

[TA 2-1]

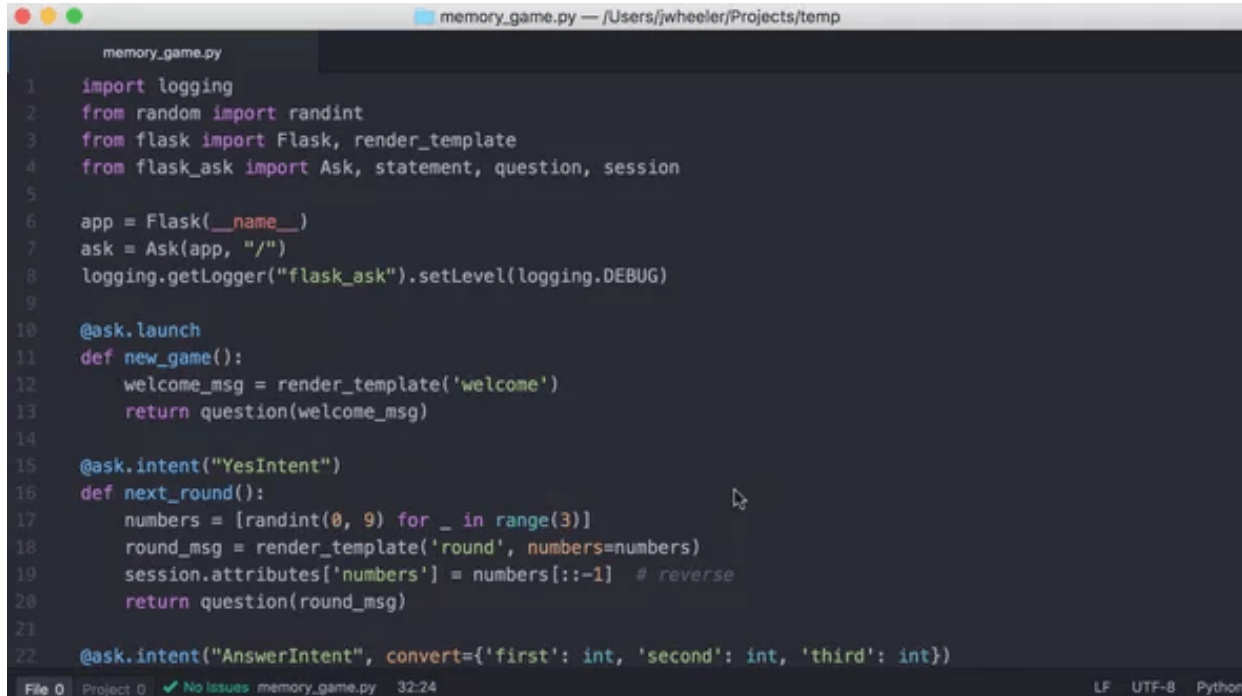
# Python Primer - 2

빅데이터 프로그래밍 기초

박진수 교수

Big Data Institute,  
Seoul National University

# 기초 데이터 타입(Basic Data Types)

A screenshot of a code editor window titled 'memory\_game.py — /Users/jwheeler/Projects/temp'. The editor shows the following Python code:

```
memory_game.py
1  import logging
2  from random import randint
3  from flask import Flask, render_template
4  from flask_ask import Ask, statement, question, session
5
6  app = Flask(__name__)
7  ask = Ask(app, "/")
8  logging.getLogger("flask_ask").setLevel(logging.DEBUG)
9
10 @ask.launch
11 def new_game():
12     welcome_msg = render_template('welcome')
13     return question(welcome_msg)
14
15 @ask.intent("YesIntent")
16 def next_round():
17     numbers = [randint(0, 9) for _ in range(3)]
18     round_msg = render_template('round', numbers=numbers)
19     session.attributes['numbers'] = numbers[::-1] # reverse
20     return question(round_msg)
21
22 @ask.intent("AnswerIntent", convert={'first': int, 'second': int, 'third': int})
```

The status bar at the bottom indicates 'File 0', 'Project 0', 'No Issues', 'memory\_game.py', '32:24', 'LF', 'UTF-8', and 'Python'.

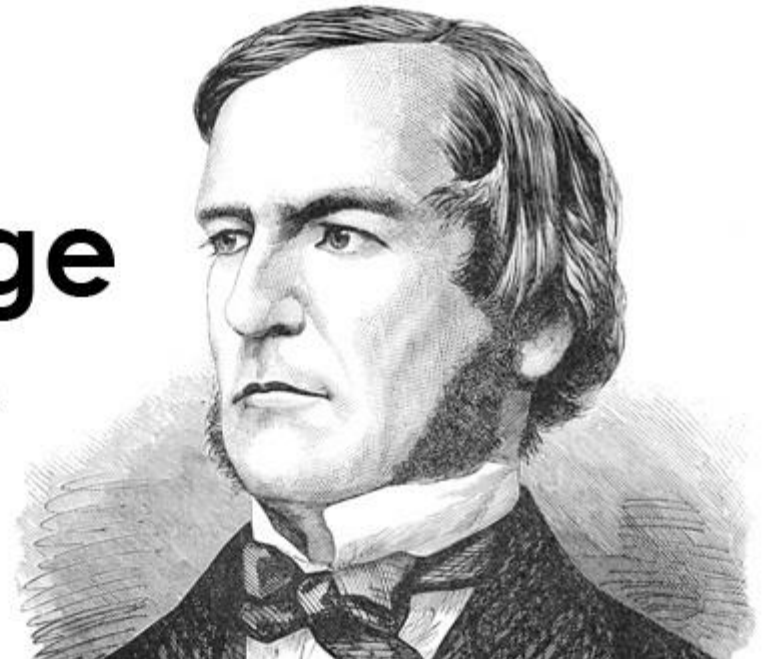
# 기초 데이터 타입

- 불린(Boolean) 타입

- '참(True)' 혹은 '거짓(False)'를 나타내는 타입

- 논리 연산자(logical operator)를 활용한 논리 연산이나, 비교 연산자(comparison operator)를 활용해 수치나 텍스트 등을 서로 비교할 때 흔히 사용됨

