#####20191211

Python IDLE사용

3장. 프로그램 구조, 제어문

I. If 문

II. while 문

III. for 문

## I. if문

“돈이 있으면 택시를 타고, 돈이 없으면 걸어 간다.”

주어진 조건을 판단하여 해당 조건에 맞는 상황을 수행하는데 if문이 쓰임

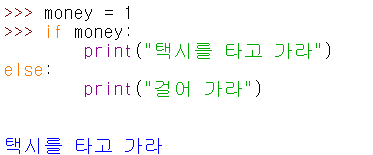
>>> money = 1

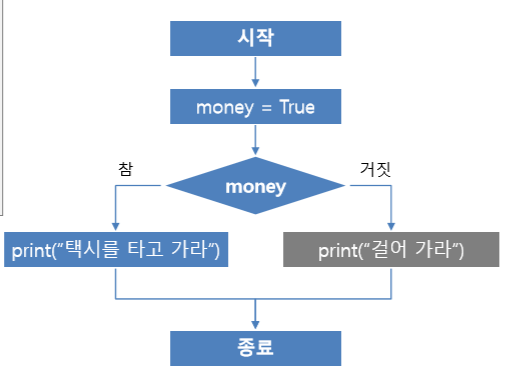
>>> if money:

print("택시를 타고 가라")

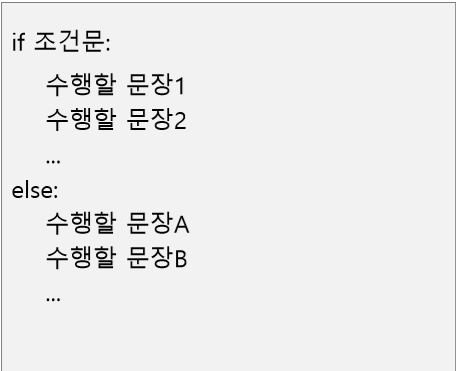
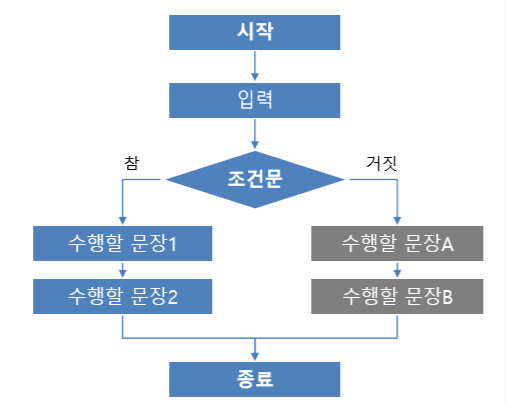
else:

print("걸어 가라")





=>if문의 기본 구조

if 조건문 뒤에는 반드시 콜론(:)이 붙임 

조건문을 테스트해서 참이면 if문 바로 다음의 문장(if 블록)들을 수행하고, 조건문이 거짓이면 else문 다음 문장(else 블록)들을 수행하게 됨 

블록은 들여쓰기로 해결함 (다른 언어는 { }로 기호 사용) 

else문은 if문 없이 독립적으로 사용할 수 없음

=>들여쓰기

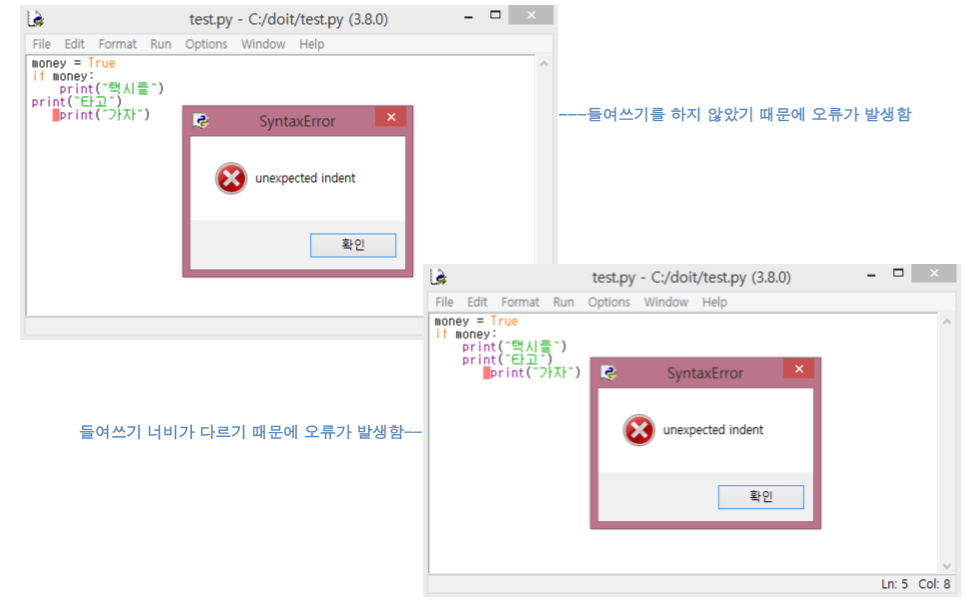
if 조건문:

수행할 문장1

수행할 문장2

수행할 문장3

들여쓰기는 공백(Spacebar) 4개나 탭(Tab)을 사용하며, 2가지를 혼용하지 말 것

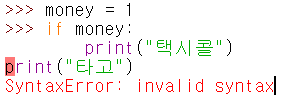


>>> money = 1

>>> if money:

print("택시콜")

print("타고")



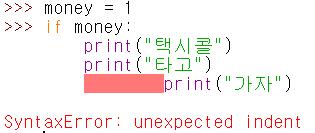
>>> money = 1

>>> if money:

print("택시콜")

print("타고")

print("가자")



=>조건문이란 무엇인가?

if 조건문에서 ‘조건문’이란 참과 거짓을 판단하는 문장을 말함

>>> money = 1

>>> if money:



money는 1이기 때문에 참이 되어 if문 다음의 문장을 수행함

=>비교연산자

조건문에 비교연산자(<, >, ==, !=, >=, <=)를 쓰는 경우가 많음



>>> x= 3

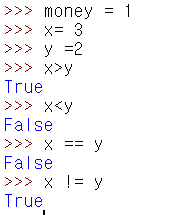
>>> y =2

>>> x>y

>>> x<y

>>> x == y

>>> x != y



만약 3000원 이상의 돈을 가지고 있으면 택시를 타고 그렇지 않으면 걸어 가라.

