Fastcampus

Computer Science Extension School

Python Basic_Day2

Recap

- Computer vs Calculator
- Computational Thinking
- Install Python, VS Code, Jupyter
- Operation, Operator
- `+ - * / // % **`
- `> < >= <= == !=`
- True or False
- Variables

Before we start

Hackerrank 30 days of code

Hackerrank python challenges

Strings

Strings

```
some_string = "python"
len(some_string)
```

- index

p	y	t	h	o	n
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

```
some_string[3:5] = "ho"
some_string[1:5:2] = "yh"
some_string[::] = some_string[0:len(some_string):1]
some_string[::-1] = some_string[-1:-len(some_string):-1]
some_string[::-1] = "nohtyp"
```

but, strings are immutable

```
>>> some_string = "python"

>>> some_string[0] = "c"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment

>>> some_string = "c" + some_string[1:]
```

String Functions

```
func = "python is easy programming language"
func.count('p')

func.find('p')

comma = ","
func = comma.join('python')

func.split(',')

python_is_easy = "python is easy"
python_is_easy.split()

python_is_easy.replace("python", "golang")
```

String Functions

```
some_string = "   computer     "
some_string.strip()
```

```
some_string = ",,,Fastcampus..."
some_string.strip(",")
some_string.strip(".")
```

String Formatting - old way

```
print("I have a %s, I have an %s." % ("pen","apple"))
```

```
%s - string
%c - character
%d - Integer(decimal)
%f - floating-point
%o - 8진수(Octal)
%x - 16진수(hexadecimal)
%% - %
```

String Formatting - New way

```
print("I have a {}, I have an {}.".format("pen", "apple"))
```

```
print("I have a {0}, I have an {1}.".format("pen", "apple"))
```

```
print("I have a {0}, I have an {0}.".format("pen", "apple"))
```

Toggl

https://blog.toggl.com/wp-content/uploads/2016/12/toggl-it-jobs-explained-with-changing-lightbulb.jpg

https://assets.toggl.com/images/toggl-how-to-save-the-princess-in-8-programming-languages.jpg

List, Tuple

List

```
animals = ['','','']
```

Tuple

```
animals = ('','','')
```

List

빈 list를 선언합니다. 선언과 동시에 값을 채워넣을 수 있습니다.

```
lang = ["python", "c", "java", "golang"]
```

lang = []

list에 요소를 추가합니다.

lang.append("python")

lang.append("java")

lang.append("golang")

print(lang)

혹은 특정한 위치에 원하는 값을 추가할 수 있습니다.

```
lang.insert(1, "c")
print(lang)
```

특정 요소를 삭제할 수도 있습니다.

```
lang.remove("golang")
print(lang)
```

혹은 리스트에 있던 값을 빼낼 수도 있습니다.

```
java = lang.pop(2)
print(lang)
print(java)
```

리스트를 정렬하는 법을 알아봅니다.

```
numbers = [2, 1, 4, 3]
print(numbers)

numbers.sort()
print(numbers)
```

리스트를 역순으로 출력하고 싶을땐 이렇게 한답니다.

```
numbers = [2, 1, 4, 3]
numbers.reverse()
print(numbers)
```

리스트를 내림차순으로 정렬하려면??

1. sort -> reverse

```
numbers.sort()
numbers.reverse()
```

2. sort(reverse=True)

```
numbers.sort(reverse=True)
```

특정 값의 위치를 출력할땐 이렇게 합니다.

```
index_of_two = numbers.index(2)
print(index_of_two)
```

리스트끼리 더할 땐 extend를 활용합니다.

```
numbers += [5, 6]
print(numbers)
numbers.extend([7, 8])
print(numbers)
```

Tuple

Tuple은 괄호를 이용해 선언할 수 있습니다.

tuple1 = (1, 2, 3, 4)

tuple은 삭제나 추가가 불가능합니다.

```
del tuple[1]
tuple1[1] = 'c'
```

tuple끼리 더하거나 반복하는 것은 가능합니다.

tuple2 = (5, 6)

print(tuple1 + tuple2)

print(tuple1 * 3)

tuple은 값을 편하게 바꿀 수 있습니다.

```
x = y
y = x (x)

temp = x
x = y
y = temp

(x,y) = (y,x)
```

혹은 함수에서 하나 이상의 값을 반환할 때 사용합니다.

```
def quot_and_rem(x,y):
    quot = x // y
    rem = x % y
    return (quot, rem)

(quot, rem) = quot_and_rem(3,10)
```

