

Intro to Computer Science

Local Laboratory

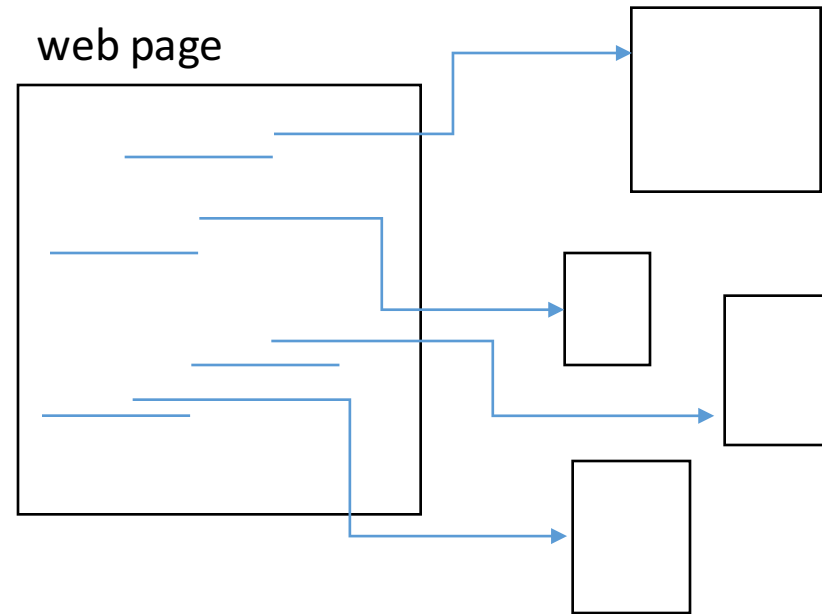
*** Udacity – Intro to Computer Science**

Introduction

Unit 2:

How to Repeat

Procedures & Control



Comparisons

< > <= ...

<Number> <Operator> <Number>
<숫자> <비교연산자> <숫자>

Boolean value : True False

code

```
print( 2 < 3 )  
print( 21 < 3 )  
print( 7 * 3 < 21 )  
  
print( 7 * 3 != 21 )  
print( 7 * 3 == 21 )
```

result

```
True  
False  
False  
False  
True
```

Quiz: Equality Comparisons

왜 동일 비교 시, `=` 대신 `==`를 사용할까?

- ❑ 왜냐하면 `=` 는 대략적으로 같다는 것을 뜻하기 때문이다.
- ❑ 왜냐하면 같지 않다는 것을 `!=` 기호로 표현하기 때문이다.
- ❑ 왜냐하면 Guido 기호 `=` 를 좋아하지 않기 때문이다.
- ❑ 왜냐하면 `=` 기호는 할당의 뜻을 의미하기 때문이다.
- ❑ 사실 `==` or `=` 기호 둘다 사용이 가능하다.

`i = 21`

-> 변수 `i`에 숫자 값 21을 할당한다

`i == 21`

-> `True` or `False`

Quiz: If Statements

True or False
if <조건 표현문>:
 <Block>

```
def absolute(x):  
    if x < 0:  
        x = -x  
    return x
```

[실습]

두 숫자를 입력받아, 둘 중 더 큰 숫자를 출력하는
`bigger` 라는 프로시저를 정의하시오.

```
bigger(2, 7) -> 7  
bigger(3, 2) -> 3  
bigger(3, 3) -> 3
```

If Statements

```
def bigger(a, b):  
    if a > b:  
        return a  
    return b
```



```
def bigger(a, b):  
    if a > b:  
        return a  
    else:  
        return b
```

```
if <조건 표현문>:  
    <Block>  
else:  
    <Block>
```

Quiz: Is Friend

[실습]

문자열을 입력(친구의 이름)받아 Boolean을 출력(친구가 맞는지 아닌지)하는 `is_friend` 라는 프로시저를 정의하시오. (이름이 'D'로 시작하는 사람을 친구로 가정)

```
print( is_friend('Diane') ) -> True  
print( is_friend('fred') ) -> False
```

Is Friend

code

```
def is_freind(name):  
    if name[0] == 'D':  
        return True  
    else:  
        return False  
  
print( is_friend('Doug') )  
print( is_friend('Fred') )
```

result

```
True  
False
```


Quiz: More Friends

[실습]

문자열을 입력(친구의 이름)받아 Boolean을 출력(친구가 맞는지 아닌지)하는 `is_friend` 라는 프로시저를 정의하시오. (이름이 'D' 또는 'N' 시작하는 사람을 친구로 가정)

```
print( is_friend('Diane') -> True  
print( is_friend('Ned') -> True
```

