단국대학교
SW중심대학

창의적 사고와 코딩

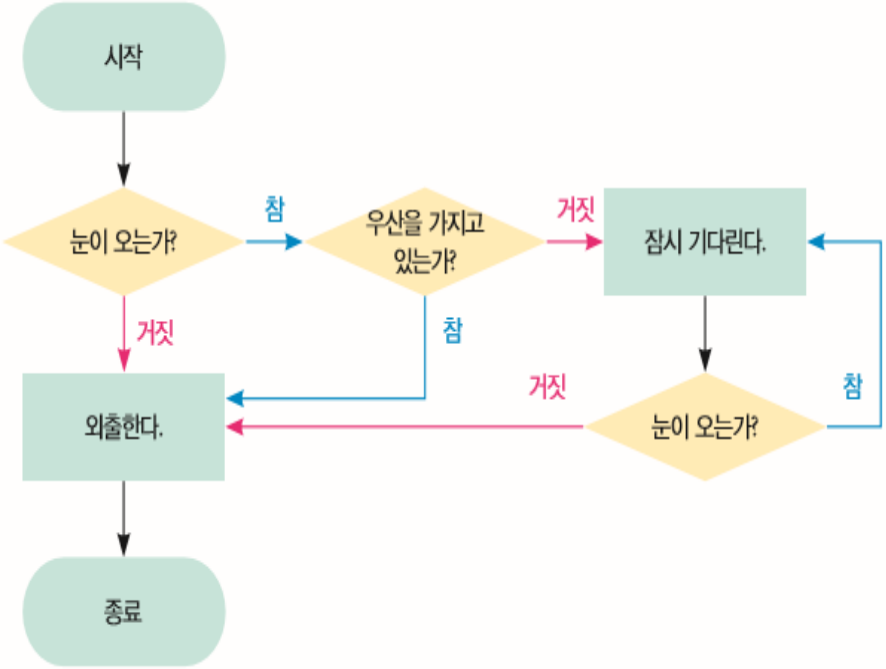
Lecture 3. 선택

1. 기본 제어구조

순서도

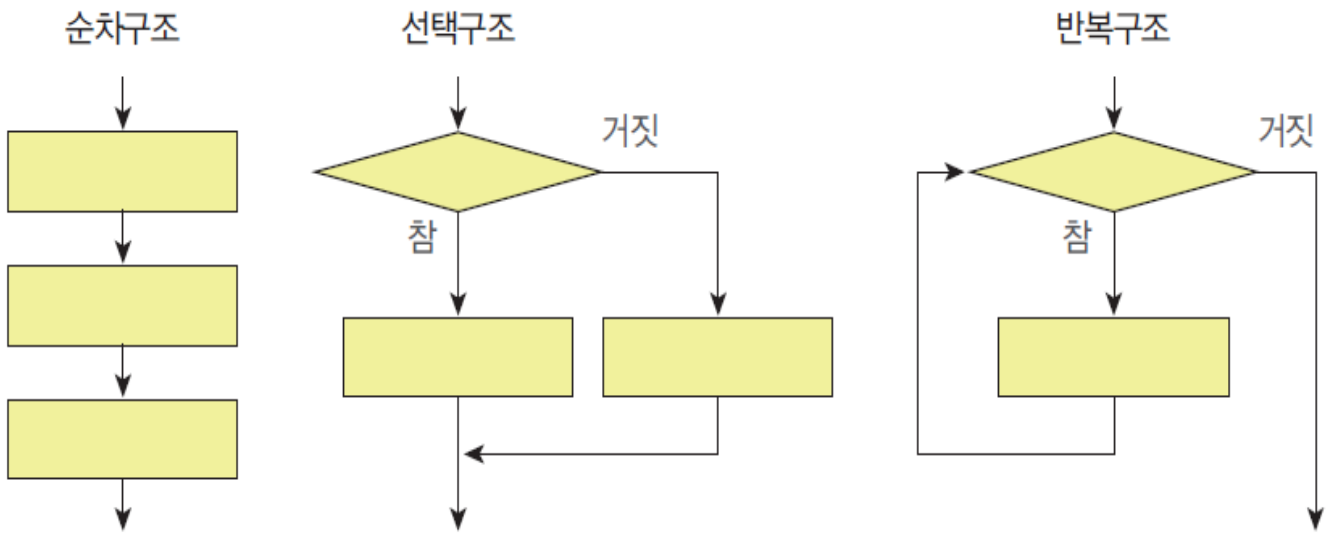
순서도 기호

기호	설명
	시작/종료
	도형들을 연결한다.
	입력과 출력
	처리
	판단



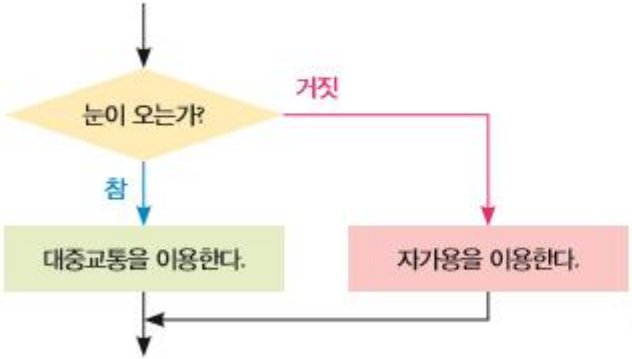
3가지의 기본 제어 구조

- 순차 구조(sequence) - 명령들이 순차적으로 실행되는 구조이다.
- 선택 구조(selection) - 둘 중의 하나의 명령을 선택하여 실행되는 구조이다.
- 반복 구조(iteration) - 동일한 명령이 반복되면서 실행되는 구조이다.



선택 구조가 필요한 이유

- 우리가 문제를 해결할 때 어떤 조건에 따라서 두 개 또는 여러 개의 실행 경로 가운데 하나를 선택해야 하는 경우가 종종 있다.



부울형 (Bool Type)

부울형은 True와 False 값을 가짐

- '참'과 '거짓'을 표현하는데 사용

True	'참'을 의미
False	'거짓'을 의미

```
>>> a = True
>>> type(a)
