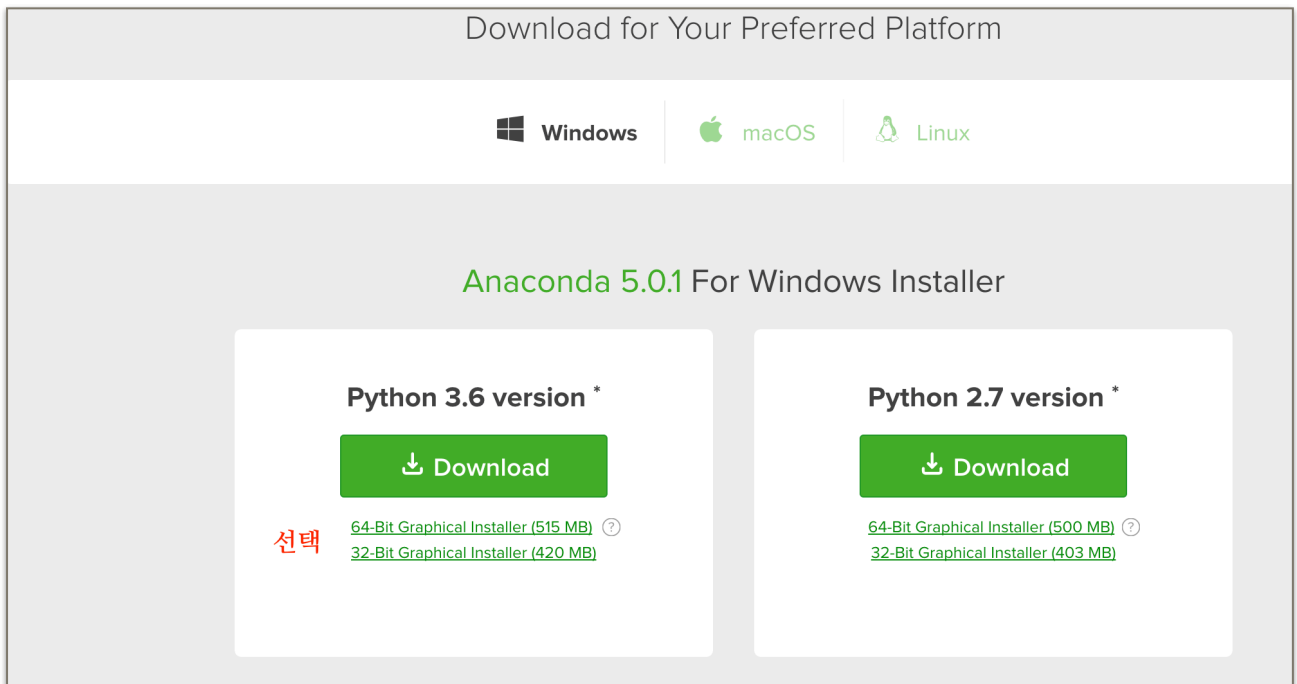
1) 홈페이지 접속

- <https://anaconda.org/>
- 상단 우측 Download Anaconda 클릭



2) Download 클릭

- exe 파일 클릭하여 설치



## 3. Docker 활용

각 패키지가 설치된 Docker 이미지 활용하기

## 실습1. 성별 분류기 만들기

코드 : gender\_identifier.py

**# 입력 단어에서 마지막 N개의 문자를 추출하는 함수 정의**

```
import random
```

```
from nltk import NaiveBayesClassifier
```

```
from nltk.classify import accuracy as nltk_accuracy
```

```
from nltk.corpus import names
```

**# 입력 단어에서 마지막 N개의 문자를 추출하는 함수 정의**

**# 입력단어에서 마지막 N개 글자를 추출해 자질로 사용**

```
def extract_features(word, N=2):
```

```
    last_n_letters = word[-N:]
```

```
    return {'feature': last_n_letters.lower()}
```

**#메인함수 정의, 사이킷런 패키지에서 학습데이터 추출 (이 데이터에는 남성과 여성으로 분류된 이름이 포함되어 있음)**

```
if __name__ == '__main__':
```

```
    # NLTK의 이름 정보를 이용해 학습셋 만들
```

```
    male_list = [(name, 'male') for name in names.words('male.txt')]
```

```
    female_list = [(name, 'female') for name in names.words('female.txt')]
```

```
    data = (male_list + female_list)
```

**# 난수 발생기에 시드 값 전달**

```
random.seed(5)
```

**# 데이터 뒤섞기**

```
random.shuffle(data)
```

**# 테스트 데이터 만들기**

```
input_names = ['Alexander', 'Danielle', 'David', 'Cheryl']
```

**# 학습셋과 테스트셋 비율 정하기**

```
num_train = int(0.8 * len(data))
```

**# 글자 수별로 성능 비교**

```
for i in range(1, 6):
```

```
    print("\nNumber of end letters:", i)
```

```
    features = [(extract_features(n, i), gender) for (n, gender) in data]
```

```
    #데이터를 학습셋과 테스트셋으로 분리
```

```
    train_data, test_data = features[:num_train], features[num_train:]
```

