Week 1: Python & JavaScript

설치

각 홈페이지에 들어가 웬만하면 가장 최근에 release된 stable version을 다운로드하면 됩니다. 설치 시 "환경 변수에 추가(Add to environmental variable)"를 꼭 체크해야 합니다. 그래야 어떤 경로에서도 Python과 JavaScript를 사용할 수 있습니다.

- Python: https://www.python.org/downloads/windows/
- JavaScript: https://nodejs.org/ko/
- Visual Studio Code: https://code.visualstudio.com/download

설치가 완료된 후, PowerShell 혹은 cmd를 실행시킨 후 "python", "node"를 입력하면 각각 파이썬과 자바스크립트가 실행됩니다.

Java와 JavaScript는 전혀 다른 언어입니다. 함부로 Java라고 줄여부르면 안 됩니다! 보통은 JS라고 줄여서 불러요.

다음과 같은 코드로 console에 문자를 출력할 수 있습니다!

```
# python
print("Hello, world!")
```

```
// javascript
console.log("Hello, world!")
```

콘솔 사용법

먼저 콘솔과 조금 친해지는 시간을 갖겠습니다.

- Directory: 폴더
- Current Directory: 현재 디렉토리
- Parent Directory: 상위 디렉토리
- Child Directory: 하위 디렉토리
- Change Directory: `cd DIRECTORY_PATH`
- Make Directory: `mkdir DIRECTORY_NAME`

- List of Current Directory: `ls` or `ls -la`
- Relative Path: `./`, `../`, `.../`
- Absolute Path: `/Users/jessekim/Projects/...`

기본 연산들

이제 python 혹은 js를 이용해 기본적인 연산들을 처리할 수 있습니다.

사칙연산

```
5 + 3 # 더하기(8)
5 - 2 # 빼기(3)
6 / 2 # 나누기(3)
3 * 3 # 곱하기(9)
4 ** 2 # 거듭제곱(16)
```

유용한 연산들

```
# python
5 // 3 # 5를 3으로 나눈 몫(1)
3 % 1 # 3을 1로 나눈 나머지(2)
```

```
// javascript
parseInt(5/3) // 5를 3으로 나눈 몫(1)
3 % 1 // 3을 1로 나눈 나머지(2)
```

복잡한(?) 연산

괄호 열고 닫기에 주의하세요!

```
((5 + 3)**2 + 5*12)/14
```

할당

할당은 특정한 값을 내가 직접 지정하는 변수에 저장합니다.

```python
# python
x = 5 # "x"라는 변수에 5를 저장합니다.
y = 3 # "y"라는 변수에 3을 저장합니다.
x + y # 8
x * y # 15

i_am_iron_man = "I am Iron Man." # 문자열을 저장합니다.
is_iron_man = True # Boolean data를 저장합니다.
```

javascript에는 두 가지 타입의 할당이 있습니다.

- const는 변하지 않는 변수를 할당합니다.
- let은 변할 수 있는 변수를 할당합니다.

```javascript
const x = 5;
let y = 3;

x = x + 1; // TypeError: Assignment to constant variable.
y = y + 1; // 4
```

데이터 타입

프로그래밍 언어에는 다양한 데이터를 다룰 수 있는 데이터 타입을 제공합니다. 가장 기본적인 데이터 타입은 다음과 같습니다.

- Integer: 정수형 데이터를 의미합니다.

```
0
-3
1
14
```

- Float: 실수형 데이터를 의미합니다.

```
3.14
2.0
-10.45
0.0
```

- String: 문자형 데이터를 의미합니다.

```
"3"
"I am Iron Man"
"Hello, world!"
```

- DateTime: 년도/월/일/시간/분/초/밀리초로 이루어진 데이터를 의미합니다.

```
"2019-12-30 00:00:00"
```

- Boolean: 참, 거짓 중 하나를 나타내는 데이터를 의미합니다.

```
# python
True
False
```

```
// JS
true
false
```

