AI 이노베이션스퀘어

실습문제

- ('문자열 ' * 변수)를 이용하여 다음의 실행 결과가 나오도록 코드를 작성해보세요.

```
콩 콩 콩 콩 콩 콩 콩 콩 콩 콩 나무를 10번 찍었습니다.
```

- 나이를 입력 받아 출력하는 코드를 작성해보세요.

• 소스코드

```
print ('How old are you?')
age = input()
print ('You are '  + age + ' years old.')
```

• 실행결과

```
How old are you?
15
You are 15 years old.
```

온도 변환 프로그램

- 아래와 같은 온도 변환 프로그램을 작성해보세요.

```
온도 변환 프로그램
화씨 온도를 입력하세요:50
섭씨 온도는 10.0 입니다.

섭씨 온도를 입력하세요:50
화씨 온도는 122.0 입니다.

>>>
```

공식 :
화씨(f) -> 섭씨(c) c= 5.0/9.0*(f-32.0)
섭씨(c) -> 화씨(f)
f=9.0/5.0*c+32.0

- 소스코드

```
print('온도 변환 프로그램.')
f = input('화씨 온도를 입력하세요:')
c = 5.0/9.0*(float(f)-32.0)
print('섭씨 온도는', c, '입니다.' )
print( )

c = input('섭씨 온도를 입력하세요:')
f = 9.0/5.0*float(c)+32.0
print('화씨 온도는', f, '입니다.' )
```

BMI 계산

- 체중과 신장을 입력 받아 BMI(신체질량지수)를 계산하여 출력하는 프로그램을 작성하시오.
  - BMI 계산식 : 체중(kg) / (신장(cm)/100)^2

- 실행결과

```
체중을 입력하세요(kg): 48.4
신장을 입력하세요(cm): 178.7

BMI는 15.16 입니다.
```

- 소스코드

```
weight  = input('체중을 입력하세요(kg):')
height  = input('신장을 입력하세요(cm):')
print( )
bmi = float(weight) / (float(height)/100)**2
print ('BMI는 %.2f'%BMI, '입니다')
```

- Tip : 소수점 두자리까지만 출력하는 방법

```
print ('BMI는 %.2f'%BMI, '입니다')
```

변수 BMI의 값에서 소수점 두 자리 까지만 출력하라는 뜻

- 출력결과는 항상 int형

```
In [ ] : a = 0b010011
In [ ] : a
Out[ ] :  19
```

```
In [ ] : type(a)
Out[ ] :  int
```

```
In [ ] : b = bin(19)
In [ ] : b
Out[ ] : '0b10011'
```

```
In [ ] : type(b)
Out[ ] :  str
```

- 출력결과는 항상 int형

```
In [ ] : a = 0o724
In [ ] : a
Out[ ] :  468
```

```
In [ ] : type(a)
Out[ ] :  int
```

```
In [ ] : b = oct(468)
In [ ] : b
Out[ ] : '0o724'
```

```
In [ ] : type(b)
Out[ ] :  str
```

```
In [ ] : a = 0x19AF
In [ ] : a
Out[ ] :  6575
```

```
In [ ] : type(a)
Out[ ] :  int
```

```
In [ ] : b = hex(6575)
In [ ] : b
Out[ ] : '0o19af'
```

```
In [ ] : type(b)
Out[ ] :  str
```

논리(Boolean)형


```
In [ ] : a = True
In [ ] : a
Out[ ] :
```

```
In [ ] : type(a)
Out[ ] : bool
```

```
In [ ] : a = bool(True)
In [ ] : a
Out[ ] : True
```

```
In [ ] : type(a)
Out[ ] : bool
```

```
In [ ] : b = False
In [ ] : b
Out[ ] : False
```

```
In [ ] : type(b)
Out[ ] : bool
```

```
In [ ] : b = bool(False)
In [ ] : b
Out[ ] : False
```

```
In [ ] : type(b)
Out[ ] : bool
```