### #학습데이터를 사용해 나이브 베이즈 분류기 생성

```
classifier = NaiveBayesClassifier.train(train_data)
```

### # 분류기 정확도 계산

```
accuracy = round(100 * nltk_accuracy(classifier, test_data), 2)
```

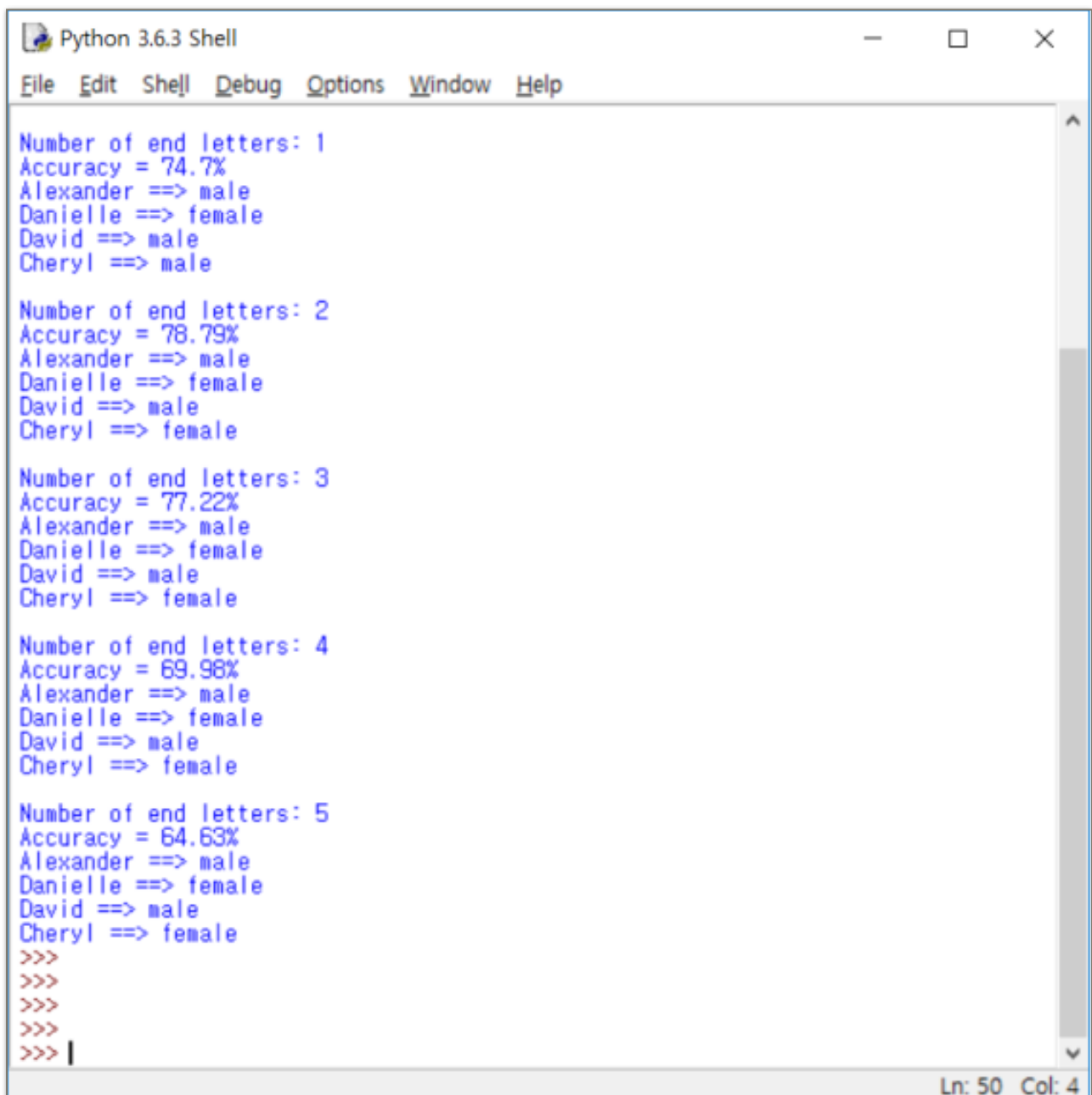
```
print('Accuracy = ' + str(accuracy) + '%')
```

### # 학습한 분류기로 결과 예측

```
for name in input_names:
```

```
    print(name, '==>', classifier.classify(extract_features(name, i)))
```

### 실행 결과



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help

Number of end letters: 1
Accuracy = 74.7%
Alexander ==> male
Danielle ==> female
David ==> male
Cheryl ==> male

Number of end letters: 2
Accuracy = 78.79%
Alexander ==> male
Danielle ==> female
David ==> male
Cheryl ==> female

Number of end letters: 3
Accuracy = 77.22%
Alexander ==> male
Danielle ==> female
David ==> male
Cheryl ==> female

Number of end letters: 4
Accuracy = 69.98%
Alexander ==> male
Danielle ==> female
David ==> male
Cheryl ==> female

Number of end letters: 5
Accuracy = 64.63%
Alexander ==> male
Danielle ==> female
David ==> male
Cheryl ==> female
>>>
>>>
>>>
>>>
>>> |