List <-> Tuple

```
list((1,2))
tuple([1,2])
```

Dictionary, Set

dictionary의 선언

```
dict1 = {}
print(dict1)
```

dictionary는 key와 value로 이루어져 있으며, 추가하는 법은 다음과 같습니다.

```
dict1 = {'name': 'foo bar'}
print(dict1)

dict1 = {'korean': 95, 'math': 100, 'science': [80, 70, 90, 60]}
print(dict1)

dict1['english'] = "pass"
print(dict1)
```

요소 삭제는 del을 활용합니다.

```
del dict1['math']
print(dict1)
```

key를 활용해 value를 출력하는 법을 알아봅시다.

```
print(dict1['korean'])
```

key만 출력하는 법을 알아봅시다.

```
print(dict1.keys())
```

value만 출력할땐 이렇게 합니다.

```
print(dict1.values())
```

key와 value를 함께 출력합니다.

```
print(dict1.items())
```

Small Quiz

A = 'fastcampus'

B = 'python'

A ∪ B

A ∩ B

A - B

A Δ B

Set

- 수학 집합 연산을 쉽게 하기 위해 만든 자료형
- 순서없음
- 중복없음

Set

Set 선언

```
ppap = {'pen', 'apple', 'pineapple', 'pen'}
print(ppap)
```

```
'apple' in ppap
'applepen' in ppap
```

```
pineapple = set('pineapple')
pineapple
```

Set

A = set('fastcampus')

B = set('python')

A ∪ B == `A | B`

A ∩ B == `A & B`

A - B == `A - B`

A Δ B == `A ^ B`

조건문

Let's get back to the Day1

`배가 고프다!!!`

- case 1: 집이라면
  - 밥이 있다면
  - 밥이 없다면
- case 2: 밖이라면
  - 현금이 10만원 초과라면
  - 현금이 5만원 초과라면
  - 현금이 없다면

If

```
if 조건:
        실행문

if 조건1 and 조건2:
        실행문

if 조건1 or 조건2:
        실행문
if not 조건:
        실행문
```

Comparison Operators

```
x == n
x != n

x < n
x > n
x <= n
x >= n
```

if

```
if 현금 > 100000:
        레스토랑으로 간다
```

```
cash = 120000
if cash > 100000:
        print("go to restaurant")
```

else

```
if 조건:
        실행문1
else:
        실행문2
```

```
cash = 120000
if cash > 100000:
    print("go to restaurant")
else:
    print("go to cvs")
```

else if

```
if 조건1:
        실행문1
else:
        if 조건2:
                실행문2
        else:
                실행문3
```

```
cash = 120000
if cash > 100000:
    print("go to restaurant")
else:
        if cash > 50000:
            print("go to bobjib")
        else:
            print("go to cvs")
```

if in else in if in else in ..

```
cash = 120000
if cash > 100000:
    print("go to restaurant")
elif cash > 50000:
    print("go to bobjib")
elif cash > 30000:
    print("go to buffet")
elif cash > 20000:
    print("go to ramen store")
elif cash > 10000:
    print("go to chinese restaurant")
else:
    print("go to cvs")
```

elif

```
if 조건1:
        실행문1
elif 조건2:
        실행문2
elif 조건3:
        실행문3
...
else:
        실행문n
```

elif

```
cash = 120000
if cash > 100000:
    print("go to restaurant")
elif cash > 50000:
    print("go to bobjib")
elif cash > 30000:
    print("go to buffet")
elif cash > 20000:
    print("go to ramen store")
elif cash > 10000:
    print("go to chinese restaurant")
else:
    print("go to cvs")
```

numguess

```
import random


answer = random.randint(1,100)
print(answer)
```

numguess

```
username = input("Hi there, What's your name?? ")
guess = eval(input("Hi, "+ username + "guess the number: "))

if guess == answer:
        print("Correct! The answer was ", str(answer))
else:
        print("That's not what I wanted!! The answer was ", str(
```

numguess advanced!!

how to make it with more fun??

For, while

```
for 변수 in (리스트 or 문자열):
        실행문1
    ...
```

```
for i in ["python", "java", "golang"]:
        print(i)
```

For, while

```
sum = 0
for i in range(1,11):
        sum += i
    sum = sum + i
        print(sum)
```

For, while

```
while 조건:
        실행문1
        ...
```

```
while name != "foo bar":
        name = input("What's your name? ")
        print("Hi, " + name + "So, where is foo bar?")
```

```
while 1:
        print("Hello world!")
```