<class 'bool'>
>>> a
True
```

```
>>> a = TRUE
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    a = TRUE
NameError: name 'TRUE' is not defined
>>> a = true
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    a = true
NameError: name 'true' is not defined
```

```
>>> a = False
>>> b = a
>>> type(b)
<class 'bool'>
>>> b
False
```

부울형 (Bool Type)

- Python 내부에서는 True는 1로 False는 0의 값을 갖는다.

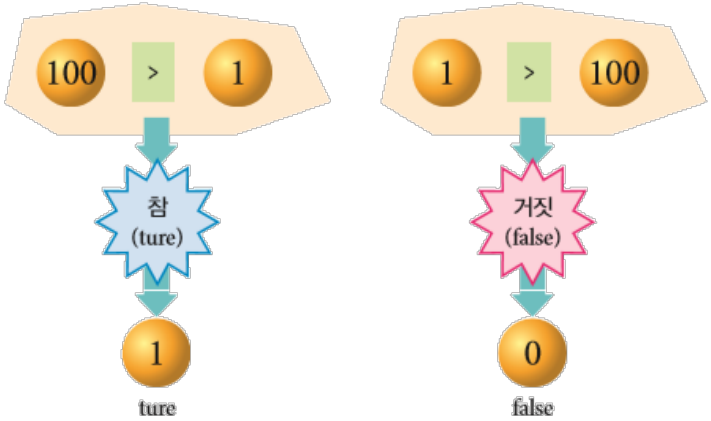
```
>>> True + 1
2
>>> False + 1
1
```

	거짓(False)	참(True)
정수	0	0 이외의 모든 정수
실수	0.0	0.0 이외의 모든 실수
문자	'' 또는 ""	빈문자 이외의 모든 문자
리스트	[]	빈리스트 [ ] 이외의 모든 리스트

