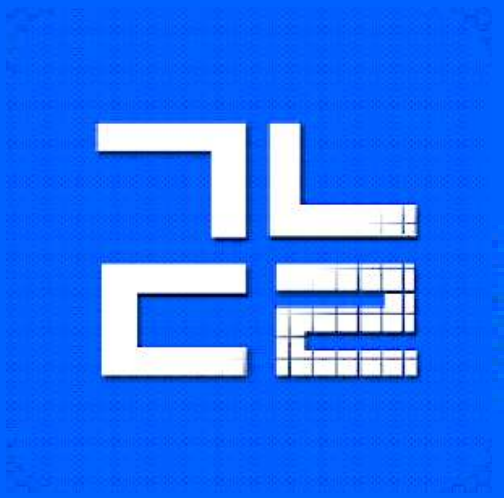
사용 설명서

아라 : 한글 프로그래밍 언어



제 32회 한국정보올림피아드(KOI) 공모 부문 출품작

서울대진고등학교
3학년
홍 승 환

차 례

● 설치와 준비

I. Windows

먼저 Python 3.4 이상 버전이 필요합니다. http://python.org에서 Python 3.4 이상을 설치하여야 정상적인 실행이 가능합니다.[1] 그리고 기본적인 환경 변수가 설정되어 있어야 합니다. 환경 변수는 아라 패키지 파일 내의 **환경_변수_설정하기.bat**로 할 수 있습니다. 혹은 명령 프롬프트에서 **setx PATH "%PATH%;C:₩Python34"**와 같이 명령하거나, 컴퓨터 > 속성 > 고급 시스템 설정[2]에서 환경 변수 설정을 직접 조작할 수 있습니다.

그 후 **Ara_for_Windows** 폴더로 돌아와 **아라_설치.bat**을 실행합니다. 실행 중 Python의 설치 디렉터리를 묻는 말이 나오면, 디렉토리를 입력해주시면 모든 설치가 끝납니다. 그리고, 설치된_파일.txt가 생성되며 이곳에 설치된 파일들이 모두 기록됩니다. 이 시점부터는 ara 명령어를 통하여 전 디렉터리에서 접근 가능하며, '아라 실행기'를 사용할 수 있게 됩니다.

ara 명령어는 **ara [아라 파일] (-o, -n)**의 형태로 사용합니다. 예를 들어, **ara sample.ara -o ../sample/ -n** 과 같은 형식입니다. **ara** 명령어는 몇 가지 옵션을 부여할 수 있습니다.

먼저 **-o** 옵션은, **--output**으로도 부여할 수 있으며, 출력되는 Python 코드 파일의 위치를 지정합니다. 부여하지 않을 시, 현재 프롬프트가 실행되고 있는 곳에 생성됩니다.

다음으로 **-n** 옵션은, **--dontrun**으로도 부여할 수 있으며, 변환 작업 후 해당 파일을 실행할 것인지에 대한 여부입니다. 지정하지 않으면 항상 실행합니다.

II. Mac & Linux

역시나 Python 3.4 이상 버전이 필요합니다. 먼저 터미널에 **python**을 입력하여 현재 설치되었는지 여부와, 버전을 확인하셔야 합니다.[3] 만약 Python 2가 설치되어 있다면, 기존 버전을 삭제하고 버전 3을 설치하거나, 복수 설치하는 방법이 있습니다. 이후 별다른 설정은 필요 없습니다. 터미널의 디렉터리 위치를 **Ara_for_Mac&Linux** 로 설정한 후, **python setup.py install (--record uninstall.txt)**을 실행합니다. 만약 Python 3을 복수 설치하였다면, **python3 setup.py install (--record uninstall.txt)**을 실행하면 됩니다. 괄호 부분을 입력하면, 설치된 파일의 목록이 **uninstall.txt**에 저장됩니다. 후에 **ara** 패키지를 삭제하고 싶을 때, '**cat uninstall.txt | xargs rm -rf**' 등의 명령어를 통하여 패키지를 삭제할 수 있습니다. 명령어의 사용 방법은 'I. Windows'를 참고하십시오.

