0. 프로그래밍 기초

1. 컴퓨팅적 사고

인간의 사고(사고력, 논리력, 창의력)와 컴퓨터의 능력(계산 등)을 통합한 사고

분해, 추상화, 패턴 분석, 알고리즘, 자동화 등으로 구성됨.

분해 : 큰 문제를 작은 문제로 쪼개는 것. -> 이런 경우 여러 사람이 문제에 참여할 수도 있음.
추상화 : 복잡한 내용에서 핵심적 내용을 간추려 내는 것.
패턴 분석 : 규칙을 찾아내는 것 -> 자동화를 위한 중간 단계.
자동화 : 사람의 개입 없이 컴퓨터가 혼자 수행하는 것.

2. 알고리즘

주어진 문제를 해결하기 위해 입력이 주어졌을 때 유한시간 내에 결과를 얻어내는 잘 정의된 명령들의 절차
입력, 출력, 절차(순서)가 반드시 필요하고, 유한시간 내에 종료되어야 하며, 잘 정의되어 있어야 함.

1) 알고리즘의 표현 방법

1. 자연어로 표현
2. 순서도로 표현
3. 의사코드(스도코드)로 표현

2) 의사코드(스도코드)

가상의 코드.
프로그래밍 형식으로 알고리즘을 표현한 것.

아이디어의 표현이 목적이기 때문에 특정한 형식이 정해져 있지는 않음.

3. 순서도

알고리즘을 표현하는 방법 중 하나

1) 그리는 방법

화살표와 도형을 사용함.

- 최근 방식

(평행사변형은 입력 뿐만이 아니라 출력도 나타냄)

+ Flowgorithm

순서도를 쉽게 다룰 수 있게 해주는 프로그램.

순서도의 실행 결과와 소스 코드까지 확인할 수 있음.

화살표 위에서 마우스 오른쪽 버튼을 눌러 도형을 추가할 수 있음.
도형을 더블 클릭하여 내용 입력
문자열은 따옴표로 묶어서 작성해야 함.
"배정" 도형이 계산식 도형임.
출력 시 변수의 값을 출력하려면 "<문자열>" & <변수이름> 형식으로 출력 도형에 작성해 줘야 함.

1. 파이썬 기초

1. 파이썬 사용환경 구축

1) 파이썬 IDLE

간단한 파이썬 개발환경

기본적인 기능만 가능, 라이브러리를 추가 설치해야 다른 기능들 사용 가능.

2) 아나콘다

파이썬, 데이터사이언스, 머신러닝을 위한 통합 개발환경 (패키지)

관련 프로그램들과 라이브러리들을 제공함.

실행 시 여러 프로그램들을 실행할 수 있음.

3) 주피터 노트북

아나콘다 안에 들어 있음.

아나콘다에서 실행 시 웹을 띄우는데, 여기에서 파이썬을 다룰 수 있음.

2. 파이썬 IDLE의 모드

1) 콘솔 모드

쉘 모드와 유사하지만 콘솔에서 작업하는 모드.

2) 쉘 모드

하나의 명령 단위로 명령을 실행하는 모드.

3) 코드 에디터 모드

여러 줄의 소스 코드를 입력한 후 한 번에 실행하는 모드.

메모 포함[이1]: + 윈도우 cmd에서도 비슷한 것을 쓸 수 있다.
1. windows키 + R키를 누른다.
2. cmd를 입력하여 cmd 창을 띄운다.
3. python 명령을 입력한다
-> 파이썬 프로그램이 실행된다.

3. 파이썬 기초 문법

1) ;의 유무

파이썬에서는 ;를 써도 되고 안 써도 됨.

두 개 이상의 명령을 하나의 줄에 작성할 때는 명령들을 구분하기 위해 사용할 수 있음.

2) 따옴표

파이썬에서는 큰따옴표와 작은따옴표를 구분하지 않음.

단, 여는 따옴표와 닫는 따옴표는 동일한 것이어야 함.

3) 주석 (설명문)

실행되지는 않지만 효율적인 표현을 위한 문법.

: 한 줄 주석. (c 에서의 //와 방식은 동일)
""" . . . """ : 여러 줄 주석. 따옴표를 세 개 연달아 사용함. (c 에서의 /* . . . */와 방식은 동일)

4) 자료형

파이썬에서는 기본적으로 자료형을 명시하지 않음.
자동으로 결정됨.