Fizzbuzz

1부터 100까지 반복

3의 배수 = "Fizz"
5의 배수 = "Buzz"
15의 배수 = "FizzBuzz"
나머지 = 그 숫자

Fizzbuzz

```
num = eval(input("type the number: "))

for i in range(1, num + 1):
        if i % 15 == 0:
                print("fizzbuzz")
        elif i % 3 == 0:
                print("fizz")
        elif i % 5 == 0:
                print("buzz")
        else:
                print(i)
```

Refactoring numguess

```
import random


answer = random.randint(1,100)
username = input("Hi there, What's your name?? ")

while True:
        guess = eval(input("Hi "+ username + ", guess the number

        if guess == answer:
                print("Correct! The answer was ", str(answer))
                break
        else:
                print("That's not what I wanted!! Try again!!")
```

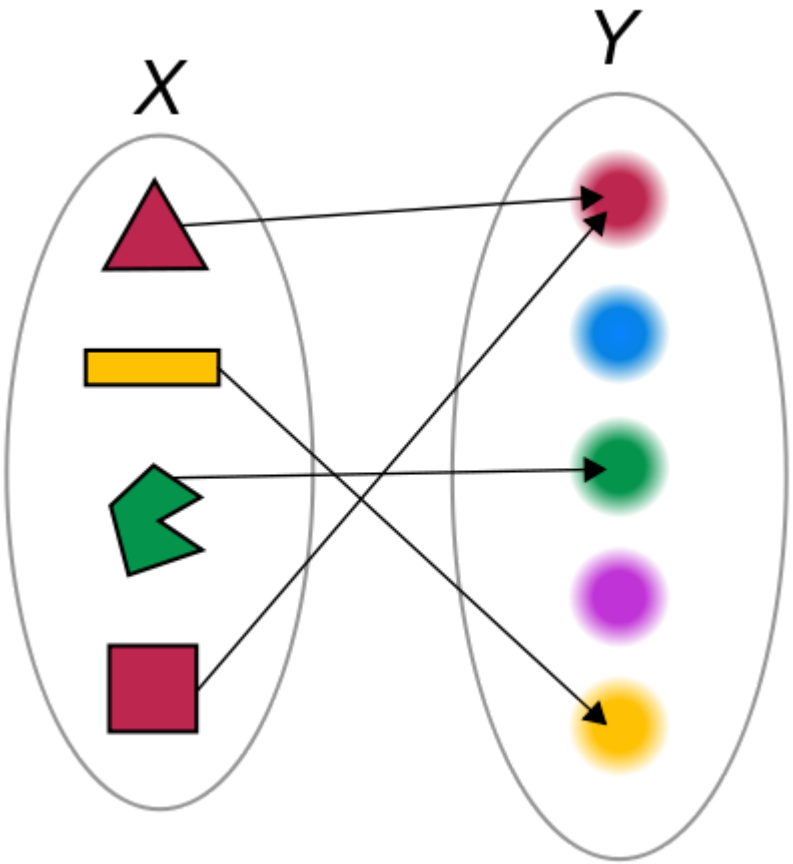
give a hint!!

```
import random


answer = random.randint(1,100)
username = input("Hi there, What's your name?? ")

while True:
    guess = eval(input("Hi, "+ username + "guess the number: "))

    if guess == answer:
        print("Correct! The answer was ", str(answer))
        break
    elif guess > answer:
        print("Too high!! Try again!!")
    elif guess < answer:
        print("Too Low!! Try again!!")
```

limit trial

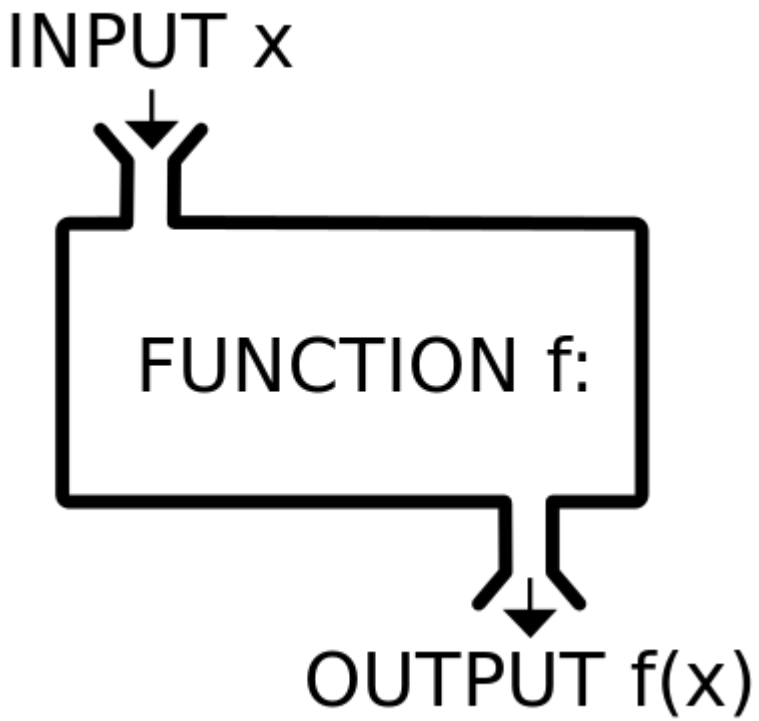
```
import random


answer = random.randint(1,100)
username = input("Hi there, What's your name?? ")
trial = 5
while trial:
    guess = eval(input("Hi, "+ username + ". guess the number: "
    if guess == answer:
        print("Correct! The answer was ", str(answer))
        break
    elif guess > answer:
        trial -= 1
        print("Too high!! Try again!!(%d times left)" % (trial))
    elif guess < answer:
        trial -= 1
        print("Too Low!! Try again!!(%d times left)" % (trial))

if trial == 0:
    print("You are Wrong! The answer was ", str(answer))
```