- Array: 데이터들의 배열을 의미합니다.

```
[1, 2, 3, 4]
["I", "am", "Iron", "Man"]
[1, "two", 3, "four"]
```

Python과 JavaScript의 indexing은 0부터 시작합니다. (R, octave의 indexing은 1부터 시작합니다.)

```
array[0] # 첫 번째 값을 가져옵니다.
array[1] # 두 번째 값을 가져옵니다.
array[-1] # 뒤에서 첫 번째 값을 가져옵니다.
array[1:5] # 두 번째에서 네 번재까지의 값을 가져옵니다!
```

- Dictionary: 데이터의 값들만 저장하는 array와는 달리 key와 value를 함께 저장합니다.

```
{
  "Bob": 3,
  "Stuart": 1,
  "Jesse": 2
}
```

Dictionary의 value를 가져오는 방법은 다음과 같습니다.

```
students["Bob"] # 3
students["Jesse"] # 2
```

비교 연산자

비교 연산자는 무엇이 더 큰지, 작은지, 같은지 등을 비교한 후 boolean 값을 return합니다.

```
# python
3 > 2 # True
3 >= 3 # True
4 < 2 # False
5 <= 13 # True
3 == 3 # True

x = 3 == 3
print(x) # True
```

javascript에는 `==` 비교와 `===` 비교가 있습니다. `==` 는 쓰지 마세요.

```
// JS
3 > 2 // true
3 >= 3 //true
4 < 2 // false
3 <= 13 // true
3 === 3 // true
```

단, float 데이터는 웬만하면 서로 비교하지 마세요. 버그에 취약합니다!

```
4/2 == 2 # true?
```

조건문

특정 조건을 만족할 때(특정 연산의 결과값이 참일 때)와 그렇지 않을 때 각각 다른 코드를 실행시켜야 할 때가 아주 많습니다. 조건문은 if, else if, else 블럭으로 구성되는데, 원칙은 다음과 같습니다. (Python과 JS의 문법이 꽤 다릅니다.)

- If 뒤의 컨디션을 만족시키면, 해당하는 코드 블럭을 실행합니다.
- 조건을 만족시키지 않으면, 다음 else if 조건문을 평가하고, 조건문이 참이면 해당 코드 블럭을 실행합니다.
- 만족하는 조건이 하나도 없으면 else 코드 블럭을 실행합니다. 없으면 그냥 넘어갑니다.

예컨대 놀이 기구 탑승 시 키가 130 이상이면 "탑승 가능합니다", 130 미만이면 "탑승 불가합니다"라는 문구를 보여줘야 한다고 합시다.

```
# python
height = 160

if height >= 130:
  print("탑승 가능합니다.")
else:
  print("탑승 불가합니다.")
```

```
// JS
const height = 160;

if (height >= 130) {
  console.log("탑승 가능합니다.")
} else {
  console.log("탑승 불가합니다.")
}
```

이번엔 조금 더 복잡한 상황을 상정합시다. 키가 만약 190 이상이면 또 탑승 불가능하다고 합시다.

```
# python
height = 150

if height >= 190:
```

```
    print("탑승 불가합니다.")
elif height < 130:
    print("탑승 불가합니다.")
else:
    print("탑승 가능합니다.")

# equivalent

if height >= 190 or height < 130:
    print("탑승 불가합니다.")
else:
    print("탑승 가능합니다.")
```

```
// JS
const height = 150;

if (height >= 190) {
    console.log("탑승 불가합니다.")
} else if (height < 130) {
    console.log("탑승 불가합니다.")
} else {
    print("탑승 가능합니다.")
}
```

루프(Loop)

여러 개의 데이터 등에 대해 같은 코드를 반복적으로 실행하고 싶을 때, 특정 조건을 만족할 때까지 코드를 반복적으로 실행하고 싶을 때 등 반복적으로 같은 코드를 실행하려 할 때 loop를 사용합니다. 루프에는 크게 for loop과 while loop이 있습니다.

- for loop은 주어진 횟수만큼 코드를 실행합니다.
- while loop은 특정 조건을 만족할 때까지 코드를 실행합니다.

