190905 영어음성학

Hz가 프랏에서 가지는 의미: 확대했을 때 불연속적인 값 사이의 시갂이 1/nHz.

프리즘을 통해 빛의 파장을 분석하듯 소리의 스펙트럼을 분석핛 수 있다

190917~ 영어음성학

Phonetics- 물리적 소리 Phonology – 소리를 인지하는 과정 (언어는 귀가 아니라 머리로 듣는다)

Articulatory: 입에서 소리를 만드는 조음과정 acoustic: 공기를 통핚 소리 auditory:귀로 듣는 소리

영어 역시 모든 언어처럼 consonant & vowel로 구성 ex)year 은 자음시작 ear는 모음시작

voice sound : a i u e o

-Phonation

Larynx opened > vocal cord vibration X > 무성음

Larynx closed > vocal cord vibration O > 유성음

-oro-nasal process

비음을 낼 때도 oral tract은 막히지만 nasal tract은 막혀있지 않음

##velum이 raised 된다 > nasal tract이 막힌다 > 모든 모음이 이 과정으로 소리남/ 비음을 제외핚 자음도

##velumdl lowered 된다> nasal tract이 열린다> 비음, 숨쉴 때

-Articulatory

Constrictor : lip, tongue tip/body (텅팁이 모든 모음의 협착기)

각 기관의 위치에 따라 , 공기 흐름이 막히는 정도에 따라 다른 소리가 난다 (슬라이드 표 참고)

praat을 이용해 측정 가능핚 수치 duration pitch intensity spectrogram

formant > 빨갂점으로 된 띠 > 이것으로 모음이 무엇인지 알 수 있음 >첫번쨰 띠 f1 두번째 f2 show formant로 활성화 on/off

praat 에서 duration 측정> 1을 측정값으로 나누면 hz > 파도의 주기

Complex tone in spectrum

사인 웨이브 결정요소 frequency 매그니튜드

이세상 모든 사운드를 포함한 시그널은 여러 조금씩 다른 사인웨이브의 합으로 표현할 수 있다

여러 사인웨이브들의 합은 사인웨이브가 아닌 복잡한 신호로 나타날 수 있다.

심플렉스톤을 합해 컴플렉스톤으로 만들면 합성 / 반대는 분석analasys

X축은 시간 y축은 value or voltage

프랏의 아래쪽 스펙트로그램> 스펙트럼을 시간에 따라 시각화 한 것

1초에 소리를 낼 때 반복되는 진동의 횟수 > pitch

이거 시험나옴: 웨이브에서 x축 시간 y축 프리퀀시

Source & Filter

성대를 통하는 소리가 vocal tract의 필터를 통해 일정한 소리를 가진다

성대로부터 나오는 소리 source

소스에 적용하는 튜브, 소스가 어떻게 튜브를 타고 바뀌느냐 : filter

f0 = pitch / 배음의 숫자 남자 > 여자

Harmonics 는 Sin wave의 배음(f0)으로 이루어짐

퓨어 톤이 프리퀀시가 높아질수록 내려가는 패턴

입모양을 아 로 하느냐 이 로 하느냐 따라서 다른 소리의 패턴(피크와 마운틴, 산맥)이 나타난다 > 같은 소리를 내면 누구든 같은 패턴

콜라병의 비유> 같은 콜라병을 누가 불든 같은 소리가 난다 > 이거슨 콜라병의 모양에 따라 이미 결정된 것이기 때문이다

Synthesizing Source

스테레오까진 심플 웨이브의 분리된 소리

모노로 만들면 콤플렉스 웨이브

모노 사운드는 처음에 만든 100hz 와 주기가 같음

인지심리학적으로 모노 사운드는 주기가 같은 100hz와 같은 음으로 인식한다

변수

변수명과 치환문의 개념

- 치환문: 입력 시점에 등호 왼쪽의 값에 오른쪽의 값을 할당하여 저장하는 문형.
- 변수명: 등호 왼쪽에 위치하는 문자.
변수명을 입력함으로써 변수에 저장한 값을 불러오는 것이 가능함.
- 등호(=)는 앞뒤의 값이 같다는 것을 의미하는 것이 아님을 유의할 것.
예) a=a+3
- 변수로 할당하지 않으면 그 값은 <u>저장되지 않음</u>. 변수로 할당하면 그 값은 <u>저장됨</u>.
예) a=3을 입력한 뒤 a+3을 입력해보자. 그 후, a를 입력해보라.
- 이미 존재하는 변수명을 활용해 새 값을 변수로 저장하면 <u>기존 값에 덮어쓰게 됨</u>.
-

함수(Function)

- 함수: 특정한 연산 및 기능을 수행하도록 미리 정의해둔 것.
- 형식: 함수명(논항).
- 논항(Argument): 함수가 온전히 기능하는 데 필요한 성분. 괄호 안에 들어가는 성분(들).
예) print("집에 가고 싶다"), len("집에 가고 싶다")
- 함수마다 요구하는 논항의 수와 종류가 서로 다름. 정확한 형식으로 함수를 사용하는 것이 중요함.
- 특히 함수의 논항의 자료형에 유의해야 함. 예) len(34)
- 개별 함수들을 사용하는 법은 각각의 장에서 다시 다룰 것임.