**George  
Boole**



# 기초 데이터 타입

- 불린(Boolean) 타입

- 논리연산

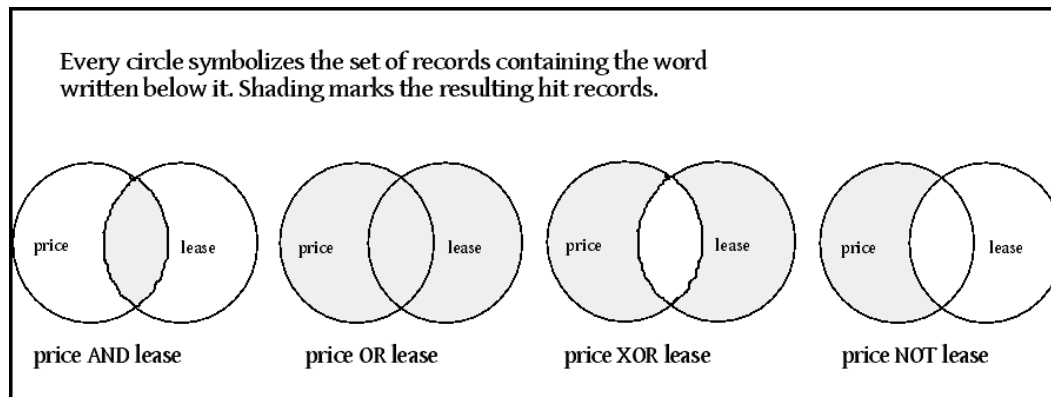
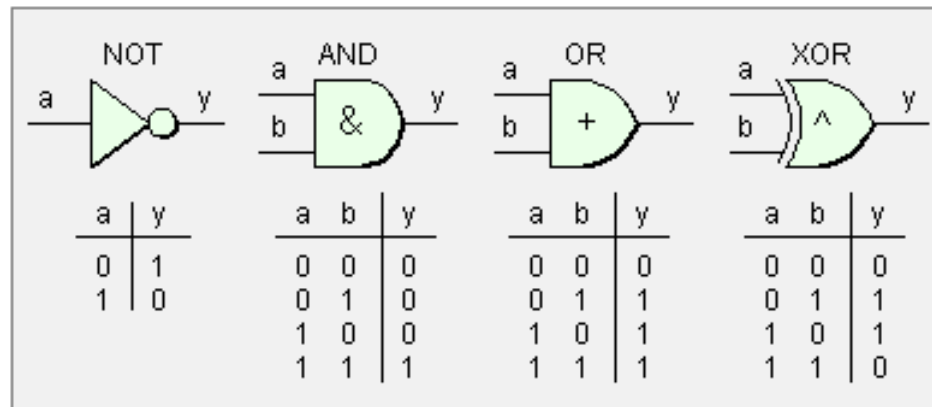
```
> True and False  
>  
> True or False  
>  
> not True  
>  
> not False  
>
```

```
> 1 and 0  
>  
> 1 or 0  
>  
> not 1  
>  
> not 0  
>
```

# 기초 데이터 타입

## · 불린(Boolean) 타입

- XOR 연산: input a와 b 중 하나만 True(1)일 때에만 True를 리턴

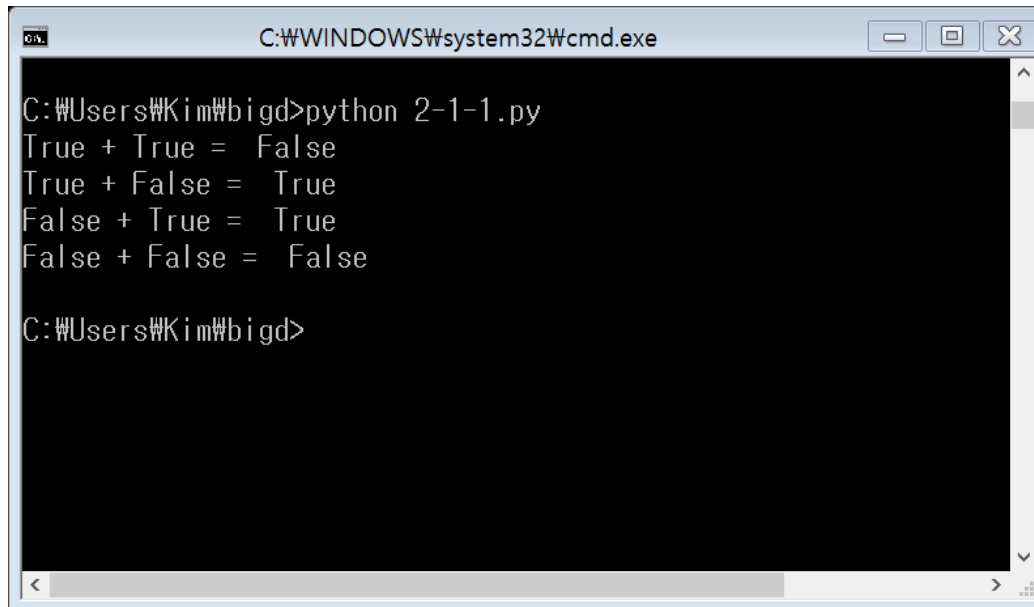


# 기초 데이터 타입

## · 실습 2-1-1. XOR 연산 구현하기

- XOR 연산을 구현한 xor()함수를 만들고 그 결과를 출력해 보자

- x, y 두 개의 파라미터를 인풋으로 받아들인다
- XOR 연산을 수행한 결과를 리턴한다. 결과를 출력해 본다
- 수행 예시



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Kim\bigd>python 2-1-1.py
True + True = False
True + False = True
False + True = True
False + False = False

C:\Users\Kim\bigd>
```

# 기초 데이터 타입

- 불린(Boolean) 타입

- 비트 연산자(^)를 통해 XOR 연산을 쉽게 구현할 수 있다

```
> True ^ True
>
> True ^ False
>
> 1 ^ 1
>
> 0 ^ 0
>
```

# 기초 데이터 타입

- 불린(Boolean) 타입

- 비교연산

```
> 1 > 2
>
> 100 < 1000
>
> 'a' == 'b'
>
> 'j' != 'i'
>
```



# 기초 데이터 타입

- 불린(Boolean) 타입

- in, not in 연산자