Ln: 50 Col: 4
```

## # 실습 포인트

### 1. Input 이름을 변경하여 다양한 결과 얻어보기

### 2. 학습셋과 테스트셋 비율을 변경하여 결과 분석하기

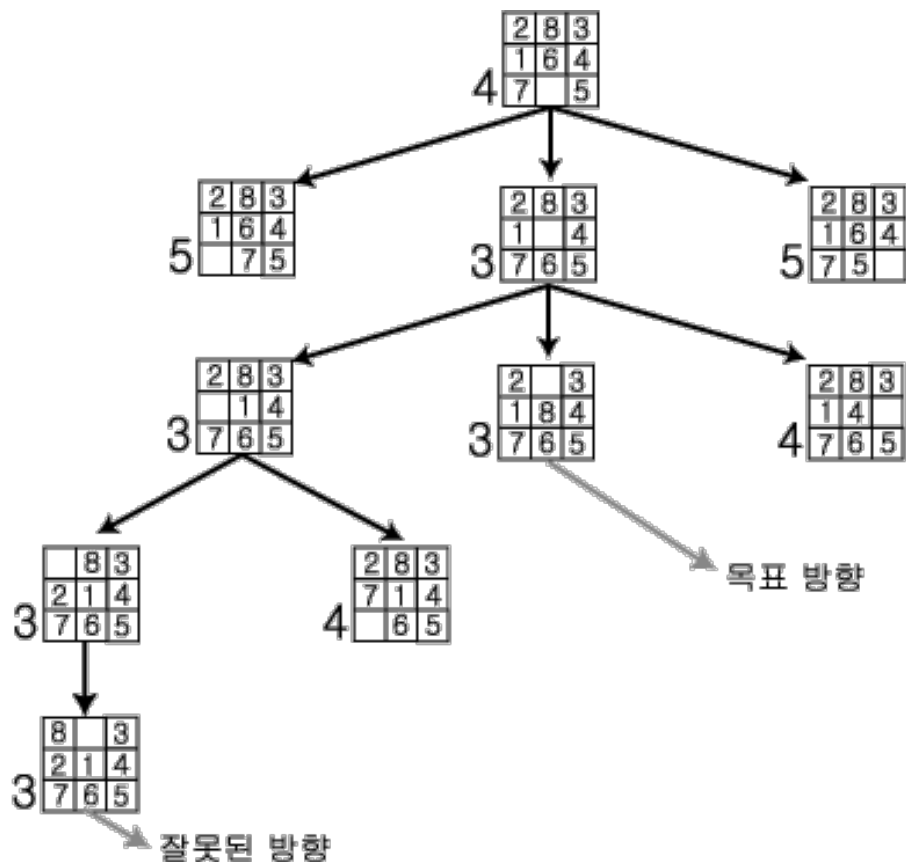
### 3. 휴리스틱 탐색(Heuristic search) 이해하기

- **정의** : 불충분한 시간이나 정보로 인하여 합리적인 판단을 할 수 없거나, 체계적이면서 합리적인 판단이 굳이 필요하지 않은 상황에서 사람들이 빠르게 사용할 수 있는 어림짐작의 방법 (출처: 위키피디아)
- **어원** : 라틴어의 'heuristicus' 와 그리스어 'heuriskein' 에서 시작. "찾아내다(find out)" 또는 발견하다(discover)"라는 의미
- **특징** :
  - 휴리스틱을 사용하여 과제를 단순화시킨 후 후기에 규범적(normative)인 의사결정 규칙을 사용하고, 단순한 과업 상황에서는 처음부터 최종 의사결정에 이르기까지 규범적 규칙을 이용한다는 가설은 Simon(1955)이 주창한 '제한된 합리성(bounded rationality)'에서 시작되었다. '제한된 합리성'이란 다양한 의사결정 상황에서 인간의 인지적인 한계로 인해 발생하는 의사결정 문제를 인지적 한계 안에서 다룰 수 있는 범위로 축소시키고, 간단해진 과업의 수행에 한해 규범적 규칙을 이용한다는 것을 의미한다.<sup>[2]</sup>

항목	내용
가용성 휴리스틱	<b>최근 발생한 사례만을 가지고 어떤 사건이 벌어지는 빈도수나 확률을 판단</b>  <b>사례</b> 1. 소설에서 7개의 철자로 된 단어 중에 -ing로 끝나는 단어는 몇 개인가? 2. 소설에서 7개의 철자로 된 단어 중에 여섯 번째 철자가 n인 단어는 몇 개인가?
대표성 휴리스틱	<b>어떤 집합에 속하는 임의의 한 특징이 그 집합의 특성을 대표한다고 간주해 빈도와 확률을 판단하는 방법</b>  <b>"린다 문제(Linda Problem)"</b> 먼저 린다의 특성이 다음과 같이 주어졌다고 하자. 린다는 31세의 독신 여성이며, 매우 머리가 좋고 본인 생각을 뚜렷하게 이야기 하는 성격이다. 그녀는 철학을 전공했으며, 사회정의와 인종차별에 깊이 관여하였고, 반핵 시위에도 참여하였다.  이러한 설명을 한 후 린다가 ① '페미니스트'일 확률 ② '은행원'일 확률 ③ '은행원이면서 페미니스트'일 확률을 예측하라고 했더니, 실험 결과 85%의 응답자가 ①>③>②의 순서로 린다의 직업 확률을 예측했다.  응답자들의 예측 결과는 논리적으로 맞지 않다. 먼저 현실에서는 세가지 선택지중 은행원의 수가 절대적으로 많기 때문에 ② '은행원'일 확률이 제일 높다. 또한 린다가 ③ '은행원이면서 페미니스트'일 확률은 ① '페미니스트'이거나 ② '은행원'일 확률의 교집합에 속하기 때문에 절대적으로 ①이나 ②보다 더 클 수 없다. ③은 ① '페미니스트'와 ② '은행원'의 특성을 동시에 갖고 있어야 하기 때문에 집합 오류(conjunction fallacy)가 발생한 것이다.  이런 결과는 사람들이 대표성 휴리스틱을 써서 판단하는 것을 보여준다. 린다의 특성으로 보아 '은행원'이라는 특성이 그녀를 대표하기 보다는 '페미니스트'이거나 '페미니스트이면서 은행원일 것'이라는 특성이 린다를 더 대표한다고 생각해 확률을 계산한 것이다.



	<p>어떤 사건이나 상황에 대해 판단을 할 경우 경험으로 형성된 감정에 따라 평가를 다르게 하는 것</p> <p><b>솔로빅 (Solvic, Monahan, &amp; MacGregor, 2000)의 실험</b>  우선 어떤 환자를 퇴원시킬지 고르는 결정 상황에서 피 실험자들에게 두 가지 소견서를 보여 주었다. 첫 번째 소견서에는 “이 환자와 유사한 환자들이 퇴원했을 때 나중에 폭력적인 행동을 할 확률이 20%”라고 설명되어 있었고, 두 번째 소견서에는 “이 환자와 유사한 환자 100명 중 20명이 퇴원한 후에 폭력적인 행동을 보였다”라고 설명되어 있었다.</p> <p>두 소견서는 사실상 같은 의미를 내포하고 있었으나, 첫 번째 소견서를 본 후 피 실험자들의 21%가 환자의 퇴원에 반대하였고, 두 번째 소견서의 상황에서는 41%나 반대했다. 이 사례는 사람들이 확률(20%)로 표현되는 것 보다는 빈도(20명)로 표현되는 것에 더욱 감정적인 반응을 한다는 것을 보여준다.</p>
<p><b>감정 휴리스틱</b></p>	<p><b>특정문제가 갖는 정보에 크게 구속되지 않고 다양한 문제에 적용가능한 상위수준의 발견적 기법</b></p> <p><b>특징</b>  메타 휴리스틱 기법에는 유전 알고리즘, 담금질 기법, 타부 서치가 있다.</p> <p>이들 기법들은 각기 다른 특성이 있지만, 개념과 이론이 단순하고 해공간의 탐색 능력이 우수하여 공학, 자연과학 뿐만 아니라 경영학, 사회과학 등의 최적화 분야 또는 의사결정분야에 응용 가능하다는 공통점이 있다.</p> <p>이들 기법은 각기 독립적으로 여러 분야에 적용될 수 있을 뿐만 아니라, 각 기법의 단점을 상호보완하면서 장점을 결합하여 함께 적용될 수 있다. 예를 들면 유전알고리즘의 해공간 탐색 능력과 타부 서치의 지역 최적해 탐색 능력의 강점을 결합하여 사용할 수 있다.</p> <p>이러한 해법들은 <b>조합최적화*</b> 문제로 해석되는 현실의 문제들을 해결하기 위해 다양한 영역으로부터 아이디어를 얻은 결과이다. 유전알고리즘은 자연의 진화과정과 유전법칙을 모방하였고, 담금질 기법은 금속의 표면처리 과정에서 냉각과정에 기초를 두었고, 타부서치는 인간이 기억을 하는 과정을 이용했다.</p> <p><b>*조합최적화</b> : 어려운 문제들은 보통 문제 공간이 크기 때문에 조합최적화 알고리즘은 문제 공간을 좁히거나 탐색 효율을 높이는 데 중점을 둔다. 인공지능, 수학 및 소프트웨어 공학에서 사용된다.</p>



<그림> 휴리스틱 기법을 이용한 퍼즐 탐색 (제자리에 있지 않은 타일의 개수 이용)

## 인지편향

휴리스틱을 이용하여 사물을 판단할 때 발생하는 오류들