관계 연산자

- 관계 연산자는 두 개의 숫자 또는 객체 간의 크고 작음을 비교하는 연산자
  - 관계 수식은 참이나 거짓이라는 값을 생성한다

관계 연산자	연산	설명	예제
<	x < y	x가 y보다 작은가?	>>> 5 < 2 False
<=	x ≤ y	x가 y보다 작거나 같은가?	>>> 20 <= 18 False
>	x > y	x가 y보다 큰가?	>>> 20 > 15 True
>=	x ≥ y	x가 y보다 크거나 같은가?	>>> 5 >= 2 True
!=	x != y	x가 y와 같지 않은가?	>>> 10 != 11 True
==	x == y	x가 y와 같은가?	>>> 15 == 10 False



관계 연산자

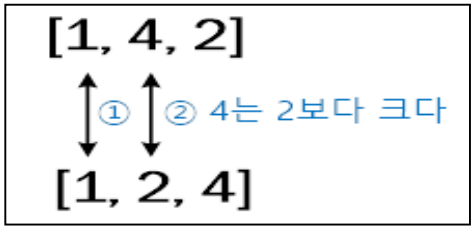
- Python은 다음과 같이 복합적인 관계식도 가능

```
>>> a = 5
>>> b = 10
>>> 0 < a < b
True
>>> 0<a and a<b
True
```

- 관계 연산자는 숫자뿐만 아니라 객체간에서도 크기를 비교할 수 있다.

```
>>> 'abcd' > 'abc'
True
>>> [1,4,2] == [1,2,4]
False
```

- 문자열의 경우에는 일반적인 사전에서 나오는 단어의 순서와 동일하다.
- 리스트의 경우에는 앞에서부터 하나씩 비교해 나간다.
- 첫번째 항목을 비교하고 같다면 두번째 항목을 비교한다. 이와 같이 계속 비교하여 같이 않을 때까지 또는 끝까지 비교하여 결과를 반환한다.



관계 연산자

- 서로 다른 데이터형 간의 값은 비교할 수 없다. ➔ 오류 발생 (정수,실수는 비교가능!!)
- 숫자 데이터 형은 예외로 비교 가능(정수와 실수인 경우만)

```
>>> 123 > 'a'  # 서로 다른 형은 비교 할수 없다.
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module>
    123 > 'a'
TypeError: '>' not supported between instances of 'int' and 'str'
>>>
```

비교가능

```
>>> 'a' == 1
False
>>> 'a' != 1
True
```

```
>>> [1,2,3] == 1
False
>>> [1,2,3] != 1
True
>>> 1 == 1.0
True
```

```
>>> 1+3j <1
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    1+3j <1
TypeError: '<' not supported between instances of 'complex' and 'int'
>>>
```

```
>>> 123 > 150.0
False
```

할당 연산자와 관계 연산자

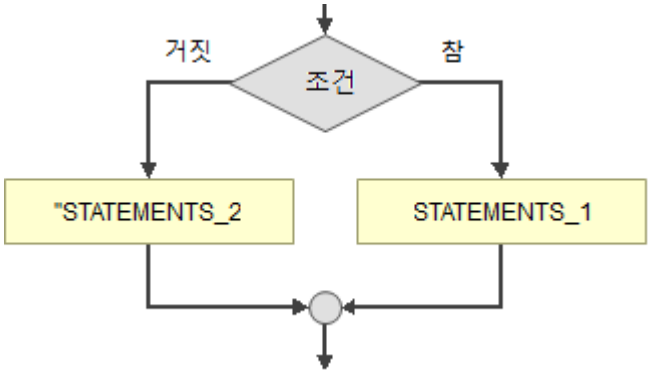
- 숫자를 입력 받아서 2의 배수인지의 여부를 flag 변수에 할당해보자.

```
>>> n = int(input('Enter a number: '))
Enter a number: 10
>>> flag = n % 2
>>> flag == 0
True
```