III. Android

Android 환경에서는 단순히 Google Play Store에서 '아라 : 한글 프로그래밍 언어 - 모바일 변환기' 애플리케이션을 찾아 설치하시면 됩니다. 앱을 실행하시면 안내되는 **QPython3**는 Python 3의 Android 포팅 버전으로, 설치하시면 모바일에서도 바로 변환된 코드를 실행하실 수 있습니다.

1) 본 사용 설명서에서는 Python 3.4를 기준으로 설명합니다. 디렉터리는 C:₩Python34로 가정합니다.

2) 실행 환경을 Windows 7 이상으로 가정합니다. 그 이하에서도 실행 가능합니다.

3) Mac OS X 전체 시리즈와 일부 Linux는 이미 Python이 설치되어 있을 확률이 있습니다.

● 문법과 명령 I. 기본 규칙

1. 구문 기초 문법

아라는 한국어의 문법 구조를 그대로 사용하여 프로그래밍 하는 실험적인 구조를 띠고 있습니다. 특수한 구문을 제외한 거의 모든 구문은 다음의 규칙을 따릅니다 :

[명사](조사) … [동사]하기

예를 들면, **결과를 보여주기**, **개수에 10을 더하기** 등입니다. 이 구조는 한국어의 문법 요소인 "서술어의 자릿수" 개념을 응용한 것으로, …하기는 명령어, 조사와 결합된 명사는 인자의 의미를 가집니다. 그러므로 명령어를 담당하는 서술어의 자릿수에 따라 인자 역시 달라지는 것이 특징입니다. 한국어의 기본적인 문장구조를 기초하여 구문을 만들었기 때문에 프로그래밍에 익숙치 않으신 분들도 매우 익숙하게 다가올 것입니다.

2. 들여쓰기 (Indentation)

아라는 Python에 기초하고 있기 때문에, 들여쓰기를 코드 단락 구분자로 채택하였습니다. 타 프로그래밍 언어의 중괄호의 기능을 담당하는 개념입니다. 들여쓰기의 양은 코드를 쓰시는 분의 마음대로이지만, 아라는 Tab 1회 혹은 Space 4회를 권장하고 있습니다. 같은 들여쓰기 횟수는 같은 깊이의 단락으로 판단되며, 만약 잘못된 위치에서 들여쓰기가 있을 경우 프로그램이 에러를 띄우며 종료될 수 있습니다.

3. 예약어와 이름 지정의 조건

모든 프로그래밍 언어에는 "예약어"라는 것이 있습니다. 예약어란, 해당 프로그래밍 언어의 문법 안에서 쓰이는 단어들로, 이것들을 변수의 이름이나 모듈의 이름 등으로 사용했을 경우 프로그램이 혼돈을 일으킬 가능성이 있습니다. 아라에서 쓰이는 예약어는 다음과 같습니다:

모든 조사(에서, 이/가, 을/를…), 명령어(보여주기, 넣기…),
만약, 아니고, 아니면, 참, 거짓, 함수, Python의 예약어

위 예약어들을 변수 이름으로 지정하는 것은 불가능하다는 점을 인지하고 이름을 지정해야 합니다.[4)]

4) 하지만 예약어들이 "포함되어 있는" 변수명은 설정이 가능합니다. 다만 참과 거짓은 포함시 실행은 정상적이나 변환된 코드가 약간 달라집니다..

예약어 외에도, 변수나 여러 가지 요소들의 이름을 지정할 때 주의해야 하는 것들이 있습니다. 이름은 기호나 숫자로 시작할 수 없으며, 띄어쓰기를 포함할 수 없습니다. 숫자는 중간에 포함되는 것이 괜찮지만, 기호는 불가하니 이 점에 조심하여 프로그래밍 해야 합니다.

4. 주석 (Comment)

아라에는 주석이라는 것이 있습니다. 주석이란, 코딩을 하는 사용자가 설명이나 참고 등의 이유로 코드에 남기는 글을 의미합니다. 일반적으로 코드를 만든 사람 등의 정보를 표기할 때나, 코드 중간 중간에 해당 코드가 무슨 일을 하는지 설명하는 역할을 합니다. 주석은 코드가 아니기 때문에, 컴파일러는 이것을 코드로 인식하지 않고 그냥 넘어갑니다. 주석을 남기는 방법은 Python과 같이, **#**을 입력한 후 그 뒤에 이어서 글을 쓰면 주석이 됩니다. 주석 기능은 현재 웹 테스터와 모바일에서 지원하지 않습니다.