### 1) 대조효과

‘대비효과’라고도 불리는 대조효과는 대상을 지각할 때, 사물을 독립적으로 지각하지 않고 시각적, 공간적으로 인접한 것과 대조하여 대상을 판단하는 것을 말한다. 때문에 실제보다 지각하려는 대상의 특성을 더 과장하거나 축소하여 생각하게 된다.

2005년에 고든 모스코비츠 교수가 두 집단의 실험참가자들 중 한 집단에게만 히틀러를 생각해보게 한 후 한 남자의 인상이 어떤지를 평가를 부탁하는 연구를 발표했다. 실험 결과, 히틀러를 생각했던 그룹의 참가자들은 그렇지 않은 참가자들보다 대상을 더 긍정적으로 평가했다. 이는 대상과 히틀러를 대조하여 생각했기 때문이었다.[15]

선택적 지각 오류[편집]

선택적 지각이란 외부 정보를 있는 그대로 받아들여 처리하지 않고, 자신의 가치관과 일치하거나 자신에게 유리한 정보만 선택적으로 받아들여 처리하려는 편향을 말한다. 선택적 지각이 일어날 때 사람들은 정보의 객관성을 중시하기보다 자신의 주관적인 가치에 따라 정보를 선택하려한다.

똑같은 운동경기를 보는데도 양쪽 팀의 응원석에서 서로 심판이 편파적이라고 비난하는 경우를 선택적 지각의 사례로 들 수 있다.[16]

### 2) 사후확신 편향

사후확신 편향은 다른 인지적 편향들과는 다르게 인지과정에서 일어나는 것이 아니라, 그 이후에 일어나는 편향이고 ‘사후인지 편향’, ‘사후해석 편향’라고도 불린다. 사람들은 어떤 일이 발생한 후에 마치 과거에도 현재의 결과를 알고 있었던 것처럼 기억을 재구성하기 때문에 이런 현상이 일어난다. 실제로 사건이 발생할 확률을 왜곡하고, 그 사건에 대해 자신이 판단했던 것의 기억을 왜곡하는 과정이 포함되어있다.

사후확신 편향은 1972년에 피쇼프 교수에 의해 연구된 바가 있다. 실험참가자들에게 리처드 닉슨 대통령의 베이징 방문이 어떤 외교성과를 거둘지에 대해 예상하도록 한 후, 베이징 방문 이후 실험참가자들의 이야기를 들었다. 당시 닉슨은 강경한 반공주의자였고, 미국이 중국을 외교적으로 인정하지 않은 상태였기 때문에 참가자들은 외교성과를 좋게 평가하지 않았었지만, 닉슨이 예상외로 좋은 성과를 거두고 귀국하자 그들은 자신이 그럴 줄 알았다는 듯이 말했다.[17]

### 3) 결합 오류

결합 오류란 단일 사건의 발생 확률이 실제로 더 높음에도 불구하고 두 사건이 동시에 발생하는 경우의 확률을 더 높다고 생각하는 것을 말한다. 이러한 오류가 발생하는 이유는 사람들이 정보가 구체적인 쪽을 더 실제에 가깝다고 믿는 경향을 가지고 있기 때문이다. 결합오류는 ‘연결오류’, ‘연합오류’라고 불리기도 한다.

위에서 설명한 린다 문제를 결합 오류의 대표적인 예시로 들 수 있다.[18]

### 4) 기저율 무시

기저율 무시란 어떤 사건이 발생할 확률을 추정할 때 기본적으로 판단이나 의사결정에 필요한 사건들과의 선후관계 및 사건들의 상대적 빈도(즉, 기저율)를 고려하지 않고 가용한 정보를 근거로 통계적 확률과 상반되는 판단을 내리는 현상을 말한다. 이는 사람들이 기저율을 무시하고 특정 사건의 발생 가능성을 더 과대평가하여 판단하기 때문에 일어난다. 기저율을 고려하여 사건 발생 확률을 계산할 때에는 베이즈 정리를 활용한다.

기저율 무시와 관련해서 카너먼과 트버스키가 1983년에 한 가지 실험을 한 적이 있다. 차 색깔에 따라 블루와 그린이라고 불리는 두 택시회사가 있는데 택시의 85%는 블루, 15%는 그린회사 소속이다. 한밤중에 택시가 사고를 냈는데 목격자는 그 택시가 그린택시라고 말했다. 목격자의 색깔 구별능력 정확도는 80%일 때 사고를 낸 택시가 그린 회사

의 택시일 가능성이 얼마인지 묻는 실험이었다. 사람들은 80%라고 대답했지만 실제 가능성은 41%정도였다. 사람들이 가능성을 판단할 때, 85%가 블루, 15%가 그린이라는 기저율은 무시하고 판단을 내린 것이다.[19]

#### 5) 평균으로의 회귀 무시

평균으로의 회귀 무시는 좋은 결과를 보이면 그 이후에도 똑같은 결과가 있을 것이라 기대하는 현상을 말한다. 만약 기대한 것과 결과가 다르면 사람들은 인위적인 이유로 결과를 설명하려고 하는데 이때 사람들은 자연적인 상태에서는 최고값이나 최저값이 지속되지 않고 평균값으로 돌아간다는 사실을 무시한 것이다.

야구선수가 데뷔 첫 해에 아주 좋은 성과를 거둔 후 2년차에 좋지 않은 성적을 보이면 이 야구선수에게 정신적으로 문제가 생겼거나, 긴장감이 떨어졌다는 등의 다른 인위적 변수를 들어 설명하려고 한다. 사실 이것은 1년차에 너무 잘 해서 2년차에는 평균으로 돌아간 것이 퇴보한 것처럼 보이는 것이다.[20]

#### 6) 측면별 제거

측면별 제거는 어떤 수행 과제가 복잡한 경우에 대상의 전체 속성을 골고루 알아보기보다, 자신이 중요시하는 몇 가지의 속성만을 고려하여 결정을 내리는 것을 말한다. 이 때 자신이 선택한 속성을 제외한 나머지 요소들을 무시하거나, 잊어버리는 경우가 많다. 과제 수행 시에는 최고 대안을 선택하는 절대적인 속성이 없기 때문에 선택 결과가 개인 또는 상황에 따라 달라지기 쉽다는 특징이 있다.

어떤 물건을 구매할 때, 많은 요소를 종합적으로 알아 본 후에 구매를 진행해야함에도 불구하고, 사람들은 자신의 상황에서 고려해야 할 속성들만 평가한 후에 결정을 내리는 것이 측면별 제거가 일어나는 예시이다.[21]

#### 7) 확률 무시

확률 무시란 불확실한 상황에서 결정을 내릴 때, 확률을 완전히 무시하거나 예상 확률과 관련된 처리 규칙을 위반하는 현상을 말한다. 사람들의 인지는 긍정적인 결과가 제시되었을 때 확률을 쉽게 무시한다.

확률무시의 사례로는 1993년 미국 펜실베이니아 대학의 조너선 바론 교수가 진행한 실험이 있다. 연구진이 아이들에게 “수잔은 안전띠를 매야 한다, 제니퍼는 매지 않아야 한다.”라는 시나리오를 말해주었을 때 그들은 안전띠를 매야 한다고 말했지만, 이후 “사고가 나서 차가 물속으로 들어갔을 때나 불이 났을 때 안전띠 때문에 빠져나오지 못했다는 뉴스를 들은 기억이 있다고 제니퍼가 말했다.”라고 하자 아이들은 안전띠를 매지 말아야 한다고 했다. 번갈아가며 똑같은 이야기를 하자, 아이들은 계속 의견을 바꿨다. 아이들이 판단을 할 때 확률 자체를 전혀 고려하지 않았음을 보여주는 실험이었다.[22]