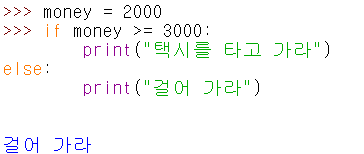
>>> money = 2000

>>> if money >= 3000:

print("택시를 타고 가라")

else:

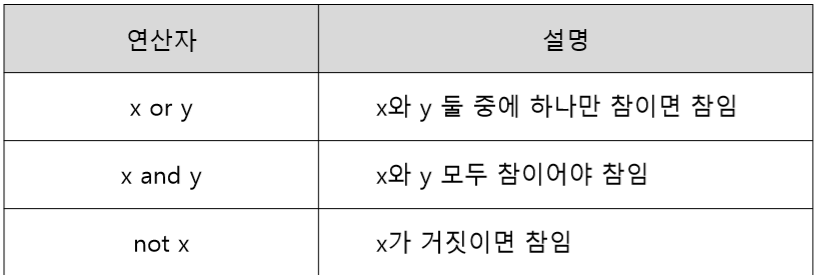
print("걸어 가라")



=> and, or, not

조건을 판단하기 위해 사용하는 연산자로는 and, or, not이 있

and, or, not 조건을 판단하기 위해 사용하는 연산자로는 and, or, not이 있음->논리 연산자



==> or 연산자의 사용법을 알아봄

돈이 3000원 이상 있거나 카드가 있으면 택시를 타고 그렇지 않으면 걸어 가라.

>>> money = 2000

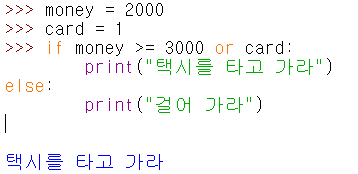
>>> card = 1

>>> if money >= 3000 or card:

print("택시를 타고 가라")

else:

print("걸어 가라")



>>> money = 2000

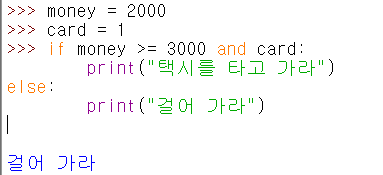
>>> card = 1

>>> if money >= 3000 and card:

print("택시를 타고 가라")

else:

print("걸어 가라")



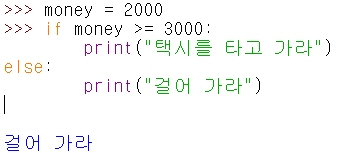
>>> money = 2000

>>> if money >= 3000:

print("택시를 타고 가라")

else:

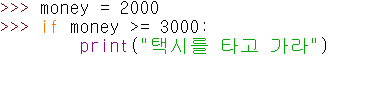
print("걸어 가라")



money = 2000

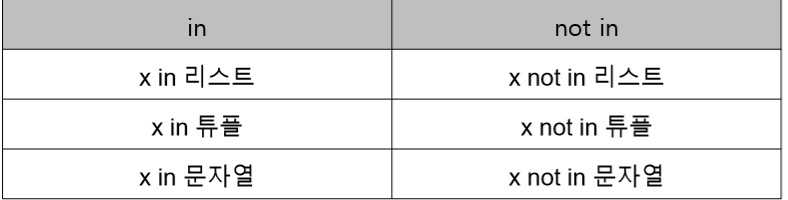
if money >= 3000:

print("택시를 타고 가라")



=>x in s, x not in s

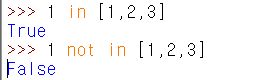
다른 프로그래밍 언어에서 볼 수 없는 조건문들을 제공함



in의 뜻이 ‘~안에‘라는 것을 생각하면 쉽게 이해될 것임

>>> 1 in [1,2,3]

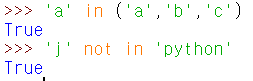
>>> 1 not in [1,2,3]



튜플과 문자열에 적용한 예

>>> 'a' in ('a','b','c')

>>> 'j' not in 'python'



=>택시 예제에 in을 적용

만약 주머니에 돈이 있으면 택시를 타고, 없으면 걸어가라.

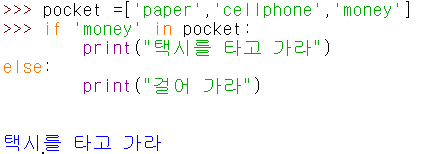
>>> pocket =['paper','cellphone','money']

>>> if 'money' in pocket:

print("택시를 타고 가라")

else:

print("걸어 가라")



=>조건문에서 아무 일도 하지 않게 설정하고 싶다면?

주머니에 돈이 있으면 가만히 있고 주머니에 돈이 없으면 카드를 꺼내라.