논리(Boolean)형


```
In [ ] : a+1
Out[ ] : 2
```

```
In [ ] : b+1
Out[ ] : 1
```

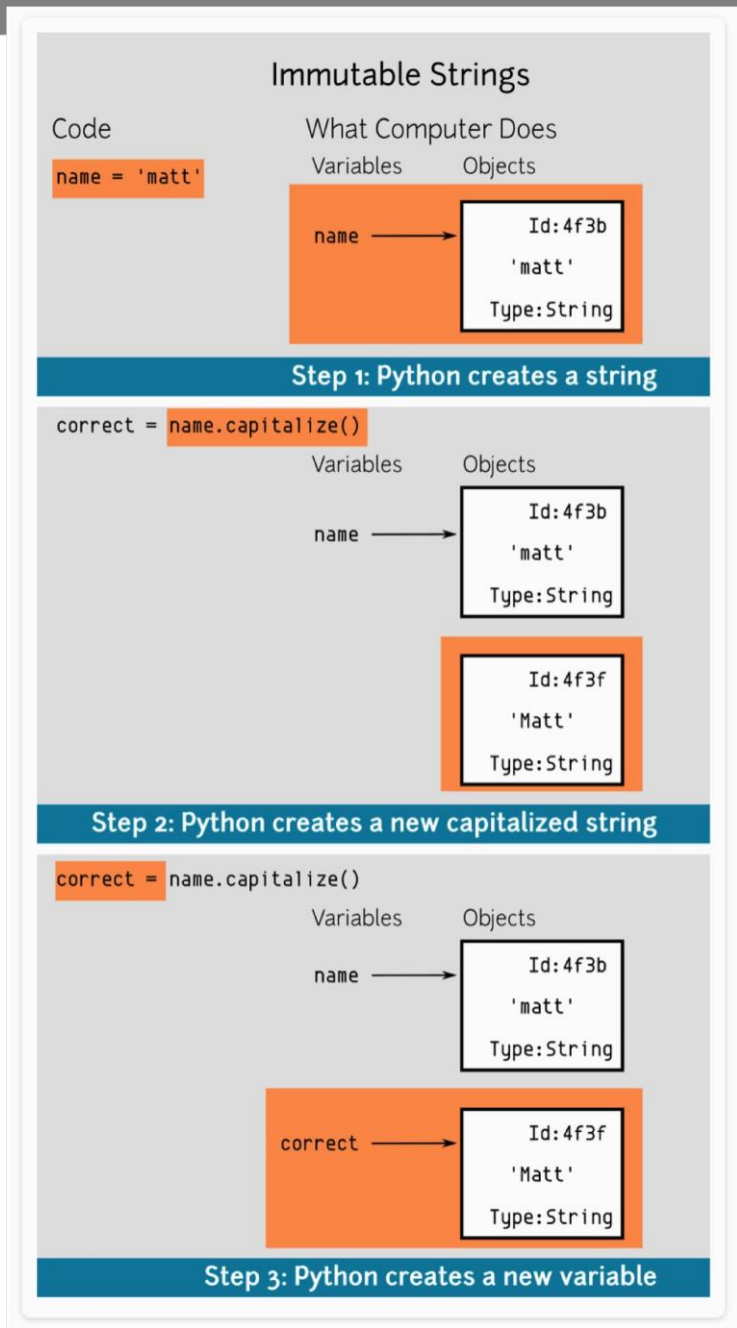
```
In [ ] : a==1
Out[ ] : True
```

```
In [ ] : b==0
Out[ ] : True
```

immutable, flat sequence

2. String

string


Immutable Strings
Code
What Computer Does
name = 'matt'
Variables
Objects
name
Id:4f3b
'matt'
Type:String
Step 1: Python creates a string
correct = name.capitalize()
Variables
Objects
name
Id:4f3b
'matt'
Type:String
Id:4f3f
'Matt'
Type:String
Step 2: Python creates a new capitalized string
correct = name.capitalize()
Variables
Objects
name
Id:4f3b
'matt'
Type:String
correct
Id:4f3f
'Matt'
Type:String
Step 3: Python creates a new variable

string

■ str

ㅇ 글자, 글자가 모여서 만드는 단어, 문장, 여러 줄의 단락이나 글 전체

■ literal

○문자열은 홑따옴표, 쌍따옴표, doc스트링(''' ''')으로 표현

ㅇ 큰 따옴표와 작은 따옴표를 구분하지 않지만 양쪽 따옴표가 맞아야 함. (문자열을 둘러싸는 따옴표와 다른 따옴표는 문자열 내의 일반 글자로 해석)