이외에도

#### 8) 확증 편향,

#### 9) 자기 귀인 편향,

10) 도박사의 오류 등의 편향이 있다.

## 휴리스틱 관련 이론

### 1) 전망 이론

전망 이론 (Prospect theory) 이란, 위험상황에서의 의사결정 이론이다. 심리학자인 카네만 (Daniel Kahneman) 과 트버스키 (Amos Tversky) 는 기대효용론이 보여주는 이론과 실제 상황 사이의 모순을 비판하며 새로운 대안 이론인 전망 이론을 제시하게 된다. 그들이 1979년도 *Econometrica* 에 발표한 논문이 전망 이론 발전의 시작이었으며 전망 이론은 기대효용 이론이 설명하는 선택의 합리성과 일관성을 비판하는 이론으로 사회과학분야에서 폭 넓게 받아들여진다.[23][24]

### 메타 휴리스틱

휴리스틱 기법은 해결해야 하는 상황마다 그 상황에 맞는 경험이나 직관을 이용해 해결해야하는 어려움이 있다. 그러므로 특정 문제가 갖는 정보에 상대적으로 덜 구속되며 다양한 문제에 적용할 수 있는 상위수준의 발견적 기법, 메타 휴리스틱이 사용된다. 메타 휴리스틱에는 유전알고리즘, 시뮬레이티드 어닐링, 터부탐색 등이 있다.

#### 1) 유전알고리즘(GA, genetic algorithms)

유전알고리즘은 메타휴리스틱의 고전이며 진화알고리즘 등 여러 이론의 바탕이 되었으며 1975년 Holland가 제안하였다. GA는 진화이론에서 사용하는 개념과 용어를 사용하고 있는데, 이는 유전자에서 발생하는 진화 현상을 모방하여 만들어졌기 때문이다. GA는 우선 가능해의 집합(군집)을 형성하여 그 집합에 목적함수와 연관된 적합도 평가 후, 개체의 적합도에 따라 자연선택을 확률적으로 이루어지게 하여 새로운 집합을 만든다. 그 후 집합의 개체들 간 유전연산(이종교배, 돌연변이)을 적용, 새로운 종들을 만든 후 그 종들의 적합도를 평가한다. 그리고 다시 자연선택 해서 새로운 무리를 형성하는 과정을 반복한다. 이 반복되는 과정을 통해 자연에서의 진화현상에서 뛰어난 종들이 살아남듯, GA도 마찬가지로 뛰어난 해들이 남을 것이라 기대할 수 있다. 실제 GA를 적용할 때는 명확하게 설정해야 할 작업들이 많아지게 되고, 이에 따라 적용의 융통성도 많아진다. 스키마이론을 통하여 부분적으로 설명되고는 있으나, 아직 GA가 최적해로 수렴한다는 것은 완전하지 못하다. 그럼에도 사람들이 많이 관심을 가지는 이유는 진화현상이 자연계에서 오랜 기간 성공적으로 수행되고 있기 때문이다.[26]

실험 참가자에게 다음 문제에서 A,B 중 선택 후, 이어지는 문제에서 C,D중 선택하도록 했다.

문제 1 A: 300만원 받기

B: 400만원 받을 확률 0.8

문제 2 C: 300만원 잃을 확률 0.25

D: 400만원 잃을 확률 0.2

실험 결과 문제1에서는 A를 80%의 사람이 택하고 문제2에서는 D를 택한 사람이 65%였다. 이는 기대효용론과 맞지 않는데 이것은 사람들이 손해의 경우 위험을 받아들이고자 하고, 이득의 경우 위험을 회피하고자 하기 때문이다.

비슷한 예로,

문제 3 A: 300만원 받을 확률 0.9

B: 600만원 받을 확률 0.45

문제 4 C: 300만원 잃을 확률 0.002

D: 600만원 잃을 확률 0.001

실험 결과 참가자의 86%가 A를 선택, 73%가 D를 선택하였다. 이 또한 기대효용론과 맞지 않는데 이익의 경우 확률이 높은편을 선호, 손해의 경우 확률이 낮은편을 선호 하였다.

문제 1~4 경우 양의 상황일 때와 음의 상황일 때가 서로 반사된 이미지를 가진다. 이를 "반사효과"라고 한다.

확률	이익	손실
중간~높음	위험 회피	위험 감수
낮음	위험 감수	위험 회피

## 2) 시뮬레이티드 어닐링(SA, simulated annealing)

시뮬레이티드 어닐링(SA)는 열탕안에 고체를 넣어서 액체가 되도록 가열한 후 열탕 온도를 천천히 낮춰서 최적으로 안정된 결정인 고체를 얻는 어닐링 과정과 비슷한 메트로폴리스 알고리즘을 이용하여 최적화 문제에 적용시킨 방법이다. 물체의 상태를 측정 후  $x$ 로 가정하고 그 때의 에너지를 목적함수로 가정하여 에너지 수준이 최저점 상태의 결정을 얻으면 목적함수의 최소화가 되는 것으로 보는 것이다. SA에는 어닐링처럼 온도  $T$ 를 고정하고 해를 조금 변동하여서 보다 더 작은 목적 값인 우수한 해가 얻어지면 해를 이동시키고 보다 더 큰 목적 값이 얻어지면 확률  $\exp \Delta$ 로 이동하는 절차를  $L$ 회(충분히 많이) 반복하여 안정 상태로 만든다(내부 루프). 그 후, 온도  $T$ 를 조금 낮추고 같은 과정을 되풀이 하여(외부 루프) 점점 전체적인 최소 점으로 가도록 실험한다. 이때 온도를 지나치게 내리게 된다면 지역 최소 점에 빠질 수도 있으므로 주의한다. 종료 하는 조건은 온도가 적절히 낮은 경우와 오랜 시간 목적 값의 변화가 없을 때를 고려하여 적절히 조정한다. SA는 매개변수가 간단하고 이론적 수렴성이 증명되어 있으며, 현상과 최적화 문제에서 관련성이 대체로 명확하다는 장점이 있다. 또, 성능을 높이기 위한 많은 이론들이 연구되었고, CAD 등의 실용적인 문제에서 이미 효과를 인정받고 있다.[26]

## 3) 터부탐색(TS, tabu search)

터부탐색은 자연계 보다는 인공지능 연구에서 발전 과정을 알 수 있다. TS는 기존의 메타휴리스틱의 방법들을 종합적으로 구성한 것이라 볼 수 있다. TS도 유전 알고리즘처럼 가능해의 집단  $V$ 를 고정하지만 새로운 해를 만들 때 터부조건이라는 기억장치를 유지해서 앞서 나왔던 열등한 해를 고려하지 않고 열망조건이라는 기억장치를 생성하여 우수한 해의 등장 가능성을 높이는 특징이 있다. 터부탐색 또한 기존의 형태에서 다양화와 집중화를 고려하여 다양한 이론적 변종이 가능하다.[26]

#### 4. 파이썬 사전(Dictionary)과 리스트(List) 개념 익히기

- 입력된 단어에서 마지막 N개의 문자를 추출하는 함수 이해하기

#### 리스트(List)

