파이선으로 쉽게 배우는

기초 프로그래밍

Basics of Programming

in Python for easy learning

국형준 지음

빈칸 채우기 과제에 대한 답

1-1 빈칸 채우기

1. 프로그램은 (~~인간~~ , 컴퓨터)에 의한 문제해결을 위한 논리적 절차를 명세한다.
2. 소프트웨어에는 크게 (시스템) 소프트웨어와 (응용) 소프트웨어 두 종류가 있다.
3. 컴퓨팅적 사고를 기르기 위해서는 컴퓨터가 제공하는 (문제해결) 도구에 대한 학습이 선행되어야 한다.
4. (전산 언어)는 문제해결의 절차를 코딩하여 프로그램을 작성할 수 있도록 하는 언어며, 프로그램은 최종적으로 컴퓨터가 이해하는 (기계어)로 번역 또는 통역되어 실행된다

2-1 빈칸 채우기

1. 인쇄 가능한 문자들로 구성된 텍스트 집단을 (문자열)이라고 하며 특수 문자를 포함한 경우 (탈출문자열 or escape sequence)이라고 한다.
2. 변수를 위한 저장 공간은 프로그램 수행 초기에 기억장치의 일부 공간에 할당되며 프로그램 (종료)와 함께 해제된다. 변수에 특정 값을 저장하는 "**변수 = 값**" 형식의 명령은 (치환문)이라고 부른다.
3. 변수의 이름은 식별자를 사용하며 식별자는 알파벳으로 시작하여 (알파벳), (숫자), 또는 (언더스코어)만을 포함할수 있다.
4. (프롬프트) 입력이란 입력문 사용시 사용자에게 안내 메시지를 인쇄함을 말한다. 이것의 사용 목적은 사용자에게 입력의 내용이나 형식을 안내하고자 함이다.
5. print 명령을 사용한 인쇄문에 변수가 제시되면 해당 변수의 (값)이 인쇄되며 수식이 있을 경우 해당 수식의 (평가값)이 인쇄된다.
6. 주석기호로는 (파운드 or '#')를 사용하며 프로그램 텍스트 라인에서 이 기호 이후 부분은 프로그램으로 취급되지 않는다.

3-1 빈칸 채우기

1. 연산은 컴퓨터의 (중앙처리장치 or CPU)에서 담당하며 여기에는 산술, 관계 및 논리 연산이 있다.
2. 산술 연산의 결과는 수며, 관계나 논리 연산의 결과는 (논리값)이다.
3. 관계 연산의 우선 순위는 산술 연산보다 (~~높~~ , 낮)으며 논리 연산보다는 (높 , ~~낮~~)다.
4. 산술식 "5 – 14 / 5 * (3 / 2.0)"의 평가값은 (2.0)이다.

5. 관계식 "15 % 5 ≤ 33 / 11 – 3"의 평가값은 (True)다.
6. 논리식 "2 > 5 or (5 % 2) != 1"의 평가값은 (False)다.
7. 논리식 "False or True and (False or True)"의 평가값은 (True)다.

4-1 빈칸 채우기

1. 분기 명령에서 조건은 (관계)식 또는 (논리)식으로 표기되며 이를 분기에 사용된 조건식이라고 말한다.
2. if 문의 조건식이 Ture 로 평가되면 키워드 if 아래에 (들여쓰기 , ~~내어쓰기~~)된 내부 명령들을 수행하며 False 로 평가되면 이 명령들을 수행하지 않고 다음 명령으로 진행한다.
3. if 문의 조건식이 (~~True~~ , False)로 평가되더라도 elif 절의 조건식이 (True , ~~False~~)로 평가되면 elif 절의 내부 명령들을 수행한다.
4. (else)절의 사용은 선택적이며 만약 사용될 경우 위의 if 문과 elif 절의 조건식이 모두 (~~True~~ , False)로 평가된 경우에 이 절의 내부 명령들이 수행된다.

5-1 빈칸 채우기

1. 정의된 함수는 (호출)함으로써 비로소 사용된다.
2. 어떤 함수의 수행을 마치면 그 함수를 (호출)한 위치로 돌아와 계속 수행한다.
3. 인자는 함수가 작업을 수행하기 위해 필요한 데이터를 전달하는데 사용되며 함수 정의에서 인자로는 (변수)만 사용할 수 있다.
4. 함수 호출시 인자로 변수를 전달하면 함수에게는 그 변수의 (값)이, 수식을 전달하면 함수에게는 그 수식의 (평가값)이 전달된다.
5. 함수가 작업을 마치고 작업의 결과를 돌려주는 것을 (반환)한다고 말하고 키워드 (return)을 사용한다.
6. 전산언어 시스템에 이미 정의되어 있어 사용자가 별도로 정의하지 않고도 사용할 수 있는 함수를 (내장 함수)라고 부른다.