5. 아라 식별 코드

아라는 **.ara** 확장자를 사용합니다만, 달리 발생할 수 있는 에러를 미연에 방지하기 위하여 일종의 식별 코드를 사용합니다. 모든 아라 코드는 맨 마지막에 개행을 한 후에 '**아라**'라고 쓴 후 저장되어야 합니다. 이 규칙이 지켜지지 않은 **.ara** 파일은 아라 코드로 인식하지 않고, 에러를 출력하고 종료합니다. 이 규칙은 모바일과 웹 버전에서는 해당되지 않습니다.

여기서 안내한 특수 규칙 외의 모든 규칙은 Python을 따릅니다.

II. 변수와 값

아라에서 사용할 수 있는 변수의 유형과 값들의 유형은 Python과 매우 닮아있습니다. Python에서 사용할 수 있는 값의 유형들은 모두 사용 가능하다고 해도 과언이 아닙니다.

먼저 숫자입니다. **1**, **-3**, **46.85**, **2.719384** 따위의 값으로, 정수와 소수가 모두 가능하며, 음수와 양수 역시 가능합니다.

다음은 **참**과 **거짓**입니다. 이것은 어떠한 논리식이 존재할 때, 그 논리식의 결과에 대하여 변수가 가지는 값입니다. 예를 들어 **100 > 50**은 **참**을 가지며, **13 < 10**은 **거짓**을 가집니다.

다음은 문자열입니다. **"안녕하세요."**나 **"Hello World!"**와 같이 따옴표로 둘러싸여서 표현되는 하나 이상의 글자들의 집합입니다. 문자열은 더하기와 곱하기 연산이 가능합니다. 문자열과 문자열을 더하면 두 문자열이 이어지며, 문자열에 숫자를 곱하면 그 숫자만큼 해당 문자열이 반복됩니다.

다음은 배열입니다. **[1, 2, 3]**과 같은 형태로 표현되는데, Python에서는 리스트(List)라고 합니다. 여러 가지 다른 값들을 하나로 묶어서 표현할 때 쓰입니다. 들어가는 값은 수나 문자열, 심지어 배열도 가능한 등 모든 형태가 될 수 있습니다.

현재 아라에 구현된 값들은 위와 같으며, 이를 이용하여 여러 가지 형식의 변수를 만들 수 있을 것입니다.

III. 명령어 I : 기본

아라는 앞에서 설명하였듯 한국어의 어순 구조를 본뜬 명령어 체계를 사용합니다. 그러므로 구문 하나하나를 프로그램이 수행하는 "행동"으로 이해하였을 때 가장 정확합니다. 아라에서 사용가능한 명령어는 다음과 같습니다.

먼저 **보여주기**입니다. **출력하기**로도 사용할 수 있고, 을/를 조사에 결합된 값/목적 인자 1개를 받습니다. 예를 들어, **결과를 보여주기**, **"감사합니다."를 보여주기** 등이 있습니다.

다음은 **넣기**입니다. **넣기**는 대입 연산을 의미하며, 을/를 조사에 결합된 값 인자 1개와, 에/에게 조사에 결합된 목적 인자 1개를 받습니다. 예를 들어, **a에 10을 넣기**, **결과에 a*20을 넣기** 등이 있습니다.

다음은 연산자입니다. **더하기**, **빼기**, **곱하기**, **나누기**의 4개의 명령어를 연산 명령이라고 합니다. 각각의 명령어는 을/를 조사에 결합된 값 인자 1개와, 에/에게 조사에 결합된 목적 인자 1개를 받습니다. 예를 들어, **a에 1을 더하기**, **b에 10을 곱하기** 등이 있습니다.

다음은 **끝내기**입니다. 해당 단락의 함수를 끝내버립니다. 조사가 없는 목적 인자를 1개 받으며, '**함수 끝내기**' 이외의 용례는 현재 존재하지 않습니다.

다음은 **내보내기**입니다. **내보내기**는 역시 함수 안에서 쓰이지만, 함수의 결과를 밖으로 내보내려 할 때 쓰입니다. 을/를 조사에 결합된 값/목적 인자 1개를 받습니다. 예를 들어, **a를 내보내기**, **"실행 완료!"를 내보내기** 등이 있습니다.