```
a = [1, 2, 3]
print(a)
print(a[0])
print(a[0] + a[2])
print(a[-1], '\n')
```

```
a = [1, 2, 3, ['a', 'b', 'c']]
print(a)
print(a[-1][0])
print(a[-1][1])
print(a[-1][2], '\n')
```

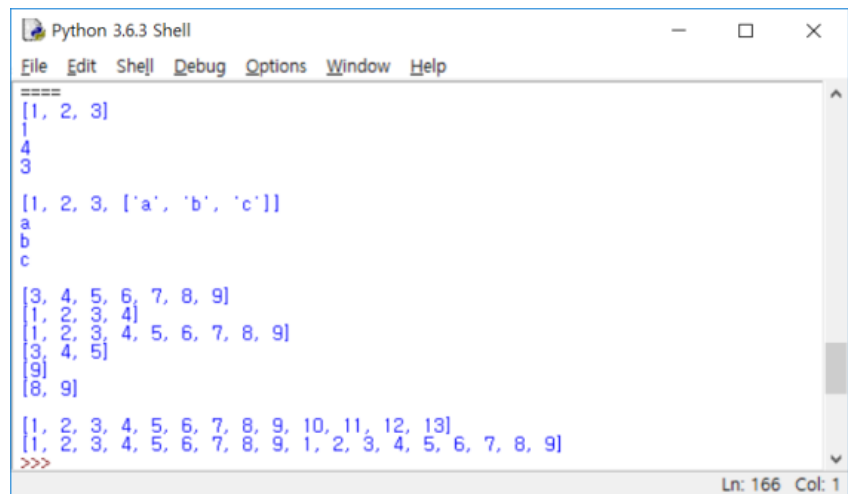
```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
print(a[2:])
print(a[:4])
print(a[:])
print(a[2:5])
print(a[-1:])
print(a[-2:], '\n')
```

#리스트 더하기