6-1 빈칸 채우기

1. 여러 개의 데이터 또는 값의 묶음을 목록이라 하며 목록 내 데이터들의 순서는 의미가 (있다 , ~~없다~~).
2. 목록은 프로그램 내에서 초기값을 (변수 or 목록 변수)에 저장하여 만들 수도 있고 입력으로부터 만들 수도 있다.
3. 목록 내의 개별 원소는 "**목록 변수[색인 번호]**"로 표기하며 목록의 첫 원소에 대한 색인 번호는 (0)이다.
4. 목록 원소를 색인 번호로 접근하는 방식은 (인덱싱 or indexing)과 (슬라이싱 or slicing) 두 가지가 있다.
5. 두 목록을 통합하는 산술 연산은 (덧셈 or '+')이다.
6. 함수가 **여러 개**의 값을 반환할 필요가 있을 경우 이 값들을 (목록)으로 만들어 반환하면 된다.

7-1 빈칸 채우기

1. 반복의 두 가지 유형에는 일정 (횟수) 반복과 일정 (조건) 반복이 있다. 첫번째 유형은 for 문을 사용해서 해결한다.
2. 빈 목록에 대해 for 문을 사용하면 (0)회 반복 수행하게 된다.
3. 내장 함수 (range)를 사용하여 연속적인 정수들의 목록을 쉽게 만들 수 있다.
4. (중첩 for 문)은 두 개의 for 문이 외부와 내부에 겹쳐진 형태로 작성된 것을 말한다.
5. for 문 내부에서 for 문의 (반복제어) 변수를 변경하려는 시도는 일반적으로 금기사항이다.

8-1 빈칸 채우기

1. while 문은 조건식이 (True , ~~False~~)인 동안 반복한다.
2. while 문의 반복제어 변수는 세 가지 특징을 가진다. 첫째 반복문 진입 전에 (초기화)된다는 점, 둘째 반복문 내에서 값이 (갱신 or 변경)된다는 점, 마지막으로 반복 조건에서 검사의 대상이 된다는 점이다.
3. while 문의 조건이 항상 (~~True~~ , False)면 while 문의 명령들이 단 한번도 수행되지 않는다. 반대로 항상 (True , ~~False~~)면 while 문은 (무한 루프 or infinite loop)에 진입하게 된다.

4. 반복의 횟수를 미리 알 수 있을 때는 (for) 문을, 알 수 없을 때는 (while) 문을 사용하는 것이 일반적이다.
5. 목록의 원소들을 차례로 처리해야 하는 경우에는 (for) 문이 유리하다. 하지만 중간에 조건에 따라 반복을 중지해야 하는 경우라면 (while) 문이 유리하다.
6. 처음부터 처리할 목록이 존재하지 않는 반복에서는 (while) 문이 적당하다. 하지만 이 경우에도 일정 횟수 만큼 반복해야 한다면 (for) 문이 유리하다.

9-1 빈칸 채우기

1. 스스로를 호출하도록 정의된 함수를 (재귀 함수)라고 부른다.
2. 재귀를 잘 설계하려면 재귀를 종료하는 부분인 (재귀종점부)와 재귀를 진행하는 부분인 (재귀호출부)가 제대로 작성되어야 한다.
3. 재귀 호출이 여러 번 진행되고 마지막에 반환이 시작되면 호출했던 방향의 (역 or 반대)방향으로 진행된다.
4. 재귀 호출시에는 원래 문제보다 크기 면에서 (~~귀찮~~ , 작아진) 문제로 호출하는 것이 반드시 필요하다.
5. 재귀는 (반복)을 대체하고 (변수) 사용을 줄여 프로그램을 간결하게 하고 가독성을 높인다. 특히 난해한 문제해결에 도움이 될 때가 많다.
6. 어떤 함수가 스스로를 두 번 호출하도록 정의되어 있으면 (이중 재귀 or double recursion)라고 한다.
7. 두 함수가 서로 호출하도록 정의되어 있으면 이를 (상호 재귀 or mutual recursion)라고 한다.