For Loop

for loop에서는 주어진 array의 값들을 순서대로 하나씩 꺼내어 변수 i 에 할당한 후 코드 블럭을 실행합니다. 기본적인 문법은 다음과 같습니다.

```python
# python
for i in array:
  # 첫 번째 루프에서 i = array[0]
  # do something with i or not
```

JavaScript에서는 python과 달리 array를 for loop에 집어넣을 경우 array의 element가 아닌 index를 `i` 에 할당합니다.

```javascript
// JS
for (i in array) {
  // 첫 번째 루프에서 i = 0
  // do something with i or not
}
```

python에서 `range(0, 10)` 는 0에서 9까지의 정수를 담은 array을 return합니다. 또 `len(array)` 는 array의 길이를 return합니다. 두 함수 모두 for loop에서 굉장히 유용하게 사용되는 함수입니다.

```python
# python
li = ["one", "two", "three"]

# len(li) == 3
# range(n) 는 range(0, n)와 같습니다. Argument를 하나만 건네면 첫 번째 arg를 0으로 가정합니다.
for i in range(len(li)):
  print(li[i])

# print(li[0]): "one"
# print(li[1]): "two"
# print(li[2]): "three"
```

javascript에서는 range같은 함수는 없지만, 다음과 같이 쓸 수 있습니다. `let` 을 사용했음에 주의하세요!

```
// JS
const li = ["one", "two", "three"]

// li.length == 3
// i++ 는 i = i + 1 의 shortcut입니다.
for (let i = 0; i < li.length; i++ ) {
  console.log(li[i])
}

// console.log(li[0])
// console.log(li[1])
// console.log(li[2])
```

이제 조금 더 복잡한 예시를 들어 보겠습니다. 예컨대 학생의 이름과 점수를 담고 있는 dictionary를 가지고, 각 학생의 학점을 출력해주고 싶다고 합시다.

- dictionary를 for loop으로 돌리면 `i` 에는 key 값이 할당됩니다.
- dictionary의 value를 얻어내는 방법은 dictionary[key]입니다.

```
# python
students = {
  "Jesse": 100,
  "Bob": 80,
  "John": 77,
  "Sonia": 63,
  "Alex": 0,
}

for student in students:
  # 첫 번째 루프에서 student = "Jesse" 가 할당됩니다.
  score = students[student] # students["Jesse"]

  if score >= 90:
    print(f"{student} is A")
  elif score >= 80:
    print(f"{student} is B")
  elif score >= 70:
    print(f"{student} is C")
  elif score >= 60:
    print(f"{student} is D")
  else:
```

```
        print(f"{student} is F")
```

JavaScript에서도 dictionary를 for loop에 집어넣으면 key를 할당해 줍니다.

```
// JS
const students = {
  "Jesse": 100,
  "Bob": 80,
  "John": 77,
  "Sonia": 63,
  "Alex": 0,
};

for (student in students) {
  score = students[student];

  if (score >= 90) {
    console.log(student, "is A")
  } else if (score >= 80) {
    console.log(student, "is B")
  } else if (score >= 70) {
    console.log(student, "is C")
  } else if (score >= 60) {
    console.log(student, "is D")
  } else {
    console.log(student, "is F")
  }
}
```

While Loop

while loop에서는 조건문이 만족될 때까지 코드를 반복적으로 실행합니다.

```python
# python
i = 0

while i < 5:
  print(i)
  i = i + 1

# 0
# 1
# 2
# 3
# 4
```

javascript에서는 `const` 가 아닌 `let` 을 써야 함에 주의하세요!

```javascript
// JS
let i = 0;

while (i < 5) {
  console.log(i);
  i++; // i = i + 1;
}

// equivalent

for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

무한 루프를 만들 수 있습니다!

```python
while True:
  print("!!!")
```

`ctrl + c` 입력을 통해 현재 실행 중인 코드를 실행 취소할 수 있습니다.

Break

`break` 를 만나면 loop를 종료합니다.