2. IF-ELSE 문

if – else 문

- 파이썬에서 선택 구조를 위해 if문 사용
- 들여쓰기(indent)를 주의하여 작성해야 올바르게 실행
  - 들여쓰기: tab 1번 또는 space 4번
  - 내어쓰기: backspace 키 이용




```
if  BOOLEAN EXPRESSION: # BOOLEAN EXPRESSION is called the condition
                        #  불리언 식의 결과는 부울형 반환
        STATEMENTS_1 # executed if condition evaluates to True
else:
        STATEMENTS_2 # executed if condition evaluates to False
```

if – else 문

전체적인 구조

조건식

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

콜론은 아직 문장이 종료되지 않았다는 것을 의미한다.

조건식이 참일 때 실행되는 문장, then 절이라고 한다.

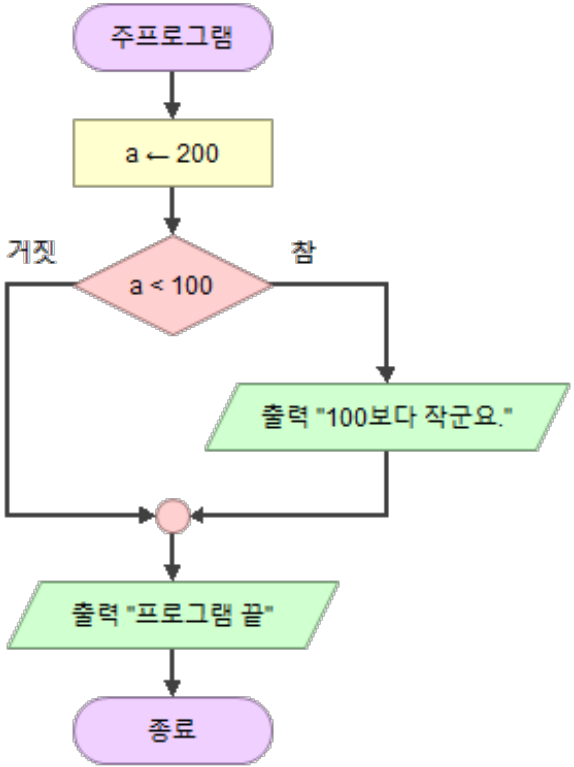
조건식이 참이 아닐 때 실행되는 문장, else 절이라고 한다.

'그렇지 않으면'이라는 의미이다.

if – else 문

if 문만 있는 경우: else 생략

- a의 값이 100 미만인 경우, 즉 조건이 참(True)인 경우 "STATEMENTS_1" 블록이 수행되는 예제




```
a = 200

if a < 100 :
    print("100보다 작군요")

print("프로그램 끝")
```

```
a = 200

if a < 100 : print("100보다 작군요")
#권장하지 않음
print("프로그램 끝")
```

yunyi96@gmail.com

예제

```
food = '스테이크'

if food == '스테이크':
    print('내가 제일 좋아하는 음식 !')
    print(10 * food)
```

```
내가 제일 좋아하는 음식 !
스테이크스테이크스테이크스테이크스테이크스테이크스테이크스테이크스테이크스테이크
```

Lab: 물건값 계산

- 인터넷 쇼핑몰에서 물건을 구입할 때, 구입액이 10만원 이상이면 5%의 할인을 해준 다고 하자. 사용자에게 구입 금액을 물어보고 최종적으로 할인 금액과 지불 금액을 출력하는 프로그램을 작성해보자.