```
> 1 in ['1', '2', '3']  
>  
> 2 in [1, 2, 3]  
>  
> 'a' in ['a', 'b', 'c']  
>  
> a in ['a', 'b', 'c']  
>
```

# 기초 데이터 타입

## · 실습 2-1-2. 리스트(튜플) 내의 원소 찾기(search)

- 리스트(튜플) 내에 특정 원소가 들어 있으면 그 인덱스를 반환하고, 들어 있지 않으면 False를 반환하는 함수 `search_element()` 를 구현해 본다

■ 두 개의 파라미터(리스트와 변수)를 인풋으로 받아들인다

■ 수행 예시

```
In [6]: l = [1,2,3]
        e1 = 2
        e2 = 10
```

```
In [9]: search_element(l, e1)
```

```
Out[9]: 1
```

```
In [10]: search_element(l, e2)
```

```
Out[10]: False
```

# 기초 데이터 타입

- `bool()` 함수: 특정 데이터가 `True`인지 `False`인지 검증하려면 `bool()` 함수로 결과를 출력해 본다

```
In [13]: bool(0)
```

```
Out[13]: False
```

```
In [14]: bool(1)
```

```
Out[14]: True
```

```
> bool('')  
>  
> bool([])  
>  
> bool(())  
>  
> bool(100)  
>
```

# 기초 데이터 타입

- 정수(integer)/실수(float) 타입

- 일반적으로 수치를 나타내고 수치 연산을 수행하기 위해 가장 흔히 사용되는 데이터 타입
- 정수형은 소수점 이하를 표현할 수 없음
- 수치 연산

```
> 1 + 2  
>  
> 1 / 2  
>  
> 1 // 2  
>  
> 10 ** 2  
>
```

# 기초 데이터 타입

## · 실습 2-1-3. 직사각형 넓이 구하기

- 직사각형의 가로와 세로의 길이를 정수형 값으로 입력 받은 후 가로의 길이는 5 증가시키고 세로의 길이는 2배 하여 저장한 후 가로의 길이, 세로의 길이, 그리고 원래 넓이에서 확장한 후의 넓이를 나눈 값을 차례로 출력하는 함수 `area_expand()`를 만든다

- 가로 길이 = 5, 세로 길이 = 10을 넣고 결과를 테스트해 본다

- 수행 예시

```
In [16]: area_expand(5, 10)
```

```
Width = 10  
Length = 20  
Area Ratio = 0.25
```

# 기초 데이터 타입

## · 실습 2-1-4. 합과 평균 계산하기

- 정수 5개를 입력 받아서 그 합과 평균을 출력하되, 0 이 입력되면 0 전까지 까지 입력된 합과 평균을 튜플로 출력하는 함수를 calculator()를 만든다
- 평균은 소수부분은 버리고 정수만 출력한다.(0이 입력된 경우 0을 제외한 합과 평균을 구한다.)
- if-else 문을 활용한다
- (1,2,3,4,5)와 (1,2,0,4,5)를 입력으로 넣고 결과를 출력해 본다
- 수행 예시

```
In [19]: print(calculator(1,2,3,4,5))  
(15, 3)
```

```
In [20]: print(calculator(1,2,0,4,5))  
(3, 1)
```

# 기초 데이터 타입

- 문자열(String)

- 문자(text) 형태의 데이터를 담기 위한 자료형
- 작은 따옴표(') 혹은 큰 따옴표(")로 둘러 싸면 자동으로 문자열으로 변환
- 문자열 이스케이프(\)

```
> 'my car'
>
> 'jane\'s car'
>
> "jane's car"
>
> 'my car \n Paul\'s car'
>
```

# 기초 데이터 타입

- 문자열(String)

- 문자열 연산

```
> 'my' + 'car'  
>  
> 'jane'*2  
>
```

- 문자열 인덱싱

```
> 'my car'[1]  
>  
> 'abc'[1:]  
>
```



# 기초 데이터 타입

- 문자열(String)

- 문자열 포매팅

```
> 'I ate {} apples'.format(3)
>
> 'Her name is {}'.format(Jane)
>
```

# 기초 데이터 타입

## · 실습 2-1-5. 문자열 검색하기

- 단어(혹은 문장)와 문자 하나를 입력 받아 단어에서 입력 받은 문자와 같은 문자를 찾아서 그 위치(인덱스)를 반환하는 함수 `search_text()`를 작성한다

■ 두 개의 파라미터(단어와 문자)를 인풋으로 받아들인다

■ 문자가 없을 경우 `False`를 반환한다

■ 수행 예시

```
In [13]: def search_text(word, char):  
         if char in word:  
             return word.index(char)  
         return False
```

```
In [16]: print(search_text('name', 'a'))  
         print(search_text('name', 'e'))  
         print(search_text('jane', 'p'))  
         print(search_text('jane', 'n'))
```

```
1  
3  
False  
2
```

# 기초 데이터 타입

## · 실습 2-1-6. 문자열 뒤집기

- 문자열 5개를 입력 받아 만약 문자열의 길이가 홀수면 거꾸로 뒤집어 출력하는 함수 `reverse_text()`를 작성한다

■ 문자열의 길이가 짝수이면 그대로 출력한다

■ 결과로 거꾸로 뒤집은 횟수를 리턴한다

■ 수행 예시