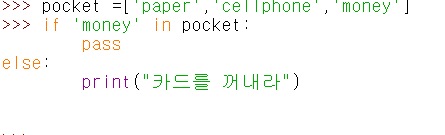
>>> pocket =['paper','cellphone','money']

>>> if 'money' in pocket:

pass

else:

print("카드를 꺼내라")



pocket이라는 리스트 안에 money라는 문자열이 있기 때문에 if문 다음 문장 인 pass가 수행되고 아무런 결과값도 보여 주지 않음

#125페이지

#주머니에 카드가 없다면 걸어가로 , 있다면 버스 타라

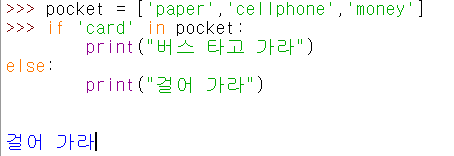
>>> pocket = ['paper','cellphone','money']

>>> if 'card' in pocket:

print("버스 타고 가라")

else:

print("걸어 가라")



=> 다양한 조건을 판단하는 elif

주머니에 돈이 있으면 택시를 타고, 주머니에 돈이 없지만 카드가 있으면 택시를 타고, 돈도 없고 카드도 없으면 걸어 가라.

if와 else만으로 표현

>>> pocket = ['paper', 'cellphone']

>>> card = 1

>>> if 'money' in pocket:

print("택시를 타고 가라")

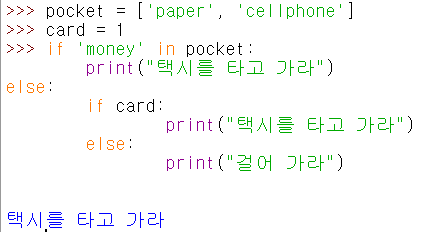
else:

if card:

print("택시를 타고 가라")

else:

print("걸어 가라")



복잡함을 해결하기 위해 다중 조건 판단을 가능하게 하는 elif 사용

>>> pocket = ['paper', 'cellphone']

>>> card = True #혹은 1

>>> if 'money' in pocket:

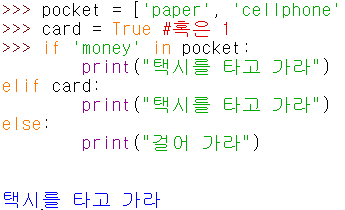
print("택시를 타고 가라")

elif card:

print("택시를 타고 가라")

else:

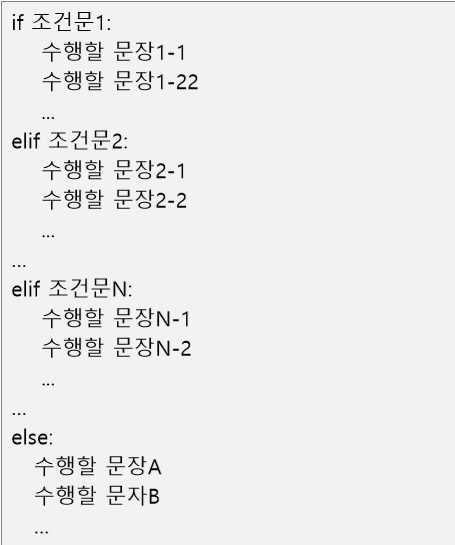
print("걸어 가라")



elif는 이전 조건문이 거짓일 때 수행됨

elif는 개수에 제한 없이 사용할 수 있음

=> if, elif, else를 모두 사용할 때의 기본 구조



=> if문을 한 줄로 작성하기

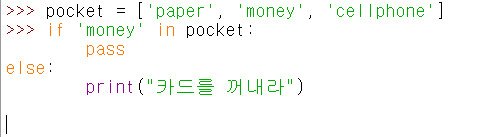
>>> pocket = ['paper', 'money', 'cellphone']

>>> if 'money' in pocket:

pass

else:

print("카드를 꺼내라")

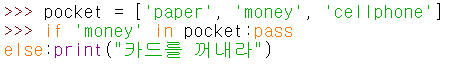


수행할 문장이 한 줄일 때 조금 더 간략하게 코드를 작성하는 방법이 있음

>>> pocket = ['paper', 'money', 'cellphone']

>>> if 'money' in pocket:pass

else:print("카드를 꺼내라")



if문 다음 수행할 문장을 콜론(:) 뒤에 적어 주었고, else문도 마찬가지임

=> 조건부 표현식

>>> score = 70

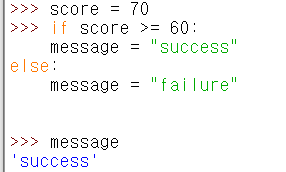
>>> if score >= 60:

message = "success"

else:

message = "failure"

>>> message

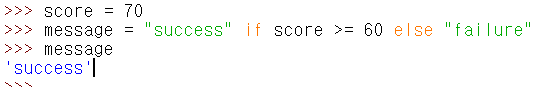


조건부 표현식(conditional expression)을 사용하면 간단히 표현할 수 있음

>>> score = 70

>>> message = "success" if score >= 60 else "failure"

>>> message



조건부 표현식 정의

조건문이 참인 경우 if 조건문 else 조건문이 거짓인 경우

가독성에 유리하고 한 줄로 작성할 수 있어 활용성이 좋음

=>응용예제01. 학점 세분화 프로그램"

점수를 입력받은 후 90점 이상은 A, 80점 이상은 B, 70점 이상은 C, 60점 이상은 D,나머지는 F 로 처리하는 프로그램을 구현해 보자

score = int(input("점수를 입력하세요:"))

if score >= 90:

grade = "A"

elif score >= 80:

grade = "B"

elif score >= 70:

score = "C"

elif score >= 60:

grade = "D"

else:

grade = "F"

print(f'{score}는 {grade}학점입니다.')

=>while로 변경하기 ->계속 이력 가능

while True:

score = int(input("점수를 입력하세요:"))

if score >= 90:

grade = "A"

elif score >= 80:

grade = "B"

elif score >= 70:

score = "C"

elif score >= 60:

grade = "D"

else:

grade = "F"

if score == 0:

break

print(f'{score}는 {grade}학점입니다.')

=>while로 변경하되 주석 print방식을 3가지로 확인하기

while True:

score = int(input("점수를 입력하세요:"))

if score >= 90:

grade = "A"

elif score >= 80:

grade = "B"

elif score >= 70:

score = "C"

elif score >= 60:

grade = "D"

else:

grade = "F"

if score == 0:

break

print("%d점은 %c학점입니다." % (score,grade))

print(f'{score}는 {grade}학점입니다.')