price = int(input(" ~ "))

```
구입 금액을 입력하시오: 100500
지불 금액은  95475.0 입니다.
```

if price >= 100000:
discount = price * 0.05
real price = price - discount
final

Solution : 파일명 cal_shopping.py

```
sales = int(input("구입 금액을 입력하시오: "))
if sales > 100000 :
        discount = sales * 0.05
        sales = sales - discount
print("지불 금액은 ", sales, "입니다. ")
```

```
sales = int(input("구입 금액을 입력하시오: "))
if sales > 100000 :
        discount = sales * 0.05
        sales = sales - discount
print("지불 금액은 ", sales, "입니다. ")
print("지불 금액은 %d입니다." % sales)
```

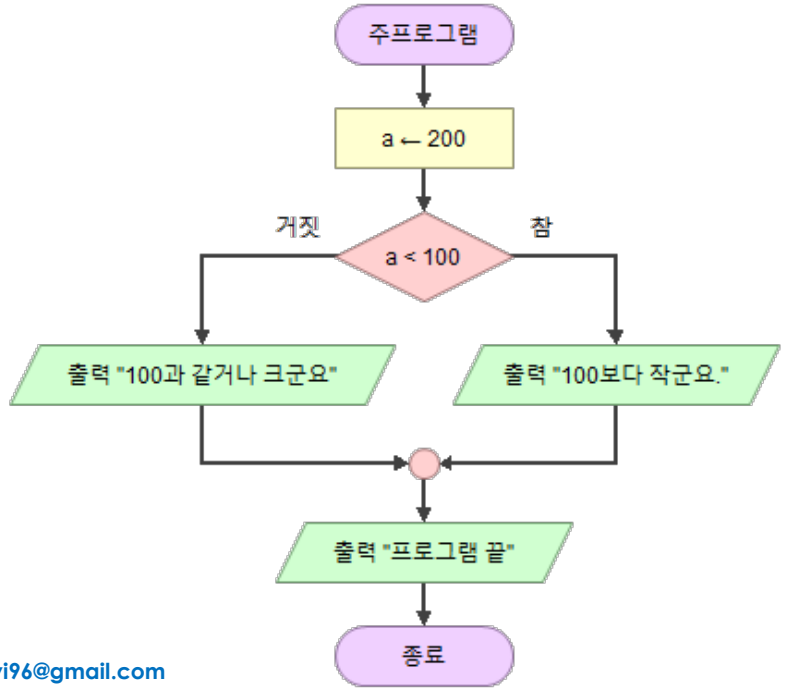
정수

```
구입 금액을 입력하시오: 100500
지불 금액은  95475.0 입니다.
지불 금액은 95475입니다.
```

if – else 문

if – else 문이 모두 있는 경우

- a의 값이 100 미만인 경우, 즉 조건이 참(True)인 경우 "STATEMENTS_1" 블록이 수행되고
  a의 값이 100 이상인 경우(조건이 거짓(False)인 경우) "STATEMENTS_2" 블록이 수행




```
a = 200

if a < 100 :
    print("100보다 작군요")
else:
    print("100과 같거나 크군요")
print("프로그램 끝")
```

yunyi96@gmail.com

if – else 문

```
num = int(input('정수 입력: '))
if num < 100 :
    print("입력한 수 %d는 100보다 작다." % num)
else:
    print("입력한 수 %d는 100보다 같거나 크다." % num)
    print("프로그램 종료")
```

```
정수 입력: 200
입력한 수 200는 100보다 같거나 크다.
프로그램 종료
>>>
정수 입력: 50
입력한 수 50는 100보다 작다.
>>>
```

```
num = int(input('정수 입력: '))
if num < 100 :
    print("입력한 수 %d는 100보다 작다." % num)
else:
    print("입력한 수 %d는 100보다 같거나 크다." % num)
print("프로그램 종료")
```

```
정수 입력: 200
입력한 수 200는 100보다 같거나 크다.
프로그램 종료
>>>
정수 입력: 50
입력한 수 50는 100보다 작다.
프로그램 종료
```

예제

```
age = 19

if( age >=  19) :
    print("마트에서 주류를 구입할 수 있습니다. ")
else :
    print("조금 기다리세요!")
```