➢ "apple", 'apple'은 모두 문자열 리터럴로 apple이라는 단어를 표현

ㅇ 두 개의 문자열 리터럴이 공백이나 줄바꿈으로 분리되어 있는 경우에 이것은 하나의 문자열 리터럴로 해석

➢ "apple," "banana"는 "apple,banana"라고 쓴 표현과 동일

- 따옴표 세 개를 연이어서 쓰는 경우에는 문자열 내에서 줄바꿈이 허용.
  - 흔히 함수나 모듈의 간단한 문서화 텍스트를 표현할 때 쓰임.

  - """He said "I didn't go to 'SCHOOL' yesterday"."""
    - He said "I didn't go to 'SCHOOL' yesterday".

  - 여러 줄에 대한 내용을 쓸 때.
  - '''HOMEWORK:

1. print "hello, world"
2. print even number between 2 and 12
3. calculate sum of prime numbers up to 100,000 ”

string

```
In [10]: '안녕하세요'
Out[10]: '안녕하세요'

In [11]: "안녕하세요"
Out[11]: '안녕하세요'

In [12]: '''안녕하세요'''
Out[12]: '안녕하세요'
```

string

1

```
In [ ] : a="hello"
In [ ] : a
Out[ ] :
```

```
In [ ] : type(a)
Out[ ] :
```

2

```
In [ ] : b="안녕하세요"
In [ ] : b
Out[ ] :
```

```
In [ ] : type(b)
```

3

```
In [ ] : d="hello₩nworld!"
In [ ] : d
Out[ ] :
```

```
In [ ] : print(d)
         hello
         world!
```

4

```
In [ ] : e = r"hello₩nworld!"
In [ ] : e
Out[ ] : 'hello₩₩nworld!'
In [ ] : print(e)
         hello₩nworld!
```

Assignment(할당)

- ' 기호를 출력하려면 " " 로 문자열 처리
- " 기호를 출력하면 ' ' 로 문자열 처리

```
In [ ] : 'Hello world!'
In [ ] : "who is alice's best friend?"
In [ ] : 'Bob said "carry" '
```

string

- 앞에 f를 붙여 사용하면 파이썬 코드의 실행결과를 문자열로 대입할 수 있음

```
In [1]: b={1==2}

In [2]: b
Out[2]: {False}

In [3]: type(b)
Out[3]: set

In [4]: b=1==2

In [5]: b
Out[5]: False
```

```
In [6]: type(b)
Out[6]: bool

In [7]: b=f'{1==2}'

In [8]: b
Out[8]: 'False'

In [9]: type(b)
Out[9]: str
```

string

- 변수로 한글사용 가능

```
In [13]: 일= 1

In [14]: 일
Out[14]: 1

In [15]: 이름='변해선'

In [16]: 이름
Out[16]: '변해선'

In [17]: type(이름)
Out[17]: str
```

Escape Sequence	Output
\newline	Ignore trailing newline in triple quoted string
\\	Backslash
\'	Single quote
\"	Double quote
\b	ASCII Backspace
\n	Newline
\r	ASCII carriage return
\t	Tab
\u12af	Unicode 16 bit
\U12af89bc	Unicode 32 bit
\N{BLACK STAR}	Unicode name
\o84	Octal character
\xFF	Hex character

string

print(...)

```
In [ ] : print('Single-quoted string' )
Out[ ] :
```

```
In [ ] : print("Double-quoted string" )
Out[ ] :
```

```
In [ ] : print('String with₩nnewline')
Out[ ] :
```

```
In [ ] : print('Unbroken₩
string')
Out[ ] :
```

```
In [ ] : print(r'₩n is an escape code')
Out[ ] :
```

```
In [ ] : print ("""String with
newline""")
Out[ ] :
```

- ASCII 코드값을 출력하려면,
  - ord(character)

```
In [34]: ord('a')
Out[34]: 97
```

- 문자열의 길이를 출력하려면,
  - len(string)

```
In [35]: len("hello")
Out[35]: 5
```

- 코드값을 가지고 해당 기호를 알아내려면
  - chr(codepoint)

```
In [37]: chr(10)
Out[37]: '
'
```

