Git config --global user.name “yerim”

Git config --global user.email [yerim2300@gmail.com](mailto:yerim2300@gmail.com)

Git init

Git status

Git add .

Git status

Git commit -m “first commit”

Git log –gragh

Git branch -M main

Git remote add origin 주소

Git remote -v

Git push -u origin main

[Git remote get-url origin 주소] 주소 변경

Git add .

Git commit -m “first commit”

Git push

파이썬 특징

* 인터프리터 방식이다
* 인코딩이 UTF-8이다 (규격)
* 인코딩이 무엇인가요 -> 규격대로 맴핑하는 작업
* ASCII(아스키)영어만 가능, UTF-8
* 세미콜론은 옵션으로 써도되고 안써도 된다.
* \*\*들여쓰기 (중요)
* 띄어쓰기를 통해 코드의 범위를 구분한다.
* 범위를 나눌 때 띄어쓰기 4회를 권장하고 있습니다.(가독성과 유지보수성을 위해 4번)탭x
* 주석을 많이 달자!

Jypter lab

* A -> 밑에 새로운 셀 추가
* B -> 위에 새로운 셀 추가
* D + D => 현재셀 삭제

Edit Mode

* 컨 + /키

리터럴은 값 자체이다.

2023.02.14(화)

어제 배운내용 복습

Jupyter Lab의 단축키

1. **ESC**를 눌러 커맨드 모드로 진입하여 **ENTER**를 통해 cell 을 수정할 수 있습니다. 아래 커맨드는 커맨드 모드에서 동작합니다.
2. **A**는 현재 cell 위에 새로운 cell 을 추가합니다.
3. **B** 는 현재 cell 밑에 새로운 cell 을 추가합니다.
4. **D + D** D를 연속해서 두번 누르면 현재 cell 을 삭제합니다.
5. **M** 은 Markdown 셀로 변환하며, **Y** 는 Code 셀로 변환하고  **R** 은 Raw Cell 로 변환합니다.
6. **CTRL + B**화면을 더 크게 사용할 수 있습니다. 왼쪽 파일 탐색기가 사라집니다.
7. **SHIFT + M**두 개의 셀을 한개의 셀로 Merge 합니다.
8. **CTRL + SHIFT + –** 현재 커서 위치를 기준으로 두 개의 셀로 구분합니다.
9. **SHIFT+J** or **SHIFT + DOWN** 현재 셀에서 아래쪽 위치로 새로운 셀을 같이 선택합니다.
10. **SHIFT + K** or **SHIFT + UP** 현재 셀에서 위쪽 위치로 새로운 셀을 같이 선택합니다.
11. **CTRL +**/ 선택한 코드를 주석처리합니다.

기타 잘 알려지지 않은 단축키

1. **CTRL+D:**한줄 삭제하는 단축키입니다.
2. **CTRL+[ or CTRL+]:**단체 indentation / Tab과 Shift+Tab 으로도 가능하지만, text editor 에서는 Shift+tab 이 안먹혀서 CTRL+[ 을 쓰면 유용합니다.

8진수표기/16진수표기/2진수표기

변수 이름은 원하는 대로 지으면 되지만 다음과 같은 규칙을 지켜야 합니다.

* 영문 문자와 숫자를 사용할 수 있습니다.
* 대소문자를 구분합니다.
* 문자부터 시작해야 하며 숫자부터 시작하면 안 됩니다.
* \_(밑줄 문자)로 시작할 수 있습니다.
* 특수 문자(+, -, \*, /, $, @, &, % 등)는 사용할 수 없습니다.
* 파이썬의 키워드(if, for, while, and, or 등)는 사용할 수 없습니다.

변수 다뤄보기

빈 값 할당 None (=null)

비교연산자

삼중따옴표를 사용

s3 = """

안녕하세요.

나는 파이썬을 공부하고 있습니다.

반갑습니다.

"""

print(s3)

s3 = """\

안녕하세요.

나는 파이썬을 공부하고 있습니다.