다음으로 **그만하기**가 있습니다. 그만하기는 후에 설명할 제어문에서 쓰이는 명령어로써, 조사가 없는 목적 인자를 1개 받습니다. 현재 '**반복 그만하기**' 이외의 용례가 존재하지 않습니다.

다음으로 **불러오기**가 있습니다. **불러오기**는 다른 곳에 있는 모듈을 불러오는 용도로 사용되며, 조사가 없는 목적 인자 1개를 받습니다. 예를 들어, **거북이 불러오기** 등이 있습니다.

마지막으로 **입력받기**입니다. 이 명령어는 인자가 제일 많은 명령어로써, 사용자에게 특정 유형의 값을 입력받을 수 있습니다. 을/를 조사에 결합된 목적 인자 1개와 ~(으)로 조사에 결합된 값 조사 1개와, ~(으)로써 조사에 결합된 부사형 인자 1개를 받습니다. 예를 들어, a**를 "a를 입력하세요 : "로 정수로써 입력받기** 등이 있습니다.

기본적인 명령어는 이것이 끝입니다. 위의 여러 "행동"들을 조합하여 여러 가지 알고리즘을 만들어 볼 수 있을 것입니다. 아라 패키지 내부의 **sample** 폴더에서 예제들을 찾아보실 수 있습니다.

IV. 명령어 II : 거북이

아라는 Tkinter[5]와 Python Turtle Graphics에 기반을 둔 일명 "**거북이**" 그래픽 디스플레이를 지원합니다. 흰색 바탕의 화면이 나타나며 화살촉 표식이 한 가운데 표시됩니다. 사용자는 이것을 아라 코드로써 방향과 전/후진 제어를 할 수 있습니다. 이것으로 정보 교과서 예제 중 하

5) Python에서 지원하는 GUI 프로그래밍 툴 킷 중 하나입니다.

나인 “달팽이 껍질 그리기” 등의 여러 알고리즘들을 시도할 수 있습니다.

일반적인 아라 코드에는 거북이가 존재하지 않습니다. 거북이를 표시하기 위해서는, 먼저 거북이를 불러와야 합니다. **거북이 불러오기**라고 코드에 적으면, 거북이 그래픽 엔진이 불러와집니다.

다음은 거북이를 소환할 차례입니다. **[이름] 거북이 등장**이라고 코드에 적으면, 해당 이름을 가진 거북이가 화면 위에 나타납니다. 여기서 지정한 이름은 후에 거북이를 조작하는 것에 쓰입니다. 예를 들어, **홍승환 거북이 등장** 이라고 하면, 이 거북이의 이름은 **홍승환**이 되며, 이 이름으로 조작합니다. 이 ‘**등장**’은 여러 마리의 거북이를 동시에 출현시킬 수 있으므로, 이름이 다른 거북이를 여러 개 소환 후 각각 조작할 수 있습니다.

다음은 전진과 후진입니다. **[이름] [픽셀 수]만큼 앞으로**를 통하여 주어진 픽셀 수만큼 해당하는 이름의 거북이를 앞으로 보낼 수 있습니다. 거북이는 앞으로 걸어가며 지나온 길에 직선으로써 흔적을 남깁니다. **앞으로** 대신 **뒤로**를 써서 뒤로 복귀하게 할 수 있습니다. 예를 들어, **홍승환 100만큼 앞으로**, **홍승환 50만큼 뒤로** 등이 있습니다.

다음은 회전입니다. **[이름] 좌회전**, **[이름] 우회전** 을 사용하면 해당하는 이름의 거북이를 좌회전, 우회전시킬 수 있습니다. 혹은, **[이름] 왼쪽으로/오른쪽으로 [각도]도 회전**을 사용하여 지정한 각도만큼만 회전시킬 수 있습니다. 예를 들어, **홍승환 좌회전**, **홍승환 오른쪽으로 60도 회전** 등이 있습니다.

위 명령어들을 이용하여 여러 가지 그림 관련 알고리즘들을 짜볼 수 있을 것입니다. 역시 아라 패키지 내부의 **sample** 폴더에서 예제들을 찾아볼 수 있습니다.