- 문자타입으로 변환하려면,
  - str(100)

```
In [38]: str(100)
Out[38]: '100'
```

문법(키워드, 식,문)을 이용해서

값을 입력받고, 계산/변환하고, 출력하는 흐름을 만드는 일

Operation

Operation(연산)

- 산술연산
  - 계산기
- 비교연산
  - 동등 및 대소를 비교. 참고로 '대소'비교는 '전후'비교가 사실은 정확한 표현
  - 비교 연산은 숫자값 뿐만 아니라 문자열 등에 대해서도 적용할 수 있음.
- 비트연산
- 멤버십 연산
  - 특정한 집합에 어떤 멤버가 속해있는지를 판단하는 것으로 비교연산에 기반을 둠
  - is, is not : 값의 크기가 아닌 값 자체의 정체성(identity)이 완전히 동일한지를 검사
  - in, not in : 멤버십 연산. 어떠한 집합 내에 원소가 포함되는지를 검사 ('a' in 'apple')
- 논리연산
  - 비교 연산의 결과는 보통 참/거짓. 이러한 불리언값은 다음의 연산을 적용. 참고로 불리언외의 타입의 값도 논리연산을 적용

연산자 우선 순위

	Operator	Description
lowest precedence	or	Boolean OR
	and	Boolean AND
	not	Boolean NOT
	in, not in	membership
	==, !=, <, <=, >, >=, is, is not	comparisons, identity
	|	bitwise OR
	^	bitwise XOR
	&	bitwise AND
	<<, >>	bit shifts
	+, -	addition, subtraction
	*, /, //, %	multiplication, division, floor division, modulo
highest precedence	+x, -x, ~x	unary positive, unary negation, bitwise negation
	**	exponentiation

PEMDAS :

Parentheses - Exponentiation - Multiplication - Division - Addition - Subtraction

- no ++ or --

Arithmetic	Bitwise
+=	&=
-=	|=
*=	^=
/=	>>=
%=	<<=
//=	
**=	

문법(키워드, 식, 문)을 이용해서
값을 입력받고, 계산/변환하고, 출력하는 흐름을 만드는 일

Statement

Statement

- 구문(statement) = 문
  - 예약어(reserved word, keyword)와 표현식을 결합한 패턴
  - 컴퓨터가 수행해야 하는 하나의 단일 작업(instruction)을 명시.
    - 할당(대입, assigning statement)
      - python에서는 보통 '바인딩(binding)'이라는 표현을 씀, 어떤 값에 이름을 붙이는 작업.
    - 선언(정의, declaration)
      - 재사용이 가능한 독립적인 단위를 정의. 별도의 선언 문법과 그 내용을 기술하는 블럭 혹은 블럭들로 구성.
        - » Ex) python에서는 함수나 클래스를 정의
      - 블럭
        - » 여러 구문이 순서대로 나열된 덩어리
        - » 블럭은 여러 줄의 구문으로 구성되며, 블럭 내에서 구문은 위에서 아래로 쓰여진 순서대로 실행.
        - » 블럭은 분기문에서 조건에 따라 수행되어야 할 작업이나, 반복문에서 반복적으로 수행해야 하는 일련의 작업을 나타낼 때 사용하며, 클래스나 함수를 정의할 때에도 쓰임.
    - 조건(분기) : 조건에 따라 수행할 작업을 나눌 때 사용.
      - Ex) if 문
    - 반복문 : 특정한 작업을 반복수행할때 사용.
      - Ex) for 문 및 while 문
    - 예외처리

- 조건문 형식 1

```
if 비교식:
    실행문장
```

- 조건문 형식 1의 일반 예

파이썬으로 어떻게 작성할까요?

score가 60점 이상이면 조건

→ 합격입니다. 실행문장

- 조건문 형식1 예

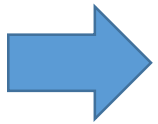
```
if score >= 60:
    print ( '합격입니다.' )
```