4. 파이썬 특징

- 파이썬은 1991 년 귀도 반 로섬이 만듦.
- 파이썬은 인터프리터 방식의 언어임.
- 파이썬은 여러 오픈소스 라이브러리가 제공됨.
- 파이썬은 쉽고, 전문가들이 사용함.

메모 포함[이2]: 컴파일러 방식
-> C, JAVA
-> 빠름

인터프리터 방식
-> Python
-> 쉬움

2. 연산자

1. 산술 연산자

+ : 더하기
- : 빼기
* : 곱하기
/ : 실수 나누기 -> 실수 나누기에 정수만을 쓸 수도 있음.
// : 정수 나누기
% : 나머지 연산
** : 거듭제곱

메모 포함[이3]: c와는 달리 실수 나누기와 정수 나누기가 구분되어 있는 것 유의하기.

메모 포함[이4]: a ** b 는 a^b이다.

2. 배정 연산자

= 을 사용함.

배정 연산자의 왼쪽에는 단 하나의 순수한 변수만 올 수 있음.

+ 복합 배정 연산자

c 에서와 동일한 원리로 산술 연산자와 배정 연산자를 줄여 쓸 수 있음.
지금까지 배운 모든 산술 연산자는 배정 연산자로 사용할 수 있음.

(ex. a = a ** 3 은 a **= 3 으로 작성할 수 있음.)

+ 증감 연산자 지원 안됨

c 에서와는 달리 파이썬에서는 ++과 --연산자를 지원하지 않음.

--x 나 ++x 등으로 작성해도 오류는 발생하지 않음 -> +/- 연산자가 두 번 수행된 것. 아무 변화도 없음.

3. 비교 연산자 (관계 연산자)

c 와 동일함.

단, 파이썬에서는 부등식을 더 편리하게 사용할 수 있음.
80 <= x < 90 과 같이 작성하면 수학에서처럼 취급할 수 있음.
즉, x 가 80 보다 크거나 같고 90 보다 작으면 참임.

== : 두 값이 같으면 참.
!= : 두 값이 다르면 참.
>, < : 부등식이 참이면 참.
>=, <= : 부등식이 참이면 참.

4. 논리 연산자

and : 두 조건이 모두 참이어야 참. -> c 에서 &&
or : 두 조건 중 하나라도 참이면 참. -> c 에서 ||
not : 조건이 거짓이면 참. -> c 에서처럼 식의 왼쪽에 작성함.

메모 포함[이5]: a and b이면 말 그대로 a와 b 모두 참이어야 한다는 것이다.

메모 포함[이6]: a or b이면 말 그대로 a나 b 둘 중 하나만 참이면 된다는 것이다.

+ is, is not 키워드

is : 할당된 메모리의 주소가 같으면 참.
is not : 할당된 메모리의 주소가 다르면 참.

파이썬에서는 자주 사용되는 몇 개의 값들을 메모리에 기본적으로 생성하고 해당 값을 변수에 배정할 때 정해진 주소를 할당함.
x = 100; y = 100; x is y 를 작성하면 True 가 나옴. 100 이 기본적으로 저장되는 값이기 때문.

+ True, False

참과 거짓은 각각 True. False 를 사용하여 표현할 수 있음.

Boolean 타입의 데이터임.

대문자 사용 유의.

3. 변수

1. 변수

변수 선언을 따로 하지는 않고 대입할 때부터 명시함.
c 에서처럼 좌측에는 단 하나의 변수만 들어갈 수 있음.

```
<변수 이름> = <값>
```

2. 변수 이름 규칙

1. 숫자, 영어(대소문자 구분), 언더라인(_) 사용 가능
2. 영어 이외의 언어들도 사용 가능
3. 숫자로 시작 불가능
4. 공백 불가능
5. 이미 지정된 키워드는 사용 불가능
6. 언더라인(_)을 제외한 특수문자는 사용 불가능

4. 함수

1) 함수 정의

함수 이름은 변수 이름을 정하는 규칙과 같음.

파라미터가 없을 경우 그냥 비워 두면 됨.

반환값이 필요 없는 경우 return 은 생략해도 됨.

```
def <이름>(<파라미터>) :
        명령문1
        명령문2
        return <결과값>
```

정의 시 파라미터에 작성하는 변수에 값을 배정해주면 해당 값이 default 값이 됨.
호출 시에 아무것도 작성하지 않으면 해당 값이 사용됨.
만약 함수 정의 시에 배정하지 않았는데 인자를 부족하게 작성한 경우, 오류 발생함.