V. 제어문

제어문은 프로그램을 어떠한 논리 조건에 따라 분기하거나, 특정 부분을 반복시킬 때 쓰는 일련의 명령어 집합입니다. 아라에서는 다음과 같은 제어문들을 사용할 수 있습니다.

먼저 **만약**입니다. **만약**은 말 그대로 후술되는 조건식에 의하여 프로그램을 분기할 때 쓰입니다. **만약 ~(이)(하)면**의 형태로 쓰입니다. 예를 들어, **만약 a가 0보다 크면**, **만약 b가 3이 아니면**, **만약 c가 10 이상이면** 등이 있습니다. 그리고 **아니고 만약**과 **아니면**이 있습니다. **아니고 만약**의 경우는 위에 적었던 **만약** 조건문이 **거짓**으로 판단되어 실행되지 않았을 경우, **아니고 만약**의 조건문으로 다시 분기합니다. **아니고 만약**은 **만약**과 **아니면** 사이에서 몇 번이고 반복할 수 있습니다. **아니면**은 위에 적은 모든 **만약**과 **아니고 만약**이 **거짓**으로 판단되었을 때에, 즉 이도저도 아닐 때에 실행해야 하는 코드를 적는 단락입니다. 이것의 예는 아라 패키지 안의 **sample** 폴더에서 확인하실 수 있습니다.[6)]

6) 만약, 아니고 만약, 아니면은 타 프로그래밍 언어의 if, else if, else에 해당합니다.

다음은 **반복하기**입니다. **반복하기**는 여러 가지 유형이 있습니다. 첫 번째로 **무한 반복하기**가 있습니다. **무한 반복하기**는 말 그대로 들여쓰기 되어있는 하위 단락을 무한 번 반복합니다. 내부에서 조건으로 분기하여 ‘**반복 그만하기**’로 반복을 멈출 수 있습니다. 해당 명령이 내려지기 전까지 계속 반복합니다. 두 번째로 **n번 반복하기**가 있습니다. n에 들어가는 숫자만큼 반복합니다. 이것 역시 필요할 경우 내부에서 ‘**반복 그만하기**’로 반복을 멈출 수 있습니다. 마지막으로 **넣어가며 반복하기**가 있습니다. 배열이나 **범위(시작, 끝)**으로 묘사되는 범위에서 순서대로 값을 추출해 지정한 변수에 담고, 해당 배열/범위에 들어있는 값의 수만큼 반복합니다. 예를 들어,

범위(1, 50)를 x에 넣어가며 반복하기가 있습니다.

VI. 함수

함수는 프로그램 소스 코드에서 일정한 동작을 수행하는 코드 집합을 의미합니다. 함수를 사용하면, 계속 반복되는 코드를 함수로 묶음으로써 프로그래밍을 좀 더 편하게 할 수 있다는 장점이 존재합니다.

함수는 **함수 함수이름(인수, 인수, ...):** 의 형태로 선언합니다. 인수는 함수를 실행할 때 필요한 입력 요소들을 의미합니다. 예를 들어, 다음과 같습니다 :

함수 합_구하기(첫째, 둘째):
    첫째 + 둘째를 내보내기

위의 예제에서, **합_구하기**는 함수의 이름이며, **첫째**와 **둘째**는 인수의 이름입니다. 이 함수는 두 개의 값을 인수로 받아서 두 값의 합을 반환하는 함수입니다. 함수는 들여쓰기로 묶인 한 단락이 되어야 합니다. 컴파일러가 예상치 못한 들여쓰기 규칙을 발견하는 경우에는 프로그램이 에러를 띄우며 종료되므로, 들여쓰기에 신중하여야 합니다. 위와 같이 함수를 선언한 다음에는, 다음과 같이 사용합니다.

a에 10을 넣기
b에 20을 넣기
c에 합_구하기(a, b)를 넣기
c를 보여주기

여기서는 **a**와 **b**를 함수에 대입한 다음 내보내어지는 값을 **c**에 넣고 **c**를 출력하고 있습니다. 이처럼 함수는 다른 구문 사이에 있는 값 인자 자리에 대신 들어가서 실행될 수 있습니다. 위 상황에서는, **c**에 30이 들어감을 알 수 있습니다.