- 조건문 형식 2

```
if 비교식:
    실행문장
else :
    실행문장
```

- 조건문 형식2 예

score가 60점 이상이면 조건
→ 합격입니다. 실행문장
그렇지 않으면 위의 조건이 아닐때
→ 불합격입니다. 실행문장

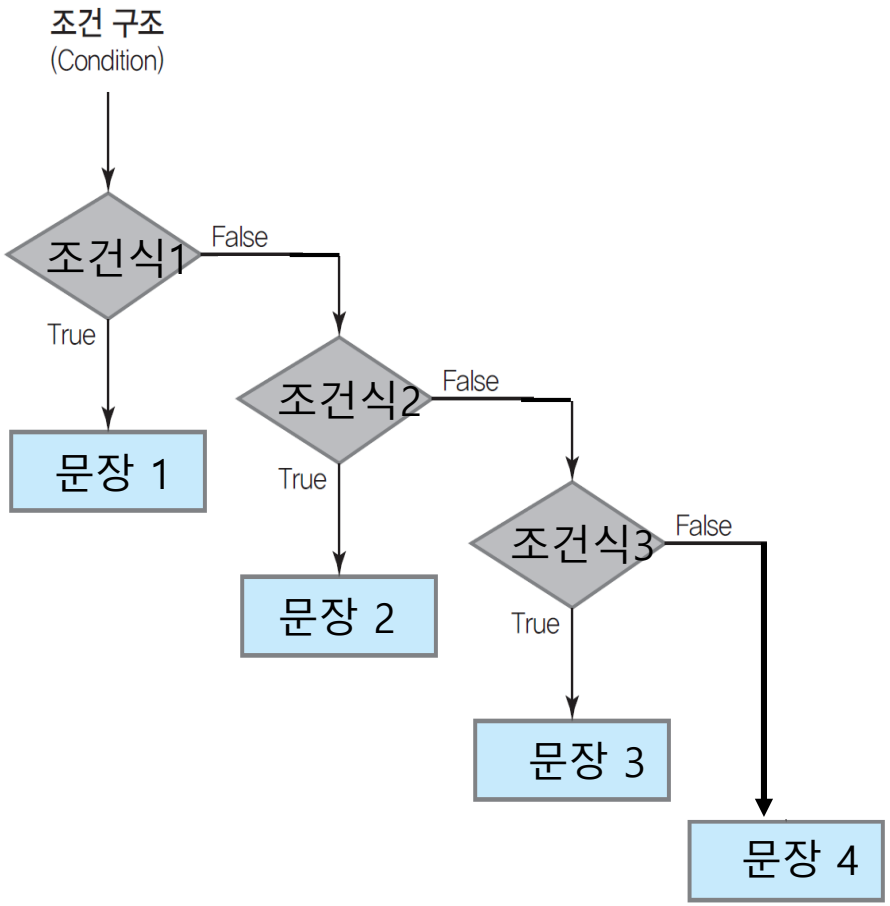


• 조건문 형식2 예

```
if score >= 60:
    print ('합격입니다. ' )
else :
    print ('불합격입니다.' )
```

- 조건문 형식 3

```
if 비교식1:
    실행문장1
elif 비교식2:
    실행문장2
elif 비교식3:
    실행문장3
else :
    실행문장4
```



- 조건문 형식3 예

```
if score >= 90:
    print ('A입니다. ' )
elif score >= 80 :
    print ('B입니다.' )
elif score >= 70 :
    print ('C입니다.' )
elif score >= 60 :
    print ('D입니다.' )
else:
    print ('F입니다.' )
```

- Quiz
  - 출력결과가 어떻게 다를까요?

```
score = 75
if score >= 90:
    print ('A입니다. ' )
elif score >= 80 :
    print ('B입니다.' )
elif score >= 70 :
    print ('C입니다.' )
elif score >= 60 :
    print ('D입니다.' )
else:
    print ('F입니다.' )
```

```
score = 75
if score >= 90:
    print ('A입니다. ' )
if score >= 80 :
    print ('B입니다.' )
if score >= 70 :
    print ('C입니다.' )
if score >= 60 :
    print ('D입니다.' )
else:
    print ('F입니다.' )
```