```
In [20]: print(reverse_text('tiger', 'lion', 'bear', 'snake', 'leopard'))  
regit  
lion  
bear  
ekans  
drapoe  
3
```

# 기초 데이터 타입

## · 실습 2-1-7. 모음 개수 계산하기

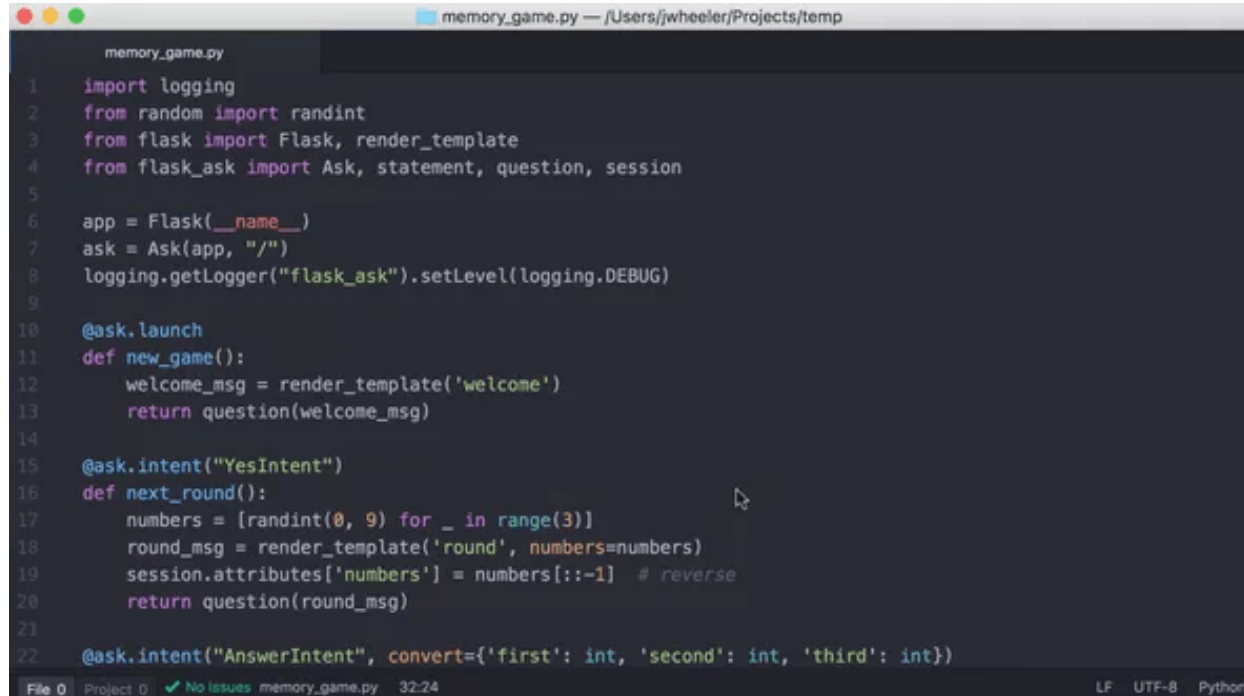
- 영어 단어/문장 문자열을 입력 받아 문자열 내에 속한 모음 알파벳(a,e,i,o,u)의 개수를 출력하는 vowel() 함수를 작성한다

■ 입력 문자열을 문자열로 이루어진 리스트로 변환한다

■ 수행 예시

```
In [30]: print(vowel('apples'))  
         print(vowel('Her name is Jane'))  
  
2  
6
```

# 배열 데이터 타입(Collections Data Types)



The image shows a code editor window titled "memory\_game.py — /Users/jwheeler/Projects/temp". The code is written in Python and uses the Flask framework. It includes imports for logging, random.randint, Flask, render\_template, Ask, statement, question, and session. The code sets up a Flask app, an Ask object, and logging. It defines two main functions: new\_game() and next\_round(). new\_game() renders a welcome message and returns a question. next\_round() generates three random numbers, renders a round message, and returns a question. The code also includes decorators for @ask.launch and @ask.intent to handle user input.

```
memory_game.py
1  import logging
2  from random import randint
3  from flask import Flask, render_template
4  from flask_ask import Ask, statement, question, session
5
6  app = Flask(__name__)
7  ask = Ask(app, "/")
8  logging.getLogger("flask_ask").setLevel(logging.DEBUG)
9
10 @ask.launch
11 def new_game():
12     welcome_msg = render_template('welcome')
13     return question(welcome_msg)
14
15 @ask.intent("YesIntent")
16 def next_round():
17     numbers = [randint(0, 9) for _ in range(3)]
18     round_msg = render_template('round', numbers=numbers)
19     session.attributes['numbers'] = numbers[::-1] # reverse
20     return question(round_msg)
21
22 @ask.intent("AnswerIntent", convert={'first': int, 'second': int, 'third': int})
```

File 0 Project 0 ✓ No Issues memory\_game.py 32:24 LF UTF-8 Python

# 배열 데이터 타입

- 리스트(list)

- 여러 요소를 담기 위한 자료형

```
> a = [1, 2, 3, 4, 5]
>
> b = ['Her', 'name', 'is', 'Jane']
>
> c = [1, 2, 3, [4,5]]
>
> d = [1, 2, 'Jane']
>
```

# 배열 데이터 타입

- 리스트(list)

- 리스트 관련 함수

```
> a.append(5)
>
> a.sort()
>
> a.reverse()
>
> a.index(1)
>
```

# 배열 데이터 타입

- 튜플(tuple)

- 튜플은 리스트와 거의 유사하지만 몇 가지 차이점을 갖는다

- 리스트는 대괄호 '[' 로 둘러싸지만, 튜플은 소괄호 '(' 로 둘러싼다

- 리스트는 값의 생성, 삭제, 수정이 가능하지만 튜플은 값을 바꿀 수 없다

```
> a = 1, 2, 3, 4, 5
>
> b = ('Her', 'name', 'is', 'Jane')
>
> c = (1, 2, 3, (4,5))
>
> d = (1, 2, 'Jane')
>
```



# 배열 데이터 타입

## · 실습 2-1-8. 점수 정렬하기

- 5명의 점수(0점 이상 100점 이하)를 받아 가장 낮은 점수부터 가장 높은 점수까지 차례대로 출력하는 함수 `sort_score()`를 작성한다

■ 점수를 받아 리스트로 만든다

■ 수행 예시

```
In [33]: sort_score(5,2,3,1,4)
```

```
1  
2  
3  
4  
5
```

# 배열 데이터 타입

## · 실습 2-1-9. 2차원 리스트 변경하기

- 알파벳 대문자(A~O)로 이루어진 3행 5열의 2차원 리스트를 생성하고 각 행을 소문자로 바꾸어 출력해 본다

■ (3, 2) 크기의 2차원 리스트를 정의한다. 행 별로 한 줄에 출력되도록 한다

■ 수행 예시