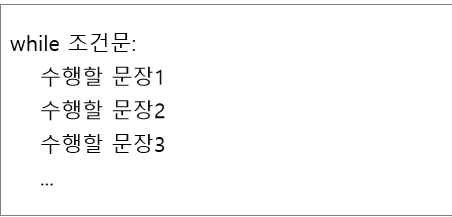
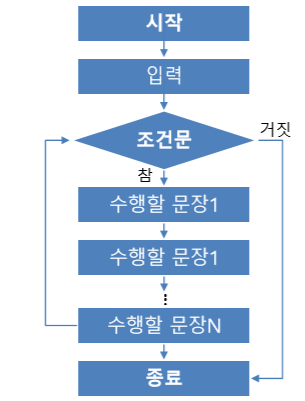
print("{0}점은 {1}학점입니다.".format(score,grade))

## II. while문

while문의 기본 구조

반복해서 문장을 수행해야 할 경우 while문을 사용함

while문의 기본구조

while문은 조건문이 참인 동안에 while문 아래에 속하는 문장들이 반복해서 수행됨

예: ‘열 번 찍어 안 넘어가는 나무 없다‘는 속담을 while문으로 만든 예

>>> treeHit = 0

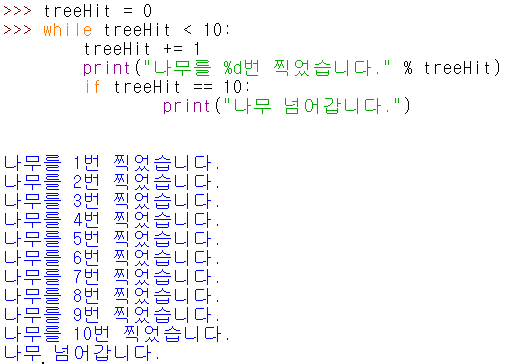
>>> while treeHit < 10:

treeHit += 1

print("나무를 %d번 찍었습니다." % treeHit)

if treeHit == 10:

print("나무 넘어갑니다.")



=> while문이 반복되는 과정을 순서대로 정리한 표



예: treeHit 가 1일 경우 10번까지 찍기

방안 1:

>>> treeHit = 1

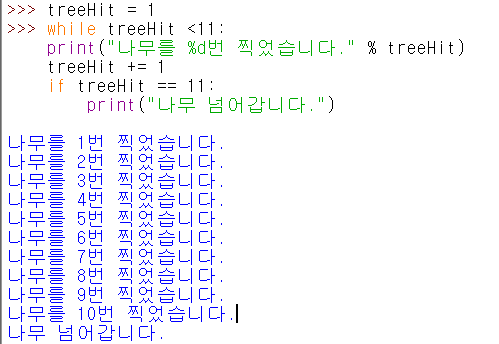
>>> while treeHit <11:

print("나무를 %d번 찍었습니다." % treeHit)

treeHit += 1

if treeHit == 11:

print("나무 넘어갑니다.")



방안 2:

>>> treeHit = 1

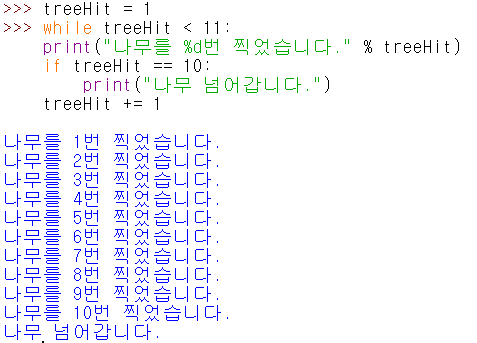
>>> while treeHit < 11:

print("나무를 %d번 찍었습니다." % treeHit)

if treeHit == 10:

print("나무 넘어갑니다.")

treeHit += 1



방안 2가 더 좋다. 마지막에 값을 추가하는 게 메모리 측면에서 더 좋다.

=>while문 만들기 

여러 가지 선택지 중 하나를 선택해서 입력 받는 예제 

여러 줄의 문자열을 변수에 대입하기 위해 큰따옴표 3개(“””)를 이용함

>>> prompt = """

1. Add

2. Del

3. 3.List

4. Quit

Enter number: """

>>> number = 0

>>> while number != 4:

print(prompt)

number = int(input())

# 결과 화면처럼 사용자가 4라는 값을 입력하지 않으면 계속해서 prompt를 출력함

1. Add

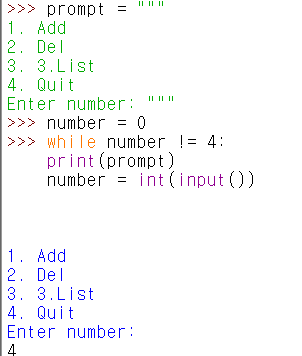
2. Del

3. 3.List

4. Quit

Enter number:

4



=> while문 강제로 빠져나가기 (break)

while문은 조건문이 참인 동안 계속해서 while문 안의 내용을 반복적으로 수행하지만 강제로 while문을 빠져나가고 싶을 때 사용하는 것이 break문임

커피 자판기 이야기를 break문으로 만든 예:

>>> coffee = 10

>>> money = 300

>>> while money:

print("돈을 받았으니 커피를 줍니다.")