More Friends

code

```
def is_freind(name):  
    if name[0] == 'D':  
        return True  
    if name[0] == 'N':  
        return True  
    return False  
  
print( is_friend('Doug') )  
print( is_friend('Nicole') )  
print( is_friend('Fred') )
```

result

```
True  
True  
False
```

code

```
def is_freind(name):  
    if name[0] == 'D':  
        return True  
    else:  
        if name[0] == 'N':  
            return True  
        else:  
            return False  
  
print( is_friend('Doug') )  
print( is_friend('Nicole') )  
print( is_friend('Fred') )
```

result

```
True  
True  
False
```

Or

code

```
def is_freind(name):  
    return name[0] == 'D' or name[0] == 'N'  
  
print( is_friend('Doug') )  
print( is_friend('Nicole') )  
print( is_friend('Fred') )
```

result

```
True  
True  
False
```

code

```
print( True or False )  
print( False or True )  
print( True or True )  
print( False or False )
```

result

```
True  
True  
True  
False
```

Quiz: Biggest

[실습]

3개의 숫자를 입력으로 받아서, 세 숫자 중 가장 큰 수를 출력하는 `biggest`라는 프로시저를 정의하시오.

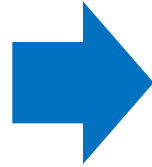
`biggest(6, 2, 3) -> 6`

`biggest(6, 2, 7) -> 7`

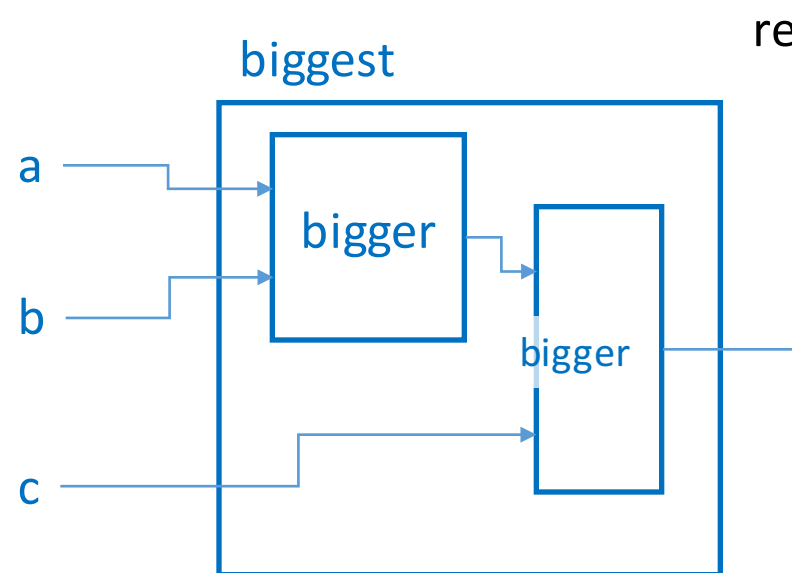
`biggest(6, 9, 3) -> 9`

Biggest

```
def biggest(a, b, c):  
    if a > b:  
        if a > c:  
            return a  
        else:  $c \geq a > b$   
            return c  
    else:  $b \geq a$   
        if b > c:  
            return b  
        else:  $c \geq b \geq a$   
            return c
```



```
def biggest(a, b, c):  
    return bigger( bigger(a, b), c )
```



```
def bigger(a, b):  
    if a > b:  
        return a  
    return b
```

Biggest

arithmetic
comparisons
procedures
if

Alan Turing
1912-1954

Developed abstract model
of a computer (1936)
("Turing Machine")

Proved that a machine with
a few simple operations
could simulate any other
machine



You now know enough
(in theory!)
to write every possible
computer program!

Quiz: While Loops

Loops

while <조건 표현문>:
 <Block>

0, 1, 2, ... times

if <조건 표현문>:
 <Block>

0 or 1 times

Quiz: While Loops

아래 프로그램은 어떤 동작을 할지 보기에서 고르시오.

```
i = 0
while i != 10:
    i = i + 1
    print( i )
```

- ☐ 에러 발생
- ☐ 0부터 9까지의 숫자를 출력한다
- ☐ 1부터 9까지의 숫자를 출력한다
- ☐ 1부터 10까지의 숫자를 출력한다
- ☐ 영원히 실행된다

Quiz: While Loops 2

아래 프로그램은 어떤 동작을 할지 보기에서 고르시오.

```
i = 1
while i <= 10:
    i = i + 2
    print( i )
```

- ☐ 에러 발생
- ☐ 숫자 2, 4, 6, 8 출력
- ☐ 숫자 1, 3, 5, 7, 9 출력
- ☐ 숫자 3, 5, 7, 9 출력
- ☐ 영원히 실행

Quiz: Print Numbers

[실습]

모든 양의 숫자 입력에 대해, 1부터 입력 숫자까지의 모든 숫자를 출력하는 `print_numbers`라는 프로시저를 정의하시오.

```
print_numbers(3)
```

```
1
```

```
2
```

```
3
```