```
a b c d e
```

```
f g h i j
```

```
k l m n o
```

# 배열 데이터 타입

- 딕셔너리(dictionary)

- ‘대응 관계’를 나타내는 자료형: 연관 배열(Associative array) 또는 해시(Hash)

- ‘이름’ = ‘홍길동’, ‘생일’ = ‘1993년 9월 28일’

- 파이썬에서는 이를 ‘딕셔너리’라고 부른다

- 요소값을 순차적으로 구하지 않고 Key를 통해 Value에 접근한다

- 사전에서 단어와 뜻을 매칭시키듯이 Key와 Value가 1대1로 매핑된다

```
{Key1: Value1, Key2: Value2, ...}
```

# 배열 데이터 타입

- 딕셔너리(dictionary)

- 딕셔너리 생성하기

```
> dic1 = {1: 'a' }
```

- 쌍(pair) 추가하기

```
> dic1[2] = 'b'
```

- 값 접근하기

```
> dic1[2]
```

# 배열 데이터 타입

- 셋(set)

- 셋도 리스트와 유사하지만 몇 가지 차이점을 갖는다

- 셋은 중복을 허용하지 않는다

- 셋의 자료들은 순서가 없다

```
> S = set([1,2,3])
```

# 배열 데이터 타입

- 셋(set)

- 셋을 통한 집합연산

```
> s1 = set([1,2,3])
> s2 = set([3,4,5])
> s1 & s2
>
> s1 | s2
>
> s1 - s2
>
```

# 배열 데이터 타입

- 실습 2-1-10. 딕셔너리의 요소값을 통해 키 필터링하기

- 아래 직원들 중 연봉이 50000이상인 직원들의 이름을 딕셔너리를 활용해 출력해 본다

- 연봉정보

- John: 30000

- Jane: 50000

- Paul: 45000

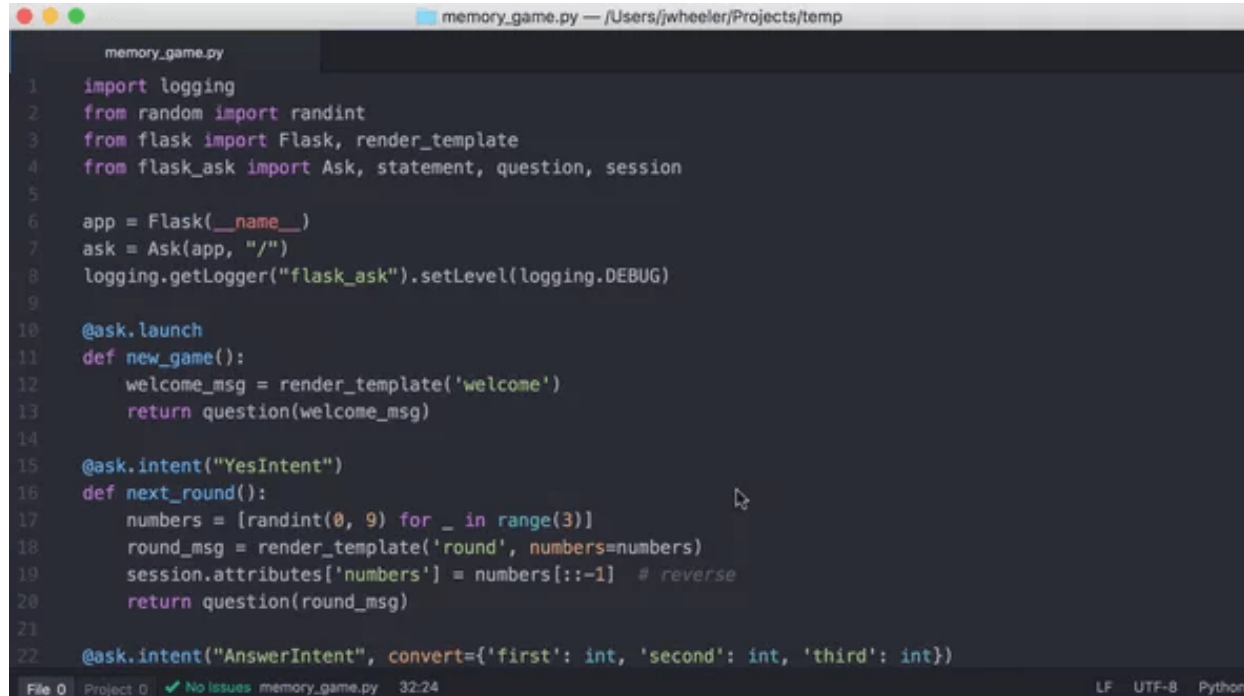
- Elizabeth: 70000

- Seth: 10000

- 수행 예시

```
Jane's salary is: 50000  
Elizabeth's salary is: 70000
```

# 제어문(Control Structures)



The screenshot shows a code editor window titled "memory\_game.py — /Users/jwheeler/Projects/temp". The code is written in Python and uses the Flask framework. It includes imports for logging, random.randint, Flask, render\_template, Ask, statement, question, and session. The code defines a Flask app, an Ask object, and sets the logging level to DEBUG. It then defines two functions: new\_game() and next\_round(). new\_game() renders a 'welcome' template and returns a question. next\_round() generates three random numbers, renders a 'round' template, and returns a question. The code also includes decorators for @ask.launch and @ask.intent to handle user input.

```
memory_game.py
1  import logging
2  from random import randint
3  from flask import Flask, render_template
4  from flask_ask import Ask, statement, question, session
5
6  app = Flask(__name__)
7  ask = Ask(app, "/")
8  logging.getLogger("flask_ask").setLevel(logging.DEBUG)
9
10 @ask.launch
11 def new_game():
12     welcome_msg = render_template('welcome')
13     return question(welcome_msg)
14
15 @ask.intent("YesIntent")
16 def next_round():
17     numbers = [randint(0, 9) for _ in range(3)]
18     round_msg = render_template('round', numbers=numbers)
19     session.attributes['numbers'] = numbers[::-1] # reverse
20     return question(round_msg)
21
22 @ask.intent("AnswerIntent", convert={'first': int, 'second': int, 'third': int})
```

File 0 Project 0 ✓ No Issues memory\_game.py 32:24 LF UTF-8 Python



# 제어문(Control Structures)

- 조건문의 기본 형태

- 조건문은 기본적으로 아래와 같은 형태를 가진다