자료형

- 자료형: 프로그래밍을 할 때 쓰이는 숫자, 문자열 등의 자료의 형식.
예) 123(숫자형), "Python"(문자열), [1,2,3,4,5](리스트), (1,2,3)(튜플),
{1,2,3}(집합 혹은 set), {1:"A+", 2:"A"}(딕셔너리)
- type(논항): 자료형을 출력해 주는 함수.
예) type(3), type(3.0), type('3'), type(3+4.0)

숫자 자료형

개요
-숫자 형식으로 이루어진 자료형.
- 정수형(int): 123, -345, 0
- 실수형(float): 123.45, -1234.5, 3.4e1
- 복소수(complex): 1+2j, -3j

- 프롬프트에 내용을 입력하고 엔터를 누르면 그 형식에 맞게 처리함.
예) 3, "글자", [1,2,3], (1,2,3)
예) 3+5를 입력하고 엔터를 누르면 8이라는 결과를 출력함.
- **처음에 공백(Space, tab 등)을 두면** 오류가 발생함. 문자, 숫자 등을 먼저 입력하고 공백을 삽입하는 것은 괜찮음.
- 주석을 # 기호로 다는 것이 가능함. # 이후에 등장하는 문자, 숫자 등은 모두 주석으로 간주하여 코드로 인식하지 않음. 다른 사람들에게 자신이 작성한 코드를 설명할 때 유용함.

리스트 자료형

개요

- 각 요소들을 쉼표로 구분하고 대괄호로 감싸준 자료형.
  예) [1,2,3,4], ['a', 'b', 'c', 'd', 'e'], ['Life', 'is', 'too', 'short'], [1, 2, 'Life', 'is'], []
- 빈 리스트를 생성하는 명령어: list(), []
- 리스트 속에 다른 리스트를 집어넣는 것도 가능함.

인덱싱

- 리스트 자료형도 인덱싱이 가능함. 각 요소들을 왼쪽부터 0번부터 번호를 매김.
- 구체적인 형식은 문자열의 인덱싱과 슬라이싱과 동일함.
- 인덱싱의 결과물은 해당 요소의 자료형과 동일함. (즉, 결과물이 문자열 자료형이면 문자열임)
- 리스트 속의 리스트 역시도 인덱싱이 가능함, 이 경우는 [번호]를 한 차례 더 사용함.

슬라이싱

- 역시 문자열의 슬라이싱과 그 기본 형식은 유사함.
  cf) [시작 번호:끝 번호]에서 끝 번호는 포함되지 않는다는 것을 유의
- 리스트의 슬라이싱의 **결과물은 똑같이 리스트**임.

딕셔너리 자료형

개요

- 대응 관계를 나타내는 자료형, {key1:value1, key2:value2...} 형식으로 나타남.
  예) {'이름':'이영웅', '성별':'남', }
- ':'를 기준으로 왼쪽에 있는 값을 Key, 오른쪽의 값을 Value라고 부름.
- 빈 딕셔너리를 불러오는 함수: dict() 또는 {}
- 딕셔너리 자료형은 Key값을 통해 Value를 불러오는 것이 가능함.
- '형식'은 인덱싱과 동일함, 슬라이싱(형식)은 지원하지 않음.

집합 자료형

개요

- 집합과 관련된 것을 쉽게 처리하기 위해 만든 자료형.
- {} 안에 쉼표로 값을 구분하여 입력함.
- 빈 집합을 만드는 함수: set()

제어문 – if문

개요

- 제어문: 조건에 따라 실행해야 할 명령문을 제어하는 데 사용하는 명령문.
- if문: 제시된 조건이 참인지 거짓인지 판단하여 참이라면 해당 조건에 맞는 상황을 수행함.
  거짓이면 if문 이하의 내용을 실행하지 않음.
- 제어문은 **들여쓰기에 매우 민감함**. **'if 조건문:'에 속하는 모든 문장에 들여쓰기를 해주어야 함**.
- 공백은 space나 tab 중 아무 거나 선택하되, 일관성 있게 사용하는 것이 중요함.

제어문 – while문

(1) 개요

- 반복해서 문장을 수행해야 할 경우에 사용함. (반복문이라고도 부름.)
- while 이후의 조건문이 참인 경우에만 수행문을 처리함.
- 수행문을 모두 마친 경우에는 다시 첫 줄의 조건식으로 돌아와서 수행 여부를 판단함.
- while문 내부에 if문을 활용해 조건을 걸어주는 것도 가능함.

- 제어문 – for문

- (1) 개요

- for문: 리스트, 튜플, 문자열의 첫 요소부터 마지막 요소를 차례로 변수에 대입하여 수행문을 반복함

- 다른 제어문과 마찬가지로 for 이하의 수행문들을 모두 들여쓰기 해 주어야 함.
- for 이후에는 'for문 이하의 수행문에서 활용할 변수'와 'in 자료'가 위치함.

-