호출 시에 인자를 파라미터 개수보다 적게 작성하면 인자의 값은 앞쪽 파라미터부터 들어감.

2) 함수 원형 선언

이 수업에서는 배우지 않았음.

c 에서처럼 파이썬에서도 호출 전에 함수의 존재를 알 수 있어야 하므로, 함수는 코드의 위쪽에 작성해야 함.

3) main 함수

파이썬에는 main 함수가 따로 없음.

그냥 빈 공간에다가 작성하면 main 처럼 취급됨.

+ c 에서의 인자와 매개변수를 통틀어 파라미터라고 하는 듯?

메모 포함[이7]: ex.
def test(a=10, b=9) :
    명령어
이 함수를 호출할 때 test(5) 이렇게 하면 a에는 5가 들어가고 b에는 default값인 9가 들어간다.

메모 포함[이8]: ex.
파라미터가 2개인 경우 test(5,) 이렇게 써도 되고 test(5) 이렇게 써도 된다.

메모 포함[이9]: 결국 함수들은 위에 작성하고 맨 밑 빈 공간에 c에서 main 함수에 해당되는 내용을 넣어야 한다.

메모 포함[이10]: main 함수를 쓴 후 코드 맨 마지막에 main() 등으로 작성해서 main 함수처럼 사용할 수는 있다.

2. print 함수

파이썬의 기본 출력 함수

쉘 모드에서는 문자열, 숫자, 변수, 수식만 입력해도 print 함수를 생략한 것으로 취급되어 적용됨.
코드 편집기 모드에서는 print 를 생략할 수 없음.

메모 포함[이11]: ex. 1 + 2 는 print(1 + 2)와 동일하다.

1) 사용 방법

print("<문자열>")
print(<내용>)

괄호 안에 문자열, 숫자, 변수, 수식 등을 작성하여 출력할 수 있음.
문자열은 따옴표 안의 문자열이 출력됨.
숫자, 변수, 수식은 해당하는 값을 출력함.

메모 포함[이12]: 문자열은 당연히 따옴표로 씌워 줘야 한다.

여러 개의 인자를 쉼표로 구분하여 작성할 수 있음.

여러 줄에 걸쳐서 문자열을 작성하는 경우 따옴표를 세 개 사용함.

메모 포함[이13]: ex. print("""Hello~~!!!
World~~!!""")

출력:
Hello~~!!!
World~~!!

-> 개행하여 작성하면 개행한 것도 문자열에 반영된다.

2) Escape 문자

c 에서와 동일한 기능을 함.

₩n : 개행.
₩t :탭.
₩' : 작은따옴표를 그대로 출력
₩" : 큰따옴표를 그대로 출력
₩₩ : 역슬래시를 그대도 출력
₩b : 백스페이스. (역방향으로 한 글자 지움) -> IDLE 쉘 모드에서는 실행이 안 될 수도 있음.

+ %는 그냥 적어도 출력됨.

3) 끝 문자 지정 (end 파라미터)

print 의 마지막 문자를 지정할 수 있음.

print("<문자열>", <변수>, end="<문자>")

end 파라미터는 맨 마지막 인자에 작성해 줘야 함.

""과 같이 아무것도 작성하지 않으면 아무것도 안 들어감.

default 값은 ₩n 임. (print 로 어떤 것을 출력하면 마지막에는 자동으로 개행됨)

4) 구분 문자 지정 (sep 파라미터)

print("<문자열>", <변수>, sep="<문자>")

쉼표 자리에 들어가는 문자를 지정할 수 있음.

sep 파라미터는 맨 마지막 인자에 작성해 줘야 함.

""과 같이 아무것도 작성하지 않으면 아무것도 안 들어감.

default 값은 공백 문자 하나임. (print 에서 쉼표로 이으면 사이에 빈칸 하나가 자동으로 들어감)

메모 포함[이14]: separate. 쉼표로 떨어져 있는 인자들 사이에 들어가는 것이다.

+ 양식문자

c 에서와는 달리, 파이썬에서는 양식문자를 모든 문자열에서 사용 가능한

문자열의 바로 오른쪽에 양식문자에 들어갈 값을 명시함.

print 함수에서 유용함.