```
if [불린 타입]:  
    수행할 내용  
elif [불린 타입]:  
    수행할 내용  
...  
elif [불린 타입]:  
    수행할 내용  
else:  
    수행할 내용
```

■ elif와 else는 선택사항임

# 제어문(Control Structures)

- 조건문의 기본 형태

- 조건문은 기본적으로 아래와 같은 형태를 가진다

```
x = 100
y = 10
if x > y:
    print('x is bigger than y')
elif x < y:
    print('y is bigger than y')
else:
    print('x equals y')
```

# 제어문(Control Structures)

- while문

- while문은 기본적으로 아래와 같은 형태를 가진다

```
while [불린 타입]:  
    수행할 내용  
else:  
    수행할 내용
```

■ if문과 같이, while문에서도 else는 선택사항이다

```
x = 0  
while x < 10:  
    print(x)  
    x += 1
```

# 제어문(Control Structures)

- for문

- for문은 기본적으로 아래와 같은 형태를 가진다

```
for 요소 in 반복형:  
    수행할 내용  
else:  
    수행할 내용
```

■ if문과 같이, for문에서도 else는 선택사항이다

```
for i in [1,2,3,4,5]:  
    print(i)
```

# 제어문(Control Structures)

## · 실습 2-1-11. 구구단 - 1

- for문을 활용해 1단부터 9단까지의 구구단을 출력해 본다

■ 한 줄에 한 단 씩 출력한다

■ 수행 예시

```
1 * 1 = 1  1 * 2 = 2  1 * 3 = 3  1 * 4 = 4  1 * 5 = 5  1 * 6 = 6  1 * 7 = 7  1 * 8 = 8  1 * 9 = 9
2 * 1 = 2  2 * 2 = 4  2 * 3 = 6  2 * 4 = 8  2 * 5 = 10  2 * 6 = 12  2 * 7 = 14  2 * 8 = 16  2 * 9 = 18
3 * 1 = 3  3 * 2 = 6  3 * 3 = 9  3 * 4 = 12  3 * 5 = 15  3 * 6 = 18  3 * 7 = 21  3 * 8 = 24  3 * 9 = 27
4 * 1 = 4  4 * 2 = 8  4 * 3 = 12  4 * 4 = 16  4 * 5 = 20  4 * 6 = 24  4 * 7 = 28  4 * 8 = 32  4 * 9 = 36
5 * 1 = 5  5 * 2 = 10  5 * 3 = 15  5 * 4 = 20  5 * 5 = 25  5 * 6 = 30  5 * 7 = 35  5 * 8 = 40  5 * 9 = 45
6 * 1 = 6  6 * 2 = 12  6 * 3 = 18  6 * 4 = 24  6 * 5 = 30  6 * 6 = 36  6 * 7 = 42  6 * 8 = 48  6 * 9 = 54
7 * 1 = 7  7 * 2 = 14  7 * 3 = 21  7 * 4 = 28  7 * 5 = 35  7 * 6 = 42  7 * 7 = 49  7 * 8 = 56  7 * 9 = 63
8 * 1 = 8  8 * 2 = 16  8 * 3 = 24  8 * 4 = 32  8 * 5 = 40  8 * 6 = 48  8 * 7 = 56  8 * 8 = 64  8 * 9 = 72
9 * 1 = 9  9 * 2 = 18  9 * 3 = 27  9 * 4 = 36  9 * 5 = 45  9 * 6 = 54  9 * 7 = 63  9 * 8 = 72  9 * 9 = 81
```

# 제어문(Control Structures)

## · 실습 2-1-12. 구구단 - 2

- while문을 활용해 1단부터 9단까지의 구구단을 출력해 본다

■ 한 줄에 한 단 씩 출력한다

■ 수행 예시