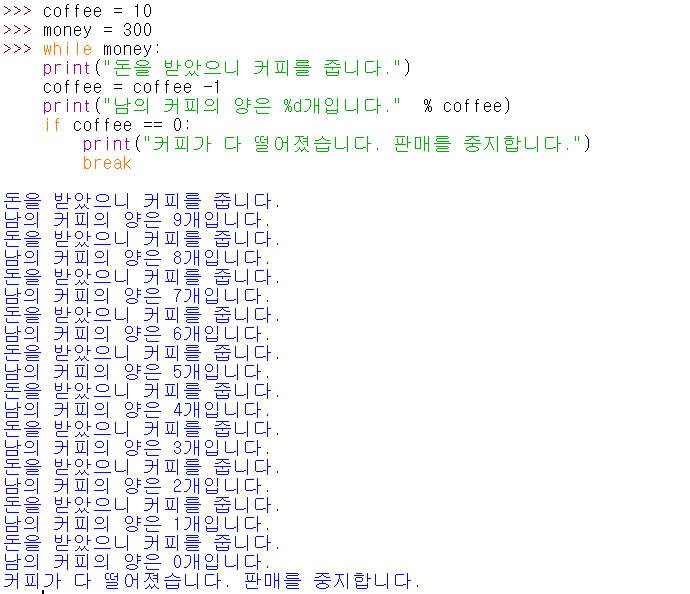
coffee = coffee -1

print("남의 커피의 양은 %d개입니다." % coffee)

if coffee == 0:

print("커피가 다 떨어졌습니다. 판매를 중지합니다.")

break



=>실제 자판기 작동 과정을 만든 예

coffee = 10

while True:

money = int(input("돈을 넣어 주세요:"))

if money == 300:

print("커피를 줍니다.")

coffee -= 1

elif money > 300:

print("거스름돈 %d를 주고 커피를 줍니다." % (money - 300))

coffee -= 1

else:

print("돈을 다시 돌려주고 커피를 주지 않습니다.")

print("남은 커피의 양은 %d개입니다." % coffee)

if coffee == 0:

print("커피가 다 떨어졌습니다. 판매를 중지합니다.")

break

=>개선 버전

# PPT\_3.2\_while\_coffee.py

coffee = 10

while True:

#money,cnt = input("돈과 개수를 넣어 주세요:").split()

money = int(input("돈을 넣어 주세요:"))

if money == 300:

print("커피를 줍니다.")

coffee -= 1

elif money > 300:

#print(money%300)

#print(money//300)

maxCnt = money//300

if maxCnt > coffee:

maxCnt = coffee

if maxCnt > 1:

cnt = int(input(f'%d잔 이하 가능합니다.몇잔 하시겠습니까?' % maxCnt))

else:

cnt = maxCnt

print("거스롬돈 %d를 주고 커피를 줍니다." % (money - 300\*cnt ))

coffee -= cnt

else:

print("돈을 다시 돌려주고 커피를 주지 않습니다.")

print("남은 커피의 양은 %d개입니다." % coffee)

if coffee == 0:

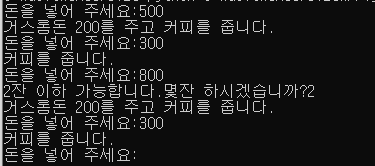
print("커피가 다 떨어졌습니다. 판매를 중지합니다.")

break

PPT\_3.2\_while\_coffee.py을 저장한 후 프로그램을 직접 실행 함



입력란에 여러 숫자를 입력해 보면서 결과를 확인함



#####20191212

논리연산자

dict

if문 -> message = "success" if score >= 60 else "failure"

message = "A" if score >= 60 "B" elif score >= 50 else "failure"#SyntaxError: invalid syntax

#실행문이 한줄일 경우 조건문 표현식 오류

while

조건문이 안 맞을때 빠져나갈수 있지만

break도 가능하다.

continue는 만나면 처음으로 시작한다. 다음의 것 실행안한다.

while 1->참이기 때문에

=>while문의 맨 처음으로 돌아가기 (continue)

while문을 빠져나가지 않고, while문의 맨 처음(조건문)으로 다시 돌아가게 만들고 싶을 때 사용하는 것이 continue문임

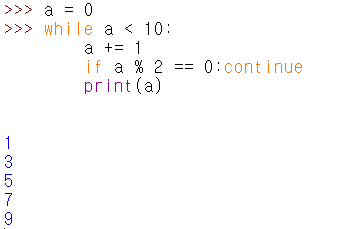
>>> a = 0

>>> while a < 10:

a += 1

if a % 2 == 0:continue

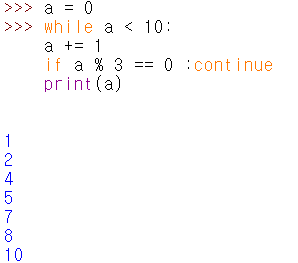
print(a)



a가 짝수이면 continue 문장을 수행하고, while문의 맨 처음으로 돌아감

교재 136페이지

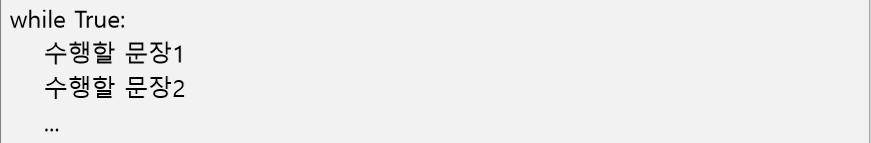
1부터 10까지의 숫자 중에서 3의 배수를 뺀 나머지 값을 출력해 보자



=>무한 루프

무한 루프(Loop)란 무한히 반복한다는 의미임 

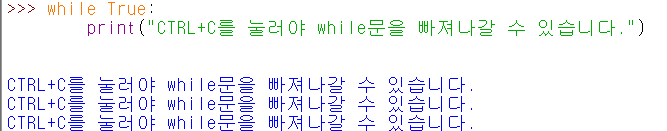
while문의 조건문이 True이면 항상 참이므로, while문 안에 있는 문장들은 무한하게 수행됨



무한 루프의 예:

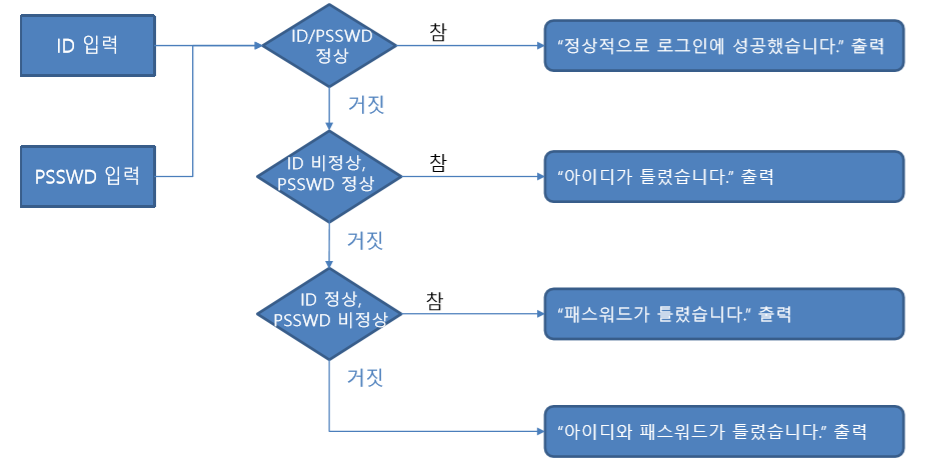
>>> while True:

print("CTRL+C를 눌러야 while문을 빠져나갈 수 있습니다.")



응용예제01. 로그인 프로그램:

아이디와 패스워드를 입력하였을 때, 등록된 정보와 비교하여 로그인을 승인하는 프로그램을 구현해 보자.



응용예제:

myId = "python"

myPsswd = "1234"

count = 0

while True:

#usrId = input("아이디를 입력하세요.")

#usrPwd = input("패스워드를 입력하세요.")

if count >= 5 :

print("시도 횟수 초과하였습니다.")

break

userId, userPwd = input("아이디와 패스워드를 ,로 구분하여 입력하세요").split()

if userId == myId and myPsswd == userPwd:

print("정상적으로 로그인에 성공했습니다.")

break

elif userId != myId and myPsswd == userPwd:

print("아이디가 틀렸습니다.\n")

elif userId == myId and myPsswd != userPwd:

print("패스워드가 틀렸습니다.\n")

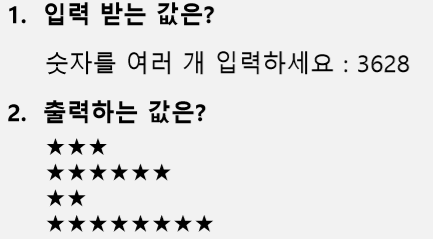
else:

print("아이디와 패스워드가 틀렸습니다.\n")

count+=1 #최대한 공간을 줄이기 위해서 마지막에 연다.

2. 별 모양 출력하기

사용자가 숫자를 여러 개 입력하면 별 모양(‘\u2605)을 입력한 숫자만큼 출력하는 프로그램이다.



실습

i = 0

while True:

i += 1

if i>5:break

print("\u2605"\*i+"\n")

#for문

#1.

input1 = input("1.숫자를 여러 개 입력하세요 :")

for i in input1:

print("\u2605"\*int(i))

print("="\*50)

#while문

#2.

cnt = 0

input1 = input("2.숫자를 여러 개 입력하세요 :")

while cnt < len(input1):

print("\u2605"\*int(input1[cnt]))

cnt+=1

print("="\*50)

#3.

cnt = 0

input1 = input("3.숫자를 여러 개 입력하세요 :")

while input1[cnt:]:

print("\u2605"\*int(input1[cnt]))

cnt+=1

print("="\*50)

#4.

cnt = 0

input1 = input("4.숫자를 여러 개 입력하세요 :")

while input1[cnt]:

print("\u2605"\*int(input1[cnt]))

cnt+=1

if len(input1) == cnt:

break

print("="\*50)

#5.

input1 = input("5.숫자를 여러 개 입력하세요 :")

for i in range(len(input1)):

print("\u2605"\*int(input1[i]))

print("="\*50)

#6.

#list내포는 result리스트로 만든다.

input1 = input("6.숫자를 여러 개 입력하세요 :")

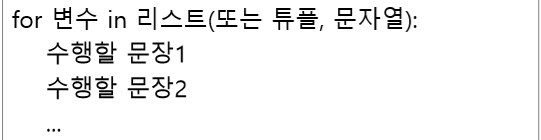
result =["\u2605"\*int(n) for n in input1]

print(result)

## III. for문

for문의 기본 구조

리스트나 튜플, 문자열의 첫 번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 수행할 문장들이 수행됨



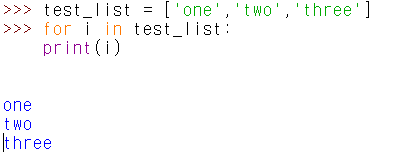
#변수를 i,j,k를 많이 사용

1. 전형적인 for문

>>> test\_list = ['one','two','three'] #list 형 자료형

>>> for i in test\_list:

print(i)

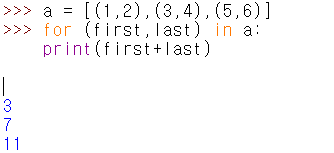


2. 다양한 for문의 사용

>>> a = [(1,2),(3,4),(5,6)]

>>> for (first,last) in a:

print(first+last)

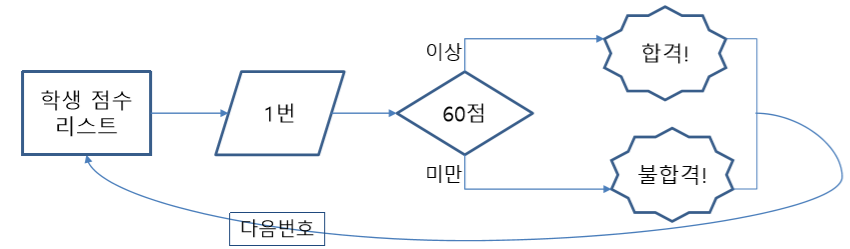


3. for문의 응용

총 5명의 학생이 시험을 보았는데 시험 점수가 60점이 넘으면 합격이고 그렇지 않으면 불합격이다. 합격인지 불합격인지 결과를 보여주시오.