%d : 정수. 자릿수와 사용 방식을 결정하는 특정 형식이 있음. (c 와 동일)

%f : 실수. 자릿수와 사용 방식을 결정하는 특정 형식이 있음. (c 와 동일)

%g : 정수, 실수 자동결정

%s : 문자열

%c : 문자

%o : 8 진 정수

%x : 16 진 정수

메모 포함[이15]: c의 변환명세와 유사하다.

메모 포함[이16]: %<부호><숫자>d (c랑 거의 똑같음)
부호로 -를 명시하면 왼쪽 정렬된다.
부호로 +를 명시하면 오른쪽 정렬이 아니라 +가 숫자 앞에 붙어서 출력된다.

전체 출력 자릿수를 숫자에 명시한다.
c에서처럼 이 숫자는 크게 의미가 없다.
정수 부분은 자릿수를 줄이지 못한다.
오른쪽, 왼쪽 정렬을 위한 것으로 생각하자.

메모 포함[이17]: %<부호><숫자>.<숫자>f (c랑 거의 똑같음)
부호로 -를 명시하면 왼쪽 정렬된다.
부호로 +를 명시하면 오른쪽 정렬이 아니라 +가 숫자 앞에 붙어서 출력된다.

전체 출력 자릿수를 첫째 숫자에 명시한다.
c에서처럼 이 숫자는 크게 의미가 없다.
항상 두 번째 숫자가 우선되고 정수 부분은 자릿수를 줄이지 못한다..
오른쪽, 왼쪽 정렬을 위한 것으로 생각하자.

소수점 몇 째 자리까지 출력할 지를 둘째 숫자에 명시한다.
c에서처럼 . 만 명시하면 0으로 취급된다.
c에서처럼 반올림되어 적용된다.
c에서처럼 6자리가 default이다.

메모 포함[이18]: 쉼표와 헷갈리지 않도록 주의.

1) 하나의 양식문자 사용

"<문자열>" % <변수>

양식문자에 들어갈 값을 %로 구분하여 지정함.

2) 2 개 이상의 양식문자 사용

"<문자열>" % (<변수>, <변수>, . . .)

양식문자에 들어갈 값들을 %로 구분하여 지정함.

이때 값들은 괄호로 묶고, 쉼표로 구분함.

3. input 함수

<변수> = input(<문자열>)

파이썬의 기본 입력 함수

문자, 문자열로 입력받음.

input()만 입력하면 키보드를 통해 입력 받고 값은 사용하지 않음.
입력 받은 값을 변수에 저장하려면 input()을 해당 변수에 배정해줘야 함.

괄호 안에 문자열, 숫자, 변수, 수식을 작성해서 프롬프트 기능을 사용할 수 있음.
이때 문자열은 따옴표로 묶어야 됨.

프롬프트를 사용해서 값을 입력 받고 변수에 저장해도, 입력 받은 값 만을 저장함. (프롬프트는 저장 안함)

4. 형변환 함수

input 함수를 사용할 때 텍스트가 아닌 숫자 상수가 필요하다면 형변환 함수를 사용해야 함.

1) int()

int(<텍스트>)

형을 정수로 변환하는 함수

아스키 값 등으로 변환하는 것이 아니라, 해당 텍스트와 동일한 형태의 정수로 변환함.
("3" -> 3)

x = int(input()) 등으로 사용함.

2) float()

float(<텍스트>)

형을 실수로 변환하는 함수

아스키 값 등으로 변환하는 것이 아니라, 해당 텍스트와 동일한 형태의 실수로 변환함.
("3.14" -> 3.14)

x = float(input()) 등으로 사용함.

+ 정수와 문자의 연산

파이썬에서 정수와 문자열, 문자는 연산이 안 됨.

(ex. '0' + 1 이렇게 작성하면 오류 발생함)

5. turtle 라이브러리(모듈)

1. turtle 모듈

turtle 모듈을 사용함.

```
import turtle
```

2. 거북이 객체 생성하기

```
import turtle
<이름> = turtle.Turtle()
```

turtle.Turtle() 함수를 사용하여 위와 같이 작성하면 명시한 이름에 해당하는 객체를 마음대로 사용할 수 있음.
함수 이름 앞에 <이름>. 을 명시하면 해당 거북이 객체에게 함수가 적용됨.

여러 개의 객체를 만들어서 여러 개의 거북이를 사용할 수 있음.