```
1 * 1 = 1  1 * 2 = 2  1 * 3 = 3  1 * 4 = 4  1 * 5 = 5  1 * 6 = 6  1 * 7 = 7  1 * 8 = 8  1 * 9 = 9
2 * 1 = 2  2 * 2 = 4  2 * 3 = 6  2 * 4 = 8  2 * 5 = 10  2 * 6 = 12  2 * 7 = 14  2 * 8 = 16  2 * 9 = 18
3 * 1 = 3  3 * 2 = 6  3 * 3 = 9  3 * 4 = 12  3 * 5 = 15  3 * 6 = 18  3 * 7 = 21  3 * 8 = 24  3 * 9 = 27
4 * 1 = 4  4 * 2 = 8  4 * 3 = 12  4 * 4 = 16  4 * 5 = 20  4 * 6 = 24  4 * 7 = 28  4 * 8 = 32  4 * 9 = 36
5 * 1 = 5  5 * 2 = 10  5 * 3 = 15  5 * 4 = 20  5 * 5 = 25  5 * 6 = 30  5 * 7 = 35  5 * 8 = 40  5 * 9 = 45
6 * 1 = 6  6 * 2 = 12  6 * 3 = 18  6 * 4 = 24  6 * 5 = 30  6 * 6 = 36  6 * 7 = 42  6 * 8 = 48  6 * 9 = 54
7 * 1 = 7  7 * 2 = 14  7 * 3 = 21  7 * 4 = 28  7 * 5 = 35  7 * 6 = 42  7 * 7 = 49  7 * 8 = 56  7 * 9 = 63
8 * 1 = 8  8 * 2 = 16  8 * 3 = 24  8 * 4 = 32  8 * 5 = 40  8 * 6 = 48  8 * 7 = 56  8 * 8 = 64  8 * 9 = 72
9 * 1 = 9  9 * 2 = 18  9 * 3 = 27  9 * 4 = 36  9 * 5 = 45  9 * 6 = 54  9 * 7 = 63  9 * 8 = 72  9 * 9 = 81
```

# 제어문(Control Structures)

- 실습 2-1-13. 피보나치(Fibonacci) 수열

- 0부터 시작하는 피보나치 수열의 열 번째 시퀀스 까지의 원소를 출력한다

$$F_n = \begin{cases} 0 & \text{if } n = 0; \\ 1 & \text{if } n = 1; \\ F_{n-1} + F_{n-2} & \text{if } n > 1. \end{cases}$$

■ while 문을 활용한다

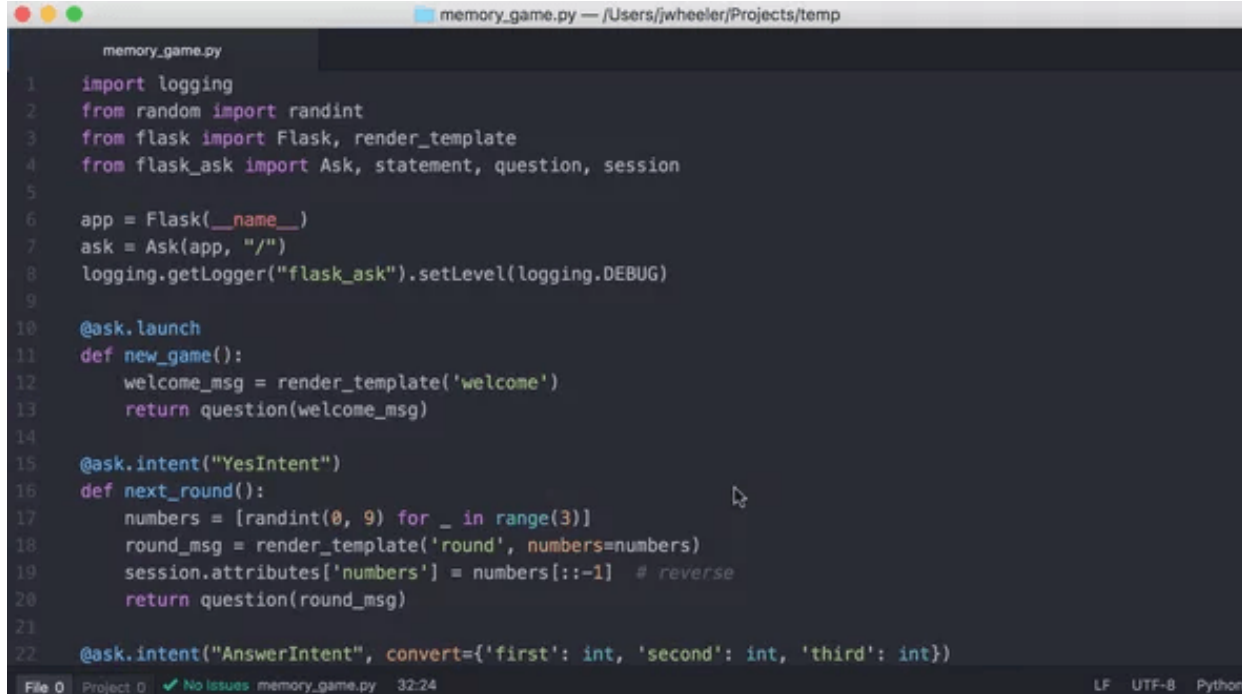
■ 재귀식(recursion)을 사용할 수도 있다!

■ 수행 예시

```
In [23]: fibonacci(10)
```

```
0
1
1
2
3
5
8
13
21
34
```

# 파일 읽고 쓰기(File Handling)

A screenshot of a code editor window titled "memory\_game.py — /Users/jwheeler/Projects/temp". The editor shows the following Python code:

```
memory_game.py
1  import logging
2  from random import randint
3  from flask import Flask, render_template
4  from flask_ask import Ask, statement, question, session
5
6  app = Flask(__name__)
7  ask = Ask(app, "/")
8  logging.getLogger("flask_ask").setLevel(logging.DEBUG)
9
10 @ask.launch
11 def new_game():
12     welcome_msg = render_template('welcome')
13     return question(welcome_msg)
14
15 @ask.intent("YesIntent")
16 def next_round():
17     numbers = [randint(0, 9) for _ in range(3)]
18     round_msg = render_template('round', numbers=numbers)
19     session.attributes['numbers'] = numbers[::-1] # reverse
20     return question(round_msg)
21
22 @ask.intent("AnswerIntent", convert={'first': int, 'second': int, 'third': int})
```

The editor interface includes a status bar at the bottom showing "File 0", "Project 0", "No Issues", "memory\_game.py", "32:24", "LF", "UTF-8", and "Python".



# 파일 읽고 쓰기(File Handling)

## · 파일 생성하기

```
> f = open('[파일 이름].txt', 'w')  
> f.close()
```

- 'r': 읽기모드(파일을 읽기만 할 때 사용)
- 'w': 쓰기모드(파일에 내용을 쓸 때 사용)
- 'a': 추가모드(파일의 마지막에 새로운 내용을 추가시킬 때 사용)

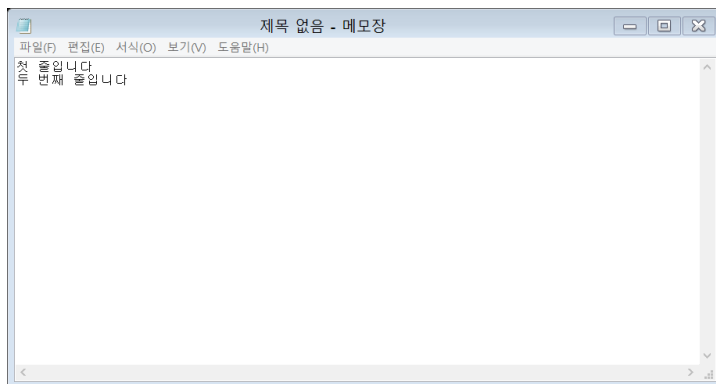
# 파일 읽고 쓰기(File Handling)

## · 파일 생성하기

### - 파일에 출력값 적기

```
> f = open('[파일 이름].txt', 'w')  
> f.write('첫 줄입니다')  
> f.write('두 번째 줄입니다')  
> f.close()
```

## ■ 결과



# 파일 읽고 쓰기(File Handling)

## · 파일 읽기

```
> f = open('[파일 이름].txt', 'r')  
> f.close()
```

## - 파일 내용 읽어오기

```
> f.readline()  
> f.readlines()  
> f.read()
```

■ 세 함수의 차이는?

# 파일 읽고 쓰기(File Handling)

## · 실습 2-1-14. csv 형식의 데이터 구조화하기

- enrollments.csv 파일을 읽어 각 열의 데이터가 하나의 딕셔너리가 되어 전체 데이터를 딕셔너리로 이루어진 리스트로 구조화한다