Quiz: Print Numbers

code

```
def print_numbers(n):  
    i = 1  
    while i <= n:  
        print( i )  
        i = i + 1  
  
print_numbers(3)
```

result

```
1  
2  
3
```

code

```
def print_numbers(n):  
    i = 0  
    while i < n:  
        i = i + 1  
        print( i )  
  
print_numbers(3)
```

result

```
1  
2  
3
```

code

```
def print_numbers(n):  
    i = 0  
    while i < n:  
        i = i + 1  
        print( i )  
  
print_numbers(0)
```

result

Quiz: Factorial


[실습]

하나의 숫자를 입력받고 그 수의 팩토리얼 값을 출력하는 `factorial` 이라는 프로시저를 정의하시오.

$$\text{factorial}(n) = n * (n-1) * (n-2) * \dots * 2 * 1$$

Break

```
while <조건문>:  
    <Code>  
    if <조건문>:  
        break  
    <More Code>  
<After While>
```



```
def print_numbers(n):  
    i = 1  
    while i <= n:  
        print(i)  
        i = i + 1
```



```
def print_numbers(n):  
    i = 1  
    while True:  
        if i > n:  
            break  
        print(i)  
        i = i + 1
```

Multiple Assignment

```
def get_next_target(page):  
    start_link = page.find('<a href=  
    start_quote = page.find('"', start_link)  
    end_quote = page.find('"', start_quote + 1)  
    url = page[start_quote + 1 : end_quote]  
    return url, end_quote
```

<변수명> = <Expression>

Multiple Assignment

<변수_1>, <변수_2>, ... = <Expression_1>, <Expression_2>, ...

a, b = 1, 2 a ↘₁ b ↘₂ url, end_quote = get_next_target(page)

Quiz: Multiple Assignment

다음 표현의 결과는 어떻게 될지 예측해보시오.

$s, t = t, s$
 2 1

$s \rightarrow 1$

$t \rightarrow 2$

$s = t$
 $t = s$

- ☐ Nothing
- ☐ s와 t 모두를 원래의 t 값으로 할당한다
- ☐ s와 t값을 변경한다
- ☐ Error

Quiz: No Links

[실습]

아래의 코드는 이전에 정의했던 `get_next_target` 함수이다. 타당한 링크가 문자열에 포함되었을 경우, 이 함수는 링크와 마지막 따옴표의 위치를 리턴한다. 문자열에 링크가 없을 경우를 검출하기 위해 함수를 수정하십시오.

(만약 링크가 문자열에 없다면, 함수는 아래 내용을 출력해야한다.)

None, 0

code

```
def get_next_target(page):
    start_link = page.find('<a href=')

    #Insert your code below here

    start_quote = page.find('"', start_link)
    end_quote = page.find('"', start_quote + 1)
    url = page[start_quote + 1:end_quote]
    return url, end_quote
```


No Links

code

```
def get_next_target(page):
    start_link = page.find('<a href=')

    if start_link == -1:
        return None, 0

    start_quote = page.find('"', start_link)
    end_quote = page.find('"', start_quote + 1)
    url = page[start_quote + 1:end_quote]
    return url, end_quote
```

Print All Links

```
page = ... contents of some web page ...
start_link = page.find('<a href=')
start_quote = page.find('"', start_link)
end_quote = page.find('"', start_quote + 1)
url = page[start_quote + 1 : end_quote]
print(url)
page = page[end_quote:]

start_link = page.find('<a href=')
start_quote = page.find('"', start_link)
end_quote = page.find('"', start_quote + 1)
url = page[start_quote + 1 : end_quote]
print(url)
page = page[end_quote:]

start_link = page.find('<a href=')
start_quote = page.find('"', start_link)
end_quote = page.find('"', start_quote + 1)
url = page[start_quote + 1 : end_quote]
print(url)
page = page[end_quote:]
```



```
def get_next_target(page):
    start_link = page.find('<a href=')

    if start_link == -1:
        return None, 0

    start_quote = page.find('"', start_link)
    end_quote = page.find('"', start_quote + 1)
    url = page[start_quote + 1 : end_quote]
    return url, end_quote
```

Quiz: Print All Links

`print_all_links`는 `page`의 모든 링크를 출력하는 프로시저이다.
`page`의 모든 링크를 출력할 수 있도록 빈칸을 채우시오.

```
def print_all_links(page):  
    while True :  
        url, endpos = get_next_target(page)  
        if url:  
            print( url )  
            page = page[endpos:]  
        else:  
            break
```

Print All Links

code

```
def print_all_links(page):  
    while True:  
        url, endpos = get_next_target(page)  
        if url:  
            print(url)  
            page = page[endpos:]  
        else:  
            break  
  
print_all_links('this <a href="test1">link 1</a> is <a href="test2">link 2</a> a <a  
href="test3">link 3</a>')
```