3. 여러 개의 도형 그리기

반복문 활용

4. 거북이 관련 명령들

1) 움직임, 그림 관련

forward(a) : 앞으로 a 만큼 이동 -> fd 로 사용 가능

backward(a) : 뒤로 a 만큼 이동 -> bk 로 사용 가능

right(a) : 오른쪽으로 a 도 만큼 회전 (degree 임) -> rt 로 사용 가능

left(a) : 왼쪽으로 a 도 만큼 회전 (degree 임) -> lt 로 사용 가능 (소문자 엘)

circle(a, b, c) -> 호출 시 작성한 인자의 개수에 따라 의미가 달라짐.

speed(a) : 거북이의 속력을 a 로 설정 -> 1~10 는 숫자가 커질수록 빠름. 나머지 숫자들은 속력이 10 보다 빠르고 전부 속력이 같음.

home() : 중심으로 순간이동 -> (0, 0)인 지점임

dot(a) : 크기가 a 인 점 그림 -> 움직이며 그리는 것이 아니어서 penup() 상태에서도 그려짐.

goto(a, b) : 거북이 좌표를 (a, b)로 변경.

setx(a) : x 좌표를 a 로 변경.

sety(a) : y 좌표를 a 로 변경.

xcor() : 거북이의 x 좌표값 리턴.

ycor() : 거북이의 y 좌표값 리턴.

2) 거북이 설정 (거북이가 기준이기 때문에 워딩이 단순함)

shape() : 개체의 모양 설정 -> shape()을 작성하지 않았다면 기본값이 적용됨.

shapesize() : 개체의 크기 설정 -> size()로 쓰는 지알아 하기 말자.

color(a) : 거북이 색깔을 a 로 설정 -> a 에는 문자열로 색깔을 작성 (웬만한 건 다 있음)

register_shape(a) : 이미지 파일로 개체 모양 설정 a 에는 파일 이름 작성.

3) 펜 설정

width(a) : 펜의 두께를 a 로 설정

pencolor(a) : 펜의 색깔을 a 로 설정 -> 펜 색을 지정하지 않은 경우 거북이 색이 펜 색이 됨.

penup() : 펜을 들어올림. -> 이 상태에서는 움직일 때 선을 안 그림

pendown() : 펜을 내려놓음 -> 이 상태에서는 움직일 때 선을 그림

메모 포함[이19]: 실수하기 쉽기 때문에 어떻게 그려질지를 머릿속으로 떠올려 봐야한다.

메모 포함[이20]: 인자가 a 1개인 경우
-> 반지름이 a인 원을 그리며 이동,

인자가 a, b 2개인 경우
-> 반지름이 a인 원을 b도 만큼만 그리며 이동

인자가 a, b, c 3개인 경우
-> 반지름을 a인 원을 b도 만큼만 그리는 궤적을 따라 이동하며, 선분이 c개인 도형을 그린다. 이때 최종적으로 그려지는 도형에서 선분을 계산한다.

메모 포함[이21]: set x. 읽는 순서대로다.
x만 설정하는 것. x좌표값만 바뀐다.

메모 포함[이22]: x coordinate. 읽는 순서대로다.
coordinate : 좌표.

메모 포함[이23]: turtle, arrow, square, circle, triangle, classic 등이 있다.
기본값은 화살표 모양이지만 arrow는 아니고, 끝이 약간 파인 화살표이다.

메모 포함[이24]: 펜 색은 그려지는 선의 색
거북이 색은 개체의 색깔

4) 배경 설정

bgcolor(a) : 배경 색상을 a 로 설정 -> a 로는 color()처럼 문자열을 쓸 수도 있고, 숫자 코드를 쓸 수도 있음.

bgcolor(a, b, c) : r, g, b 값을 각각 a, b, c 에 지정하여 배경 색깔 설정.

screensize(a) : 배경의 크기를 a 로 설정.

Screeen() : 배경 화면을 가져옴.

bgpic() : 배경 화면을 그림으로 설정.

메모 포함[이25]: RGB 코드. 알 필요 없다.

메모 포함[이26]: x = turtle.Screen()
screen.bgpic("back.png") 등으로 작성한다.

메모 포함[이27]: background picture

5) 기타

clear() : 화면 지우기

distance(a, b) : 현재 거북이의 위치와 좌표 (a, b) 사이의 거리를 리턴.

textinput(a, b) : a 에는 입력 창의 제목, b 에는 입력 안내 메시지를 받아서 출력. -> 창을 띄워 입력받음.

write(a) : a 에 작성한 내용을 현재 거북이 위치에 출력.

fillcolor(a) : a 에 작성한 색깔로 도형 색깔 채우기.