```
data = [{row1 data}, {row2 data}, ... {rowN data}]
```

■ 첫 번째 열의 헤더(header)는 제외한다

- 수행 예시

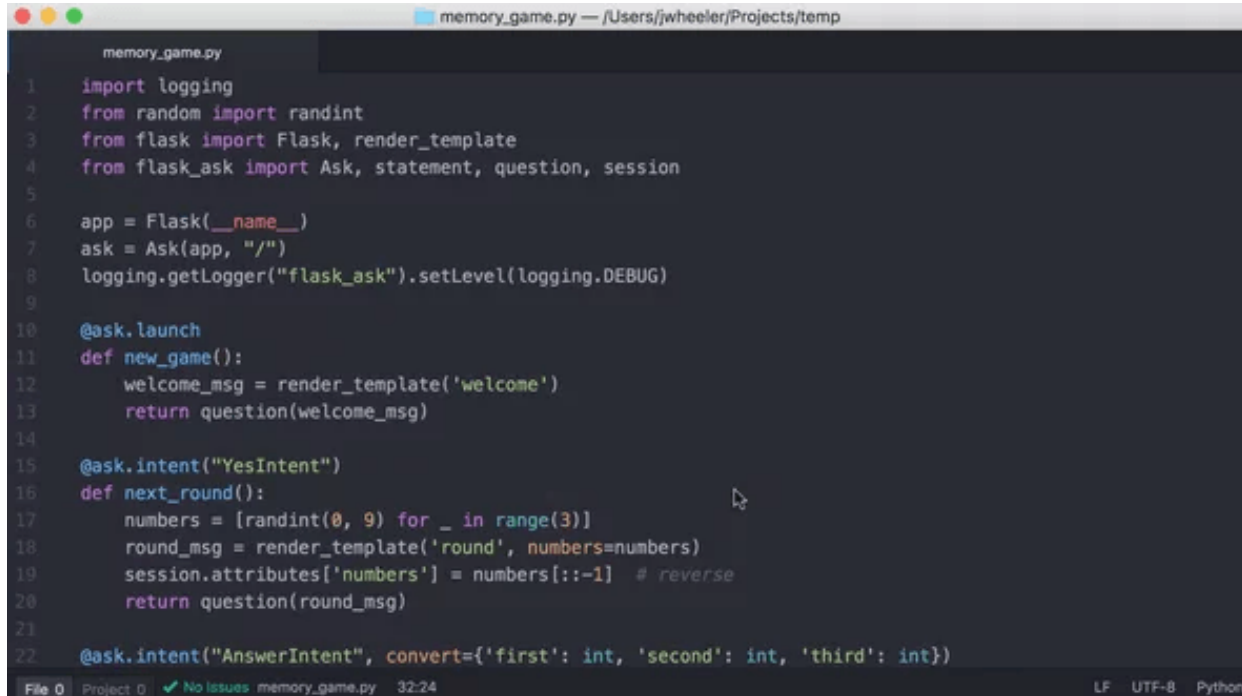
```
In [6]: data[0]
```

```
Out[6]: {'account_key': '448',  
        'cancel_data': '2015-01-14',  
        'days_to_cancel': '65',  
        'is_canceled': 'True',  
        'is_udacity': 'True',  
        'join_data': '2014-11-10',  
        'status': 'canceled'}
```

```
In [7]: data[1]
```

```
Out[7]: {'account_key': '448',  
        'cancel_data': '2014-11-10',  
        'days_to_cancel': '5',  
        'is_canceled': 'True',  
        'is_udacity': 'True',  
        'join_data': '2014-11-05',  
        'status': 'canceled'}
```

# 모듈과 패키지(Modules and Packages)

A screenshot of a code editor window titled "memory\_game.py — /Users/jwheeler/Projects/temp". The editor shows the following Python code:

```
memory_game.py
1  import logging
2  from random import randint
3  from flask import Flask, render_template
4  from flask_ask import Ask, statement, question, session
5
6  app = Flask(__name__)
7  ask = Ask(app, "/")
8  logging.getLogger("flask_ask").setLevel(logging.DEBUG)
9
10 @ask.launch
11 def new_game():
12     welcome_msg = render_template('welcome')
13     return question(welcome_msg)
14
15 @ask.intent("YesIntent")
16 def next_round():
17     numbers = [randint(0, 9) for _ in range(3)]
18     round_msg = render_template('round', numbers=numbers)
19     session.attributes['numbers'] = numbers[::-1] # reverse
20     return question(round_msg)
21
22 @ask.intent("AnswerIntent", convert={'first': int, 'second': int, 'third': int})
```

The status bar at the bottom indicates "File 0", "Project 0", "No Issues", "memory\_game.py", "32:24", "LF", "UTF-8", and "Python".

# 모듈과 패키지(Modules and Packages)

## · 모듈 불러오기

```
> import [모듈 이름]
> import [모듈 이름], [모듈 이름], ... , [모듈 이름]
> import [모듈 이름] as 별칭
```

```
> from [모듈 이름] import [객체 이름]
> from [모듈 이름] import [객체 이름], [객체 이름], ... , [객체 이름]
> from [모듈 이름] import [객체 이름] as [별칭]
```

# 모듈과 패키지(Modules and Packages)

· (데이터 분석을 위해) 자주 사용되는 파이썬 패키지/모듈

패키지명	용도	별칭	불러오기
beautifulsoup4(bs4)	웹 데이터 추출(HTML/XML)	-	from bs4 import ~
gensim	텍스트 모델링(토픽모델링, Word2Vec 등)	-	
matplotlib	데이터 시각화(plotting)	-	import matplotlib.pyplot as plt
networkx	네트워크 그래프 작성	nx	import networkx as nx
nltk	텍스트 데이터 전처리	-	import nltk
numpy	수치형 자료 프로세싱	np	import numpy as np
pandas	데이터 구조화	pd	import pandas as pd
pymysql	데이터베이스 연결(mysql)		import pymysql
scikit-learn	기계학습 알고리즘		from sklearn import ~