```
    for i in range(0, 10):
      print(i)
      if i == 2:
        break;

    # 0
    # 1
    # 2
```

```
    i = 0

    while True:
      if
      print(i)
```

Continue

`continue` 를 만나면 현재 루프를 종료하고 다음 루프로 넘어갑니다. JavaScript도 동일합니다.

```
    for i in range(0, 5):
      if i == 3:
        continue
      print(i)

    # 0
    # 1
    # 2
    # 4
```

Function

프로그래밍의 강력한 원칙 중의 하나는 DRY(Do not Repeat Yourself)입니다. 예컨대 위에서 구현했던 성적을 공표(?)하는 코드를 사용할 때마다 매번 짜는 것은 매우 비효율적입니다.

이런 상황에서 사용하는 것이 함수입니다. 함수는 크게 세 부분으로 구성됩니다.

- Name: 함수의 이름입니다. 추후에 함수를 호출하기 위해 사용합니다.
- Parameter: 함수가 출력물을 계산하기 위해 입력받는 인자들입니다.
- Statement: 함수가 출력물을 계산하는 논리입니다.
- Return: 함수의 출력물입니다.

예를 들어 x를 y로 나눈 나머지를 구하는 함수를 다음과 같이 직접 구현해볼 수 있습니다.

```python
# Python
def remainder(x, y):
  # x가 y보다 작아질 때까지 y의 값을 계속해서 뺍니다.
  while x >= y:
    x = x - y
  # x가 y보다 작아졌다면, 그 값을 return 합니다.
  return x

# 정의한 함수를 호출합니다.
remainder(3, 2) # 1
remainder(4, 1) # 0
```

```javascript
// JS
function remainder(x, y) {
  while (x >= y) {
    x = x - y;
  }
  return x;
}

remainder(3, 2) // 1
remainder(4, 1) // 0
```

return 값은 어떤 값이든 될 수 있습니다.

특히 최신 버전의 javascript에는 arrow function이란 것이 있습니다.

- 함수를 다른 함수에서 호출할 때, binding 등의 문제가 발생합니다. Arrow function은 이러한 문제를 해결해줍니다.
- 일회용 함수(instant function)을 구현할 수 있습니다.

- ...

어쨌거나 최신에 도입된 만큼 여러 가지 이점을 가집니다. 그래서 저는 웬만하면 arrow function을 애용합니다.

Instant function은 다른 함수에 인자로 전달하는 등의 목적을 위해 사용되는데, 수업을 진행하다가 사용할 일이 있으면 그 때 설명하겠습니다. 참고로 python에도 instant function을 구현할 수 있는데, lambda라는 친구를 이용합니다.

```
const remainder = (x, y) => {
  while (x >= y) {
    x = x - y;
  }
  return x;
}
```

```
const plus = (x, y) => (x + y) // 바로 return합니다.
```

Class

Class를 통해 나만의 자료형을 만들 수 있습니다. Class를 통해 정의된 자료형은 다른 자료형들과 같이 고유의 method(함수)를 가질 수 있습니다. 나중에 Django를 통해 Model을 만들 때 사용하게 될 겁니다.

```
# python
class Student:
  # 생성자입니다. Object를 생성할 때 호출됩니다.
  def __init__(self, name):
    self.name = name

  # 고유의 method입니다.
  def greeting(self):
    print(f"Hello, {self.name}!")

jesse = Student("jesse")
jesse.greeting() # Hello, jesse!
```

```
// JS
class Student {
    constructor(name) {
        this.name = name;
    }

    greeting = () => {
        console.log(`Hello, ${this.name}!`);
    }
}

const jesse = new Student("jesse");
jesse.greeting(); // Hello, jesse!
```

Using Library

Python과 JavaScript의 가장 큰 장점 중 하나가 바로 풍부한 라이브러리의 존재입니다. 개중에는 기본적으로 포함된 build-in 라이브러리도 있고, 세간의 훌륭한 개발자들이 만들어 낸 외부 라이브러리도 있습니다. 실제로 개발할 때는 모든 것을 직접 개발하는 것보다, 훌륭한 라이브러리를 적절히 튜닝하여 사용하는 것을 강력히 추천합니다.

Built-in Library

Built-in library의 경우에는 그냥 가져다가 쓰면 됩니다. 다음은 python의 내장 라이브러리인 `random` 라이브러리의 일정 범위 내의 임의의 정수를 생성하는 함수인 `randrange()` 의 사용 예시입니다.