메모 포함[이28]: turtle.fillcolor("orange")
turtle.begin_fill() -> 색칠 시작
. . .
turtle.end_fill() -> 색칠 종료

-> 색칠을 시작하면 닫혀진 공간이 자동으로 색칠된다.
-> fillcolor()를 생략하면 검은색이 default로 색칠된다.

6. 조건문

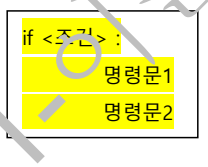
1. if 조건문

조건이 참인 경우 명령을 실행하는 조건문.

오른쪽에 명시한 if의 형식을 지키면서 c에서와 동일하게 else와 else if를 사용할 수 있음.
else는 그대로 else로, else if는 줄여서 elif로 표현함.

1) 형식

들여쓰기로 포함되는 명령들을 구분함. (tab키)



조건에는 괄호를 씌워도 되고 안 씌워도 됨.

아무 명령도 작성하지 않으면 오류 발생함.
대신 pass 키워드를 작성하면 됨.

if 와 else, elif는 같은 레벨이어야 함. -> 들여쓰기 상에서 일치해야 의도대로 기능함.

7. 반복문

1. break, continue

반복문에서 사용하는 키워드들임.

break : 반복문을 깨는 키워드
continue : 현재 남은 루프를 건너뛰고 다음 루프로 가는 키워드.

2. while 반복문

명시한 조건이 참이면 루프를 돌아 명령들을 수행하는 반복문.

```
while <조건> :
    명령문1
    명령문2
```

1) 형식

들여쓰기로 포함되는 명령들을 구분함. (tab키)

조건에는 괄호를 씌워도 되고 안 씌워도 됨.

아무 명령도 작성하지 않으면 오류 발생함.
대신 pass 키워드를 작성하면 됨.

2) 무한 루프

조건에 1이나 True를 작성하여 무한루프를 만들 수 있

4. for 반복문

list의 각 원소들이 순서대로 for 옆의 변수에 배정되며 루프가 도는 반복문.

```
for <변수> in <list>:
    명령문1
    명령문2
```

1) 형식

변수에는 일반적인 변수를 작성함.

들여쓰기로 포함되는 명령들을 구분함. (tab키)

아무 명령도 작성하지 않으면 오류 발생함.
대신 pass 키워드를 작성하면 됨.

8. list 자료형

한 번에 여러 개의 값들을 다룰 수 있게 하는 자료형

1. list 생성

수열(list)을 만드는 방법은 단순 명시, list(), range()로 세 가지가 있음. (수식까지 하면 4 가지)

list 는 출력 시 대괄호([])를 씌워서 출력됨.

list 안에 list 가 원소로 들어갈 수도 있음.

list 에는 문자열, 소수, 정수 등의 값이 들어갈 수 있음.
하나의 list 에 다른 자료형의 값이 들어갈 수도 있음.

1) 단순 명시

별다른 형식 없이 명시하는 것만으로도 list 를 생성할 수 있음.

[<값>, <값>, . . .]

각괄호 안을 비워 두면 비어 있는 list 가 만들어짐.

2) list()

수열 생성 함수.

list([<값>, <값>, . . .])

list()의 파라미터로 아무것도 작성하지 않으면 비어 있는 list 가 만들어짐.
(list() 만 작성)

3) range()

특정 패턴을 갖는 수열(리스트)을 만드는 함수.

range(시작값, 종료값, 증가값)

각 인자에는 정수를 적음. -> 실수는 적을 수 없음.
인자의 순서는 c 언어의 for 와 유사함.

시작값부터 시작해서 종료값보다 작은 값까지 증가값만큼 커지는 수열을 생성함.
종료값은 포함되지 않음.

시작값을 생략하면 default 로 0 이 적용됨.
증가값을 생략하면 default 로 1 이 적용됨.
종료값은 생략할 수 없음.
이를 이용해서 range(<숫자>) 형식으로 자주 쓰임.

인자를 두 개만 작성하면 각각 앞부터 적용되어 시작값과 종료값으로 취급됨.

메모 포함[이29]: 정확히 같은 지는 모르겠지만 일단 이 필기에서는 같은 것으로 취급한다.