function



- 수학적 정의: 첫 번째 집합의 임의의 한 원소를 두 번째 집합의 오직 한 원소에 대응시키는 대응 관계
- x: 정의역 y: 공역

function



- 프로그래밍에서의 함수: 입력값을 내부에서 어떤 처리를 통해 결과값을 출력하는 것

function

```
def function(parameter):
        실행문1
        실행문2
        ...
        return output
```

function

```
def awe_sum(a,b):
        result = a + b
    return result

a = 2
b = 3
print(awe_sum(a,b))
```

function without input

```
def print_hello():
    return "hello"

result_hello = print_hello()
print(result_hello)
```

function without return

```
def func_wo_return(a):
    print("This is function without return for " + str(a) + " ti

func_wo_return()
```

function with multiple return

```
def mul_return(a):
    b = a + 1
        return a,b
```

return skill

```
def id_check(id):
        if id == "admin":
                print("invalid id: admin")
                return
        print("valid id: ", id)
```

parameter with initialize

```
def say_hello(name="Fool", nick=True):
        print("Hi, ", name)
        if nick == True:
                print("But, you are Fool")
        else:
                print("Oh, you are not Fool")
```

초기값을 설정할땐 항상 그 인자를 마지막에 두어야 합니다.

arguments

```
def mul_sum(*args):
        sum = 0
        for i in args:
                sum += i
        return sum
```

keyword arguments

```
def show_kwargs(**kwargs):
        print(str(kwargs))

show_kwargs(a=10, b="google")
```

keyword arguments

```
def kwargs_url(server, port, **query):
        url = "https://" + server + ":" + port + "?"
        for key in query.keys():
                url += key + "=" + query[key] + "&"
        return url

kwargs_url("localhost","8080", utm_source="google", keyword="nav
```

variable outside function

```
a = "hello"
def glob_test(a):
        a += "world"
        return a

glob_test(a)
print(a)
```

```
a = "hello"
def glob_test(x):
        x += "world"
        return x

glob_test(a)
print(a)
```

variable outside function

```
def glob_test2(x):
        x += "world"
        return x

glob_test2("hello")
glob_test2(x)
```

So, how to globalize

(1) using return

```
a = "hello"
def glob_test(a):
        a += "world"
    return a

a = glob_test(a)
print(a)
```

So, how to globalize

(2) use global

```
a = "hello"
def glob_test(a):
        global a
        a += "world"
    return a

glob_test(a)
print(a)
```

global 이라는 명령을 사용하여 전역변수로 사용하게 되면 함수는 독립성을 잃게 되어 함수가 외부변수에 의존적이게 됩니다.

Leap year

4로 나뉘어 떨어지면 윤년,
100으로 나뉘어 떨어지면 평년,
400으로 나뉘어 떨어질땐 윤년

Leap year(answer)

```
leap = False
def is_leap(y):
        if y % 4 == 0 and (y % 100 != 0 or y % 400 == 0)
                leap = True
        return leap

y = int(input("Is leap?? "))
print(is_leap(y))
```

numguess with function

```
def guesser(guess):
        if guess == answer:
                print("Correct! The answer was ", str(answer))
                break
        else:
                print("That's not what I wanted!! Try again!!")
```

Recursive

```
times = int(input("How many times want to curse the beast??: "))
def recurse_beast(a):
        if a == 0:
                print("curse complete!")
        else:
                print("Fusion!!!(%d times left)" % a - 1)
                recurse_beast(a-1)

recurse_beast(times)
```