우선 학생 5명의 시험 점수를 리스트로 표현함

>>> marks = [90, 25, 67, 45, 80]



#marks1.py

marks = [90, 25, 67, 45, 80]

number = 0

for mark in marks:

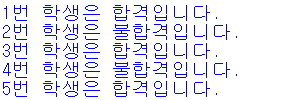
number += 1

if mark >= 60:

print("%d번 학생은 합격입니다." % number)

else:

print("%d번 학생은 불합격입니다." % number)



점수 출력 및 소수점 처리

marks = [95,25,67,45,80]

number = 0

for mark in marks:

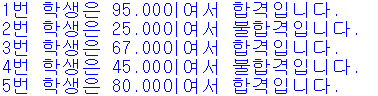
number = number +1

if mark >= 60:

print("%d번 학생은 %.2f이여서 합격입니다." % (number,mark))

else:

print("%d번 학생은 %.2f이여서 불합격입니다." % (number,mark))



=> for문과 continue

for문 안의 문장을 수행하는 도중에 continue문을 만나면 for문의 처음으로 돌아감

#marks2.py

marks = [90, 25, 67, 45, 80]

number = 0

for mark in marks:

number += 1

if mark < 60 : continue

print("%d번 학생 축하합니다. 합격입니다." % number)



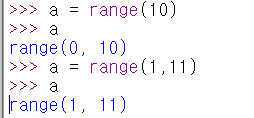
=>for와 함께 자주 사용하는 range함수

>>> a = range(10)

>>> a <--0, 1, 2, 3, 4, 5, 6, 7, 8, 9

>>> a = range(1,11)

>>> a <--1, 2, 3, 4, 5, 6, 7, 8, 9, 10



시작 숫자와 끝 숫자를 지정하려면 range(시작 숫자, 끝 숫자) 형태를 사용하는데, 이때 끝 숫자는 포함되지 않음

==>range 함수의 예시 살펴보기

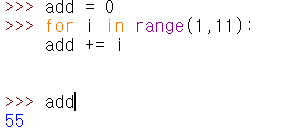
for와 range 함수를 이용하여 1부터 10까지 더하는 것을 구현함

>>> add = 0

>>> for i in range(1,11):

add += i

>>> add



#marks3.py

marks = [90, 25, 67, 45, 80]

for number in range(len(marks)):

if marks[number] < 60:continue

print("%d번 학생 축하합니다. 합격입니다. " % (number + 1))



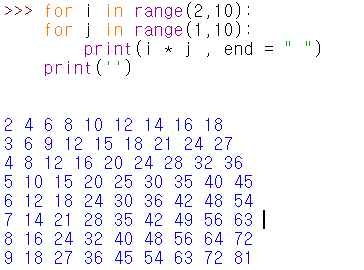
==> for와 range를 사용한 구구단

>>> for i in range(2,10):

for j in range(1,10):

print(i \* j , end = " ")

print('')

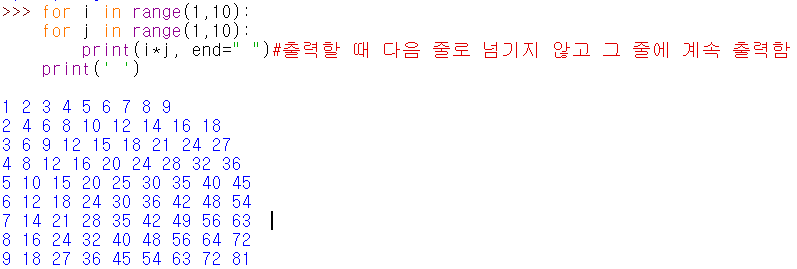


for i in range(1,10):

for j in range(1,10):

print(i\*j, end=" ")#출력할 때 다음 줄로 넘기지 않고 그 줄에 계속 출력함

print(' ')



==>실제 수자로 보고싶다.

>>> a= list(range(len(marks)))

>>> a



=> 리스트 내포 사용하기

리스트 안에 for문을 포함하는 리스트 내포(List comprehension)를 사용하면 편리함

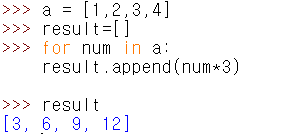
>>> a = [1,2,3,4]

>>> result=[]

>>> for num in a:

result.append(num\*3)

>>> result

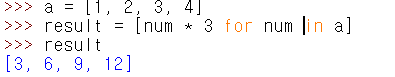


리스트 내포를 사용하면 간단히 해결할 수 있음

>>> a = [1, 2, 3, 4]

>>> result = [num \* 3 for num in a]

>>> result



>>> [num \* 3 for num in a] ->거짓이면 대입자체가 안되서

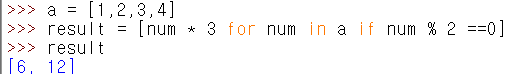


=> 짝수에만 3을 곱하여 담고 싶다면 리스트 내포 안에 ‘if 조건’을 사용함

>>> a = [1,2,3,4]

>>> result = [num \* 3 for num in a if num % 2 ==0]

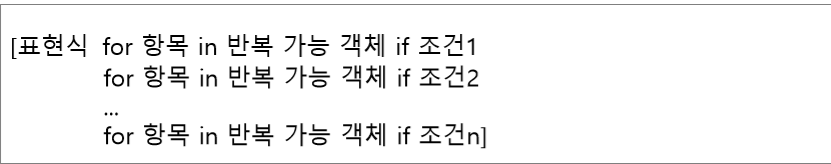
>>> result



\*\*\*리스트 내포의 일반적인 문법

[표현식 for 항목 in 반복 가능 객체 if 조건]

\*\*\*for문을 여러 개 사용할 때의 문법

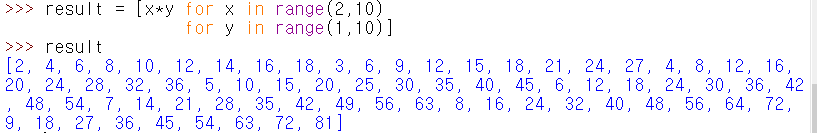


=> 구구단의 모든 결과를 리스트에 담고 싶다면 리스트 내포를 사용하여 구현함

>>> result = [x\*y for x in range(2,10)

for y in range(1,10)]