```
# python
import random # 라이브러리를 import합니다.

random.randrange(0, 10) # 0에서 10까지의 임의의 정수를 생성합니다.
```

또한 python의 numpy는 이런 저런 연산들을 매우 효율적으로 처리해주는 대표적인 라이브러리 중 하나입니다.

External Library

필요한 기능을 구글링하면 적절한 라이브러리들을 거의 항상 찾아낼 수 있습니다. 좋은 라이브러리를 찾아냈다면, 다음과 같이 해당 라이브러리를 우선 설치해야 합니다. 예시: https://blog.jesse.kim

```
# python
pip3 install LIBRARY_NAME
```

```
// JS
npm install LIBRARY_NAME
```

그리고 위와 같이 import하여 사용하면 됩니다. 외부 라이브러리의 사용은 추후에 많이 다룰 것 같습니다.

과제 1

이제 이론적으로 프로그래밍 언어를 통해 거의 모든 것을 구현해낼 수 있는 경지에 다다랐습니다! 그런 의미에서 첫 과제는 바로 python을 이용해 "숫자 맞추기 게임" 만들기입니다. 이 과제의 목표는 loop와 conditional statement에 조금 더 친해지게 되는 것입니다.

먼저 콘솔을 통해 사용자의 입력을 받을 수 있는 `input()` 함수에 대해 소개하겠습니다. 다음과 같은 코드를 유저의 입력을 기다리고, 입력된 값을 `input_value` 에 저장합니다.

```
input_value = input("숫자를 입력하세요: ")

# 숫자를 입력하세요:
```

단 입력된 값은 `string` 데이터 타입으로 저장됩니다. 이 것을 `integer` 타입으로 바꾸기 위해서는 다음과 같은 코드가 필요합니다.

```
input_integer_value = int(input_value)
```

이제 다음과 같이 작동하는 python script를 하나 만들어 보세요.

- 1~100 사이의 임의의 정수를 맞추는 게임입니다. (Hint: random 라이브러리를 사용하세요)
- 맞출 때까지 계속해서 입력을 받습니다.
- 맞추지 못했을 경우, 정답에 대한 힌트를 제공합니다.

```
> python random_number.py

숫자를 입력하세요: 25
너무 낮습니다. 다시 입력하세요!
```

```
숫자를 입력하세요: 60
너무 낮습니다. 다시 입력하세요!

숫자를 입력하세요: 80
너무 낮습니다. 다시 입력하세요!

숫자를 입력하세요: 90
너무 높습니다. 다시 입력하세요!

숫자를 입력하세요: 83
너무 높습니다. 다시 입력하세요!

숫자를 입력하세요: 81
정답입니다!
```

과제 2

이번 과제에서는 function과 친해져 봅시다. 어떤 숫자(x)가 어떤 숫자(y)의 배수인지, 즉 y가 x의 인수인지 아닌지를 판단하는 함수를 만들게 되겠습니다.

- 먼저 두 숫자 값을 입력받아, 첫 번째 숫자가 두 번째 숫자의 배수인지를 판단하여 boolean 값을 return하는 함수를 만듭니다.
- 그 후 콘솔을 통해 유저로부터 두 숫자를 차례대로 입력받습니다.
- 두 숫자에 대한 함수값을 콘솔에 출력합니다.
- 단, 두 번째 숫자가 첫 번째 숫자보다 크다면 경고 메시지를 출력해 주세요.

```
> python is_multiple.py

첫 번째 숫자를 입력하세요: 6
두 번째 숫자를 입력하세요: 3

True
```