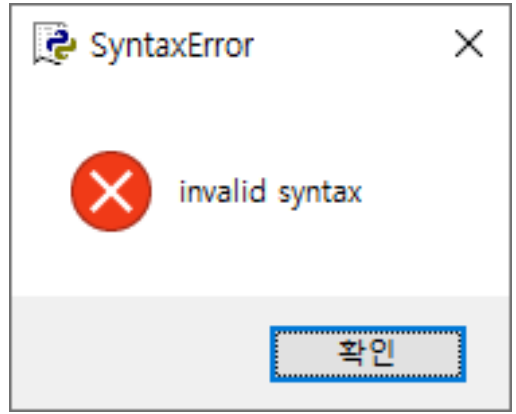
```
마트에서 주류를 구입할 수 있습니다.
```

else 문

- **다음과 같은 code의 경우는? error 발생**
  - else 문은 if문에서 마지막에 1번만 나와야 한다.

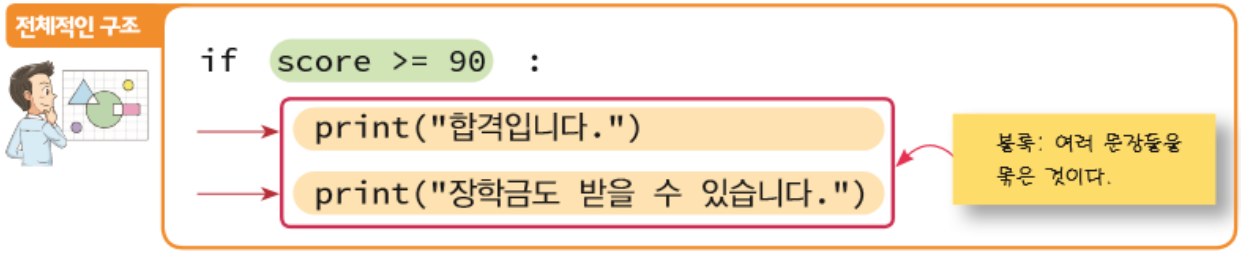
```
x = -10
if x < 0:
    print("The negative number ",  x, " is not valid here.")
else:
    print(x, " is a positive number")
else:
    print("This is always printed")
```

elif



블록(Block)

- 만약 조건이 참인 경우에 여러 개의 문장이 실행되어야 한다면 어떻게 하여야 하는가?

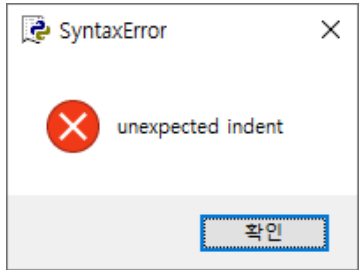


- 블록은 하나 이상의 명령문(statements)으로 구성되어 명령문 세트를 마치 단일 명령문인 것처럼 처리하는데 사용.

동일한 블록에 속해야하는데 공백이
더 추가된 경우 오류 발생

```
if score >= 90:
    print('합격입니다.')
     print('장학금도 받을 수 있습니다.')
```

```
if score >= 90:
    print('합격입니다.')
    print('장학금도 받을 수 있습니다.')
```



블록은 들여쓰기로 구분

```
if sales > 1000 :
    discount = sales*0.1
    print(discount, "할인되었음!")
else:
    if sales > 500 :
        discount = sales*0.05
        print(discount, "할인되었음!")
    else:
        print("할인은 없습니다!")
```

들여쓰기 수준 0 1 2

Lab: 수하물 비용 계산

- 항공사에서는 짐을 부칠 때, 20kg이 넘어가면 20,000원을 내야한다고 하자. 20kg 미만이면 수수료는 없다. 사용자로부터 짐의 무게를 입력받고 사용자가 지불하여야 할 금액을 계산하는 프로그램을 작성해보자.

```
짐의 무게는 얼마입니까? 18
짐에 대한 수수료는 없습니다.
감사합니다.
>>>
```

```
짐의 무게는 얼마입니까? 30
무거운 짐은 20,000원을 내셔야 합니다.
감사합니다.
>>>
```