>>> result

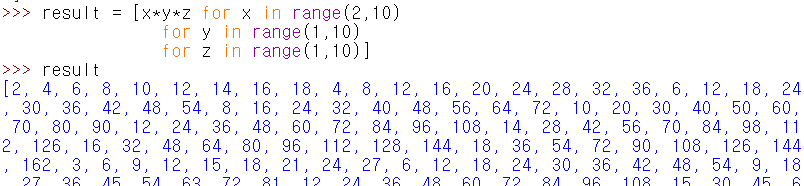


>>> result = [x\*y\*z for x in range(2,10)

for y in range(1,10)

for z in range(1,10)]

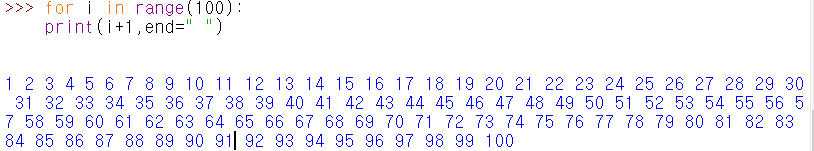
>>> result



==>for문을 사용해 1부터 100까지 출력하기

>>> for i in range(100):

print(i+1,end=" ")



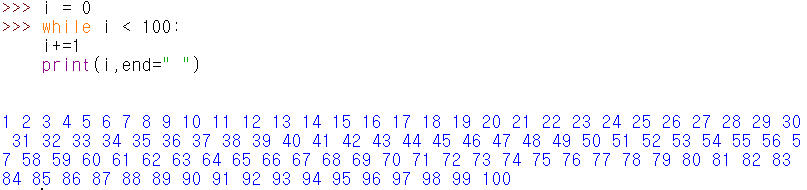
==>while문으로 바꾸기

>>> i = 0

>>> while i < 100:

i+=1

print(i,end=" ")



==>평균점수 구하기

방안 1:

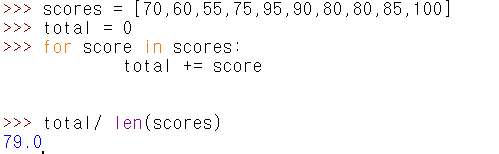
>>> scores = [70,60,55,75,95,90,80,80,85,100]

>>> total = 0

>>> for score in scores:

total += score

>>> total/ len(scores)



방안 2:

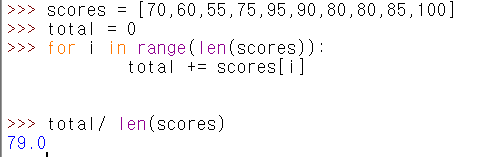
>>> scores = [70,60,55,75,95,90,80,80,85,100]

>>> total = 0

>>> for i in range(len(scores)):

total += scores[i]

>>> total/ len(scores)



>>> numbers = [1,2,3,4,5]

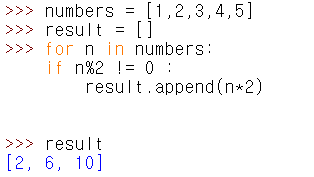
>>> result = []

>>> for n in numbers:

if n%2 != 0 :

result.append(n\*2)

>>> result



>>> numbers = [1,2,3,4,5]

>>> result = []

>>> result = [num\* 2 for num in numbers if num%2 != 0]

>>> result

treeHit=0

for i in range(10):

treeHit = i+1

print("나무를 %d를 찍었습니다" % treeHit)

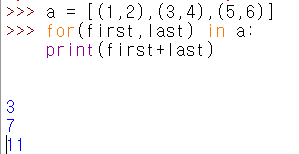
print("나무 넘어갑니다.")



>>> a = [(1,2),(3,4),(5,6)]

>>> for(first,last) in a:

print(first+last)



연습문제 146페이지

1.

>>> a = "Life is too short, you need python"

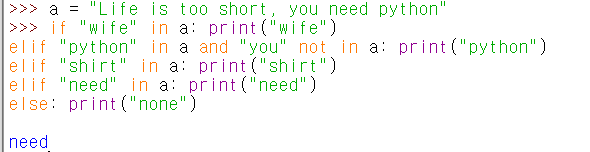
>>> if "wife" in a: print("wife")

elif "python" in a and "you" not in a: print("python")

elif "shirt" in a: print("shirt")

elif "need" in a: print("need")

else: print("none")



2.

>>> result = 0

>>> i = 1

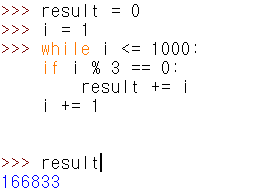
>>> while i <= 1000:

if i % 3 == 0:

result += i

i += 1

>>> result



3.

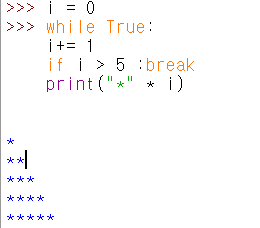
>>> i = 0

>>> while True:

i+= 1

if i > 5 :break

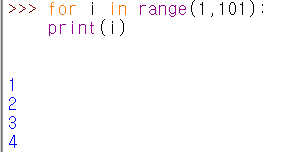
print("\*" \* i)



4.

>>> for i in range(1,101):

print(i)



5.

A= [70,60,55,75,95,90,80,85,85,100]

total = 0

for score in A:

total += score

average = total/len(A)

print(average)



6.

numbers = [1,2,3,4,5]

result = []

for n in numbers:

if n % 2 == 1:

result.append(n\*2)

numbers = [1,2,3,4,5]

result = [n\*2 for n in numbers if n % 2 == 1]

print(result)