Indentation

- 들여쓰기의 중요성
  - 조건이 True일 때 실행해야 할 문장들을 블록으로 만들어 줘야 함
  - 블록 : 동일한 조건하에 실행되는 문장들을 묶어놓은 것
  - 동일 블록은 들여쓰기가 일치해야 함

```
if score > 90 :
    print ('합격입니다.')
    print ('A 등급입니다.')
```

블록: 동일한 조건하에 실행되어야 할 문장들을 묶은 것

Indentation

- 블록

```
if guess < number:
    print ('Your guess is too low.')

if guess > number:
    print ('Your guess is too high.')

if guess == number:
    print ('Good job, ' + myName + '! You guessed my number!')
    print ('I am happy.')
```

블록 1

블록 2

블록 3

Indentation

- 블록
  - 하나의 조건 아래 들여쓰기 길이가 다르면 다른 블록으로 간주하여 에러를 발생시킴

```
if score > 90 :
    print ('합격입니다.')
    print ('A 등급입니다.')
print ('본 과정을 수료하셨습니다.')
    print ('효도하시는군요!')
```

```
if area == '서울' :
    if gu == '성동구' :
        print (area, gu,  ' 명품교육의 도시')
    elif gu == '노원구' :
        print (area, gu, ' 힐링의 도시')
```

- 들여쓰기가 다른데 에러 날까요?

Indentation

- Quiz

- guess=10, number=12일때 예상되는 실행결과는?

```
if guess < number:
    print ('Your guess is too low.')

if guess > number:
    print ('Your guess is too high.')

if guess == number:
    print ('Good job, ' + myName + '! You guessed my number!‘)
print ('I am happy.')
```

```
In [28]: 'ABC' > 'abc'
Out[28]: False

In [29]: 'z1Ab' > 'a3Bc'
Out[29]: True

In [30]: 'z'>'A'
Out[30]: True

In [31]: '123' > '96'
Out[31]: False

In [33]: '123' < '124'
Out[33]: True
```

비교연산자

- 비교연산자 종류

>	<	>=	<=	==	!=
크다	작다	크거나 같다	작거나 같다	같다	같지않다

- 비교의 결과는 불린형으로 나옴
  - Boolean type : True or False

```
>>> 0 < 6
True
>>> 0 > 6
False
>>> 10 < 10
False
>>> 10 == 10
True
```

비교연산자

- Quiz

```
>>> 11 >= 10
>>> 11 => 10

>>> 10 <= 11
>>> 10 =< 11

>>> 11 == 11
>>> 11 = 11
```

```
>>> 'Hello' == 'Hello'
>>> 'Hello' == 'HELLO'

>>> 5 == 5
>>> 5 == '5'

>>> 5 == int('5')
```

실습1 . 큰 수/작은 수/같은 수 출력하기

- 실습 2에 아래와 같이 두 수를 입력 받아 큰 수, 작은 수, 같은 수 인지를 출력하는 프로그램을 추가하세요.
  - if문과 비교연산자, int()를 사용하여 작성해보세요.

```
첫 번째 정수를 입력하시오: 100
두 번째 정수를 입력하시오: 200

큰수: 200
작은수: 100
>>>
첫 번째 정수를 입력하시오: 56
두 번째 정수를 입력하시오: 24

큰수: 56
작은수: 24
>>>
첫 번째 정수를 입력하시오: 45
두 번째 정수를 입력하시오: 45

두 수는 같습니다.
>>>
```

실습2: 합격 여부 판단하는 프로그램

- 아래와 같이 성적을 입력 받아 90점 이상이면 합격, 90점 미만이면 불합격 여부를 판단하는 프로그램을 작성하시오.

```
점수를 입력하시오.
97
합격입니다.
>>>
점수를 입력하시오.
76
불합격입니다.
>>>
```

Ln: 3 Col: 4

실습3: 짝수 홀수 검사하는 프로그램

- 아래와 같이 숫자를 입력 받아 짝수인지 홀수인지 검사하는 프로그램을 나머지 연산자(%)를 이용하여 작성하시오.

```
숫자를 입력하시오.
97
홀수입니다.
>>>
숫자를 입력하시오.
76
짝수입니다.
>>>
```

Ln: 3 Col: 4