Solution : 파일명 if_weight.py

Lab: 홀수 짝수 구별하기

- 키보드에서 입력받은 정수가 홀수인지 짝수인지를 말해주는 프로그램을 작성하여 보자.
  홀수와 짝수는 어떻게 구별할 수 있는가?

```
정수를 입력하시오: 5
입력된 정수는 홀수입니다.
```

```
정수를 입력하시오: 4
입력된 정수는 짝수입니다.
```

Solution : 파일명 even_odd.py

Lab: 두수 중 큰 수 출력

- 사용자로부터 두개의 정수를 입력받아서 둘 중에서 큰 수를 출력하는 프로그램을 작성하여 보자.

```
첫 번째 정수: 10
두 번째 정수: 20
큰 수는  20
```

Solution : 파일명 larger.py

Lab: 문자열의 중간문자

- 문자열의 중앙에 있는 문자를 출력한다. 예를 들어서 문자열이 "weekday"이라면 중앙의 문자는 "k"가 된다. 하지만 만약 문자열이 짝수개의 문자를 가지고 있다면 중앙의 2개의 글자를 출력한다. 예를 들어서 "string" 문자열에서는 "ri"를 반환한다.

```
문자열을 입력하시오: weekday
중앙글자는 k
```

```
문자열을 입력하시오: string
중앙글자는 ri
```

Solution

Lab: 임금 계산

- 사용자에게 근무 시간과 시간당 임금을 물어본다.
- 주당 근무 시간이 40시간을 넘으면 1.5배의 임금을 지급해야한다고 하자. 이번 주에 받는 총 임금을 계산하는 프로그램을 작성해보자.

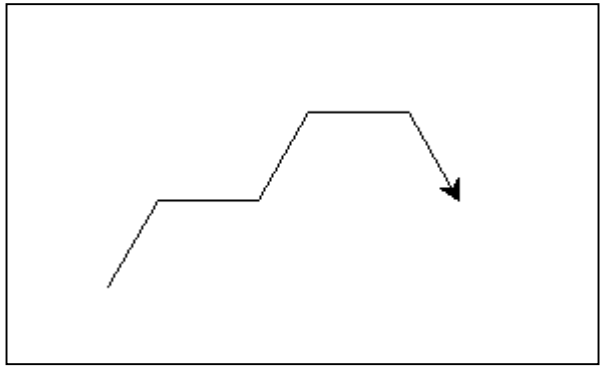
```
근무시간을 입력하시오: 45
시간당 임금을 입력하시오: 5000
총임금은 237500원 입니다.
```

Solution : 파일명 cal_wage.py

Lab: 터틀 그래픽

- 사용자로부터 명령어를 받아서 터틀을 제어해보자. 즉 사용자가 "left"를 입력하면 왼쪽으로 60회전하게 하고 사용자가 "right"를 입력하면 오른쪽으로 60회전하게 하자.

```
왼쪽=left, 오른쪽=right : left
왼쪽=left, 오른쪽=right : right
왼쪽=left, 오른쪽=right : left
왼쪽=left, 오른쪽=right : right
왼쪽=left, 오른쪽=right : right
왼쪽=left, 오른쪽=right :
```



무한 반복 구조

- 아직 학습하지 않았지만 다음과 같은 코드를 사용하면 무한 반복할 수 있다.

```
while True :
        …
        …
        …
```

Solution : 파일명 if_turtle.py

```
import turtle
t=turtle.Pen()

while True: #무한반복
```

사용자가 "left"를 입력하면 왼쪽으로 60회전하게 하고 사용자가 "right"를 입력하면 오른쪽으로 60회전하게 하자.