```
b = [10, 11, 12, 13]
print(a+b)
```

#리스트 반복하기

```
print(a*3)
```



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
====
[1, 2, 3]
1
4
3

[1, 2, 3, ['a', 'b', 'c']]
a
b
c

[3, 4, 5, 6, 7, 8, 9]
[1, 2, 3, 4]
[1, 2, 3, 4, 5, 6, 7, 8, 9]
[3, 4, 5]
[9]
[8, 9]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>>
```

## 딕셔너리(Dictionary)

대응관계를 나타낼 수 있는 자료형(연관 배열(Associative array), 해시(Hash))

Key를 통해 Value를 얻음

데이터를 찾을 때 순차적으로 검색하는 것이 아니라 원하는 데이터가 있는 위치를 통해 검색

### #딕셔너리 선언

```
dic = {'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
print(dic, '\n')
```

### #딕셔너리 추가

```
dic['day'] = '20171116'
print(dic)
dic[1] = [1, 2, 3]
print(dic, '\n')
#딕셔너리 삭제
del dic[1]
print(dic, '\n')
```

### #딕셔너리 key 사용해서 value 얻기

```
print(dic['name'])
print(dic['phone'])
print(dic['day'], '\n')
```

### #딕셔너리 관련 함수

#### #key 리스트 만들기(keys)

```
print(dic.keys(), '\n')
```

#### #value 리스트 만들기(values)

```
print(dic.values(), '\n')
```

#### #key, value 쌍얻기(items)

```
print(dic.items(), '\n')
```

#### #key로 value 얻기(get)

```
print(dic.get('name'))
print(dic.get('phone'))
print(dic.get('day'), '\n')
```

#### #key로 value 얻기(get)2 key가 없는 경우

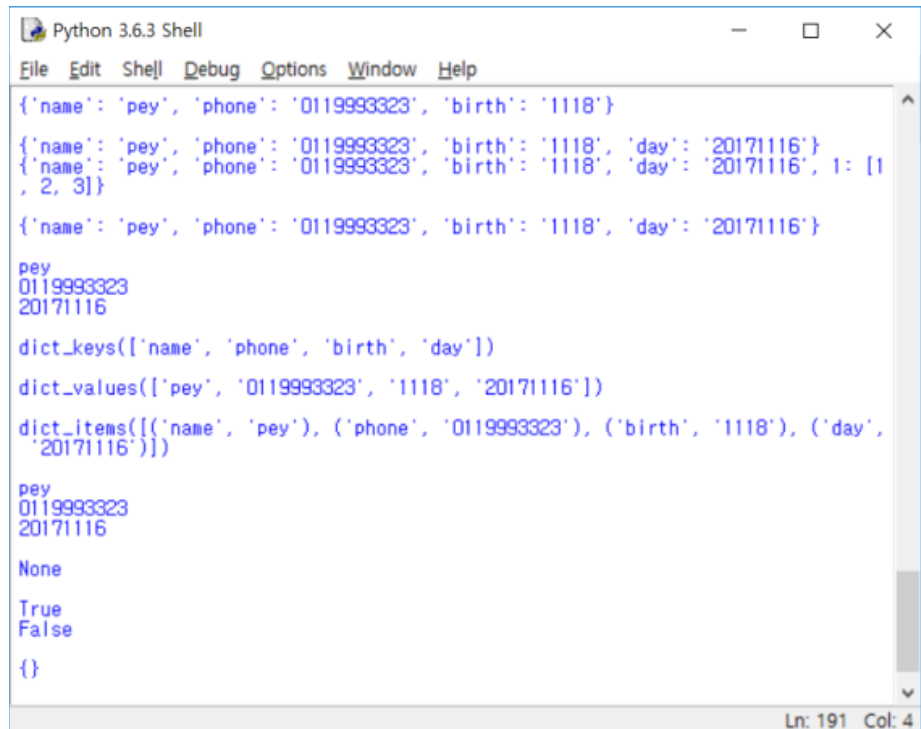
```
print(dic.get('nokey'), '\n')
```

#### #key가 딕셔너리 안에 있는지 조사(in)

```
print('name' in dic)
print('archive' in dic, '\n')
```

#### #key:value 쌍 모두 지우기(clear)

```
dic.clear()
print(dic, '\n')
```



```
Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
{'name': 'pey', 'phone': '0119993323', 'birth': '1118'}
{'name': 'pey', 'phone': '0119993323', 'birth': '1118', 'day': '20171116'}
{'name': 'pey', 'phone': '0119993323', 'birth': '1118', 'day': '20171116', 1: [1, 2, 3]}
{'name': 'pey', 'phone': '0119993323', 'birth': '1118', 'day': '20171116'}
pey
0119993323
20171116
dict_keys(['name', 'phone', 'birth', 'day'])
dict_values(['pey', '0119993323', '1118', '20171116'])
dict_items([('name', 'pey'), ('phone', '0119993323'), ('birth', '1118'), ('day', '20171116')])
pey
0119993323
20171116
None
True
False
{}
Ln: 191 Col: 4
```