메모 포함[이30]: 생성한 다음에 변수에 배정을 해 줘야 사용할 수 있다.

range()함수로 생성한 list를 변수에 배정하려면 list()안에 넣어서 배정해야 한다.
-> 이거 관련해서는 확실하지 않은 것들이 있어서, range()는 일단 웬만해선 for에서만 사용하자.

메모 포함[이31]: ex. [1, 2, [1, 2, 3]]

메모 포함[이32]: ex. [1, "Hello~!!", 2, 3.3]

메모 포함[이33]: 비어 있는 list를 만든 후에 append() 등의 함수를 사용하여 값을 넣어줄 수 있다.

메모 포함[이34]: 단순 대입과 별 차이는 없음.

메모 포함[이35]: ex. range(1, 11, 1)
-> [1, 2, 3, 4, 5, 6, ,7 ,8 9, 10]

range(a, b, 1)은 c언어의
for(int i = a; i < b; i++)과 유사함.

메모 포함[이36]: ex. range(5) 하면 [0, 1, 2, 3, 4]가 됨.

2. 인덱스

인덱스를 사용하여 변수에 배정한 list 의 여러 원소 중 하나를 특정하여 사용하거나 배정 수 있음.

c 의 배열처럼 인덱스는 0 부터 시작함.

```
<변수이름>[<인덱스>]
```

3. list를 반복문으로 다루기

일반적으로 for x in <list>에 있는 자리에 list를 넣어서 값을 사용함.

c에서 배열을 반복문으로 다루는 것처럼 list도 인덱스에 변수를 넣어 반복문으로 다룰 수도 있음.

4. append()

```
<변수이름>.append(<값>)
```

list의 맨 뒤에 항목을 추가하는 함수.

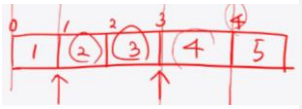
변수 이름에 list를 배정한 변수의 이름을 작성함.
값에 추가할 값을 작성함.

5. slicing

```
<변수이름>[<인덱스>:<인덱스>]
```

list를 쪼개서 사용하는 것.

명시한 인덱스 부분만큼의 원소들만 가져와서 사용
앞에 인덱스부터 시작해서 뒤에 인덱스 바로 앞 것까지 포함함.



인덱스는 0부터 시작하는 것 유의.

앞에 인덱스를 생략하면 리스트의 맨 앞부터 적용됨.
뒤에 인덱스를 생략하면 리스트의 끝까지 적용됨.

+ 수식으로 list 생성하기

오른쪽과 같이 작성하면 list 의 값이 명시한 변수에 배정되고, 수식을 계산하여 그 값을 list 의 원소로 함.

```
[<수식> for <변수> in <list>]
```

메모 포함[이37]: ex. [x ** 2 for x in range(5)]
-> [0, 1, 4, 9, 16] 이다.

9. 모듈

1. 모듈

c언어의 #include와 유사한 문법.

라이브러리의 모듈을 가져와서 그 안에 있는 함수를 사용할 수 있음

import <모듈>
import <모듈> as <string>
from <모듈> import <함수 이름>

메모 포함[이38]: import 수입하다.
라이브러리를 수입해 오는 것이다.

1) 형식

모듈을 불러올 때는 파일 이름만 명시하면 됨.

기본적인 형태는 import <모듈>임.
이때 해당 모듈에 들어있는 함수를 사용할 때는 함수 이름 앞에 <모듈>.을 붙여야 함.

import <모듈> as <string>로 작성할 수도 있음.
함수 앞에 <모듈>. 대신에 <string>.을 명시할 수 있음.

from <모듈> import <함수 이름>로 작성하여 지정한 함수를 가져올 수 있음.
가져온 함수는 사용할 때 <모듈>.을 붙이지 않아도 됨.
함수 이름에 *을 적으면 해당 모듈 내의 모든 함수를 가져옴.

2. 여러가지 모듈들

1) time

time.sleep() : 인자에 작성한 수 만큼 프로그램을 잠깐 멈추는 함수. (단위는 sec임)

2) random

random.randint(a, b) : a부터 b까지의 정수 중 하나를 리턴함. (Random int)

3) winsound (windows sound)

winsound.Beep(a, b) : 소리 출력 함수. a는 소리의 높이이고, b는 출력되는 시간임. (시간의 단위는 밀리 초임)