## <참고자료 - 파이썬 자료형 익히기>

<https://wikidocs.net/11>

## 실습2. 백오브워드 모델 사용해 단어 빈도 추출하기

코드 : bag\_of\_words.py

### #패키지 호출

```
import numpy as np
from sklearn.feature_extraction.text import CountVectorizer
from nltk.corpus import brown
from text_chunker import chunker
```

### #브라운 말뭉치에서 데이터읽기

```
input_data = ' '.join(brown.words()[:5400])
```

### # 단어 묶음에 포함될 단어수 설정

```
chunk_size = 800
```

### #텍스트를 단어 묶음으로 나눔

```
text_chunks = chunker(input_data, chunk_size)
```

### # 사전 구조체로 변경

```
chunks = []
for count, chunk in enumerate(text_chunks):
    d = {'index': count, 'text': chunk}
    chunks.append(d)
```

### #CountVectorizer함수를 사용해 단어별 빈도를 체크, 문서 단어 행렬 생성

#CountVectorizer함수의 첫 번째 매개변수는 최소 문서 빈도며 두 번째 매개변수는 최대 문서 빈도다. 최소 문서 빈도보다 빈도가 적거나 최대 빈도수보다 높은 단어는 무시

### # 문서 단어 행렬 만들기

```
count_vectorizer = CountVectorizer(min_df=7, max_df=20)
document_term_matrix = count_vectorizer.fit_transform([chunk['text'] for chunk in chunks])
```

### # 어휘 목록 추출 후 화면에 출력

```
vocabulary = np.array(count_vectorizer.get_feature_names())
print("\nVocabulary:\n", vocabulary)
```

### # 단어 묶음별 이름 붙이기

```
chunk_names = []
for i in range(len(text_chunks)):
    chunk_names.append('Chunk-' + str(i+1))
```

### # 문서 단어 행렬 출력



```

print("\nDocument term matrix:")
formatted_text = '{:>12}' * (len(chunk_names) + 1)
print('\n', formatted_text.format('Word', *chunk_names), '\n')
for word, item in zip(vocabulary, document_term_matrix.T):
    # 'item' 은 희소 행렬 'csr_matrix'는 데이터 구조체
    output = [word] + [str(freq) for freq in item.data]
    print(formatted_text.format(*output))

```

## # 실습 포인트

1. 코드 실행 결과 매트릭스의 Chunk-1, Chunk-2.. 의미 분석하기
2. 단어 묶음에 포함될 단어수 변경하고 결과 분석하기
- 3.CountVectorizer 함수 이해하기
  - CountVectorizer함수를 사용해 단어별 빈도를 체크, 문서 단어 행렬 생성
  - CountVectorizer함수의 첫 번째 매개변수는 최소 문서 빈도며 두 번째 매개변수는 최대 문서 빈도다.
  - 최소 문서 빈도보다 빈도가 적거나 최대 빈도수보다 높은 단어는 무시된다

## 실습3 : 카테고리 예측기 만들기

코드 : category\_predictor.py

### #패키지 호출

```
from sklearn.datasets import fetch_20newsgroups
from sklearn.naive_bayes import MultinomialNB
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
```

### #카테고리 맵 정의

```
category_map = {'talk.politics.misc': 'Politics', 'rec.autos': 'Autos',
                'rec.sport.hockey': 'Hockey', 'sci.electronics': 'Electronics',
                'sci.med': 'Medicine'}
```

### #학습데이터셋을 받음

```
training_data = fetch_20newsgroups(subset='train',
                                   categories=category_map.keys(), shuffle=True, random_state=5)
```

### # countvectorizer 객체를 사용해 단어 빈도를 추출

```
count_vectorizer = CountVectorizer()
train_tc = count_vectorizer.fit_transform(training_data.data)
print("\nDimensions of training data:", train_tc.shape)
```

### # tf-idf 변환기 생성

```
tfidf = TfidfTransformer()
train_tfidf = tfidf.fit_transform(train_tc)
```

### # 테스트 데이터 정의

```
input_data = [
    'You need to be careful with cars when you are driving on slippery roads',
    'A lot of devices can be operated wirelessly',
    'Players need to be careful when they are close to goal posts',
    'Political debates help us understand the perspectives of both sides'
]
```

### # 다항 분포 나이브 베이즈 분류기 학습

```
classifier = MultinomialNB().fit(train_tfidf, training_data.target)
```

### # count vectorizer를 사용해 입력 데이터 변환

```
input_tc = count_vectorizer.transform(input_data)
```

### # tfidf 변환기를 사용해 벡터 데이터 변환

```
input_tfidf = tfidf.transform(input_tc)
```

### # 카테고리 예측

```
predictions = classifier.predict(input_tfidf)
```

### # 결과 출력

```
for sent, category in zip(input_data, predictions):  
    print("\nInput:", sent, "\nPredicted category:", \  
          category_map[training_data.target_names[category]])
```

### # 실습 포인트

1. Input data에 다양한 영어문장 입력하여 결과 분석하기
2. 카테고리 맵에서 예측할 카테고리를 다양하게 설정해 보기
  - Baseball 과 관련된 영어문장을 입력하여 결과 확인하기 (아마 Hockey 등으로 구분될 것임)
  - 카테고리 맵에 Baseball 을 추가하여 원래 문장이 Baseball 카테고리로 분류되는지 확인하기
  - Hockey/Baseball 사례와 유사한 사례 찾아보기

## 텍스트 분석 및 NLP 정리

### 토큰화

- 입력된 텍스트를 여러 종류의 토큰으로 분리 (tokenizer)

### 어간 추출 및 표제화

- 입력된 텍스트를 기본형으로 변형 (lemmatizer)

### 텍스트 단어 묶음

- 입력된 텍스트를 사전에 정의된 조건에 따라 단어 묶음으로 나누기 (text\_chunker)

### 모델링 : Bag of Word, 베이지안 트리 등

- 입력된 텍스트를 위한 문서 단어 행렬 만들기

### 분류 : 머신러닝 기법

classifier