반갑습니다.\

"""

print(s3)

이렇게 하면 처음과 끝은 개행x

인덱스

범위를 넘어가면 인덱스 에러가 생긴다

문자열 변경 불가 예제 (인덱스)

문자열은 불변(immutable)의 특징을 갖는다. 문자열은 인덱스로 접근해서 값 변경 불가하다.

리스트 (list)

파이썬 언어에서 가장 융통성 있는 자료형중 하나

가변(mutable)의 특성을 갖기에 수정도 용이하다.

파이썬의 리스트는 다른 언어의 배열(Array)과 차이가 있다.

수학에서의 함수

리스트변수.append() -> .append는 메서드

Append는 요소 추가하기

Insert는 리스트 중간에 요소 추가하기

extend()

한번에 여러 요소를 추가하고 싶을때는 extend()함수 사용

인덱스로 제거하기 : del키워드, pop()

* Del 리스트명[인덱스]
* 리스트명.pop(인덱스)

값으로 제거하기 : remove()

* 인자와 리스트에 동일한 요소 제거(제일 앞에 있는 값)

모두 제거하기 : clear()

* 리스트 내부의 요소를 모두 제거
* 리스트.clear()

리스트 정렬하기 : sort()

* 리스트.sort()
* 오름차순으로 정렬

.index() -> 인덱스번호

.count() -> 인자와 같은 요소의 개수

.reverse() -> 리스트 요소를 반대 방향으로 재정렬

리스트를 skack처럼 활용하기 (후입선출 Last In First Out)

Append / pop

튜플 : 함수와 함께 많이 사용되는 리스트와 비슷한 자료형/한번 결정된 요소는 바꿀수 없다.

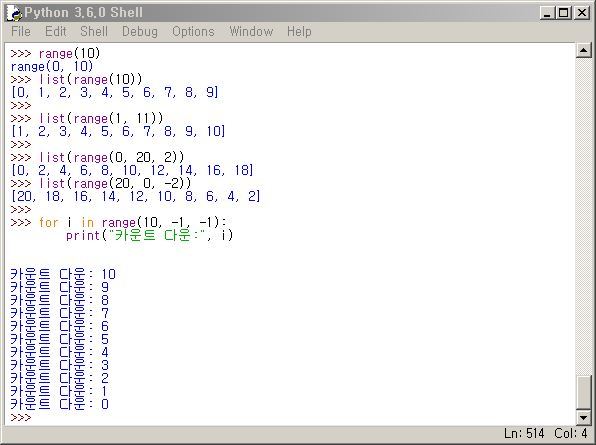
람다. 1회용 함수를 만들어야 할 때 많이 사용

1. 처음 20번째까지의 소수 [튜플] 소수는 정해져 있기 때문에 튜플
2. 프로그래밍 언어의 종류 [튜플] 프로그래밍 언어들은 이름이 바뀌지 않아 튜플 가능
3. 사람의 나이, 몸무게, 키 [리스트] 나이, 몸무게, 키는 변할 수 있다.
4. 사람의 생일과 출생지[튜플] 사람의 생일과 출생지는 바뀌지 않는다.
5. 롤 게임의 결과[튜플], 결과는 나온 대로 라서 바꾸지 않아도 된다
6. 잇단 롤 게임의 결과[리스트], 최신결과라면 전의 데이터가 사라져야 하기 때문에



<https://data-make.tistory.com/102>

리스트 메서드



range

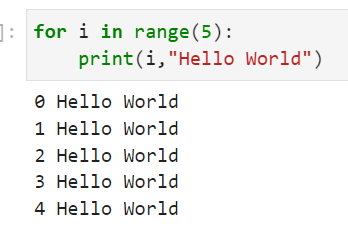
2023년 02월 15일(수)

복습

2023년 02월 16일(목)

Interable:반복가능한

파이썬에서의 for문은 다른 프로그래밍 언어와 사뭇 다릅니다



input\_value2 = int(input("자연수 하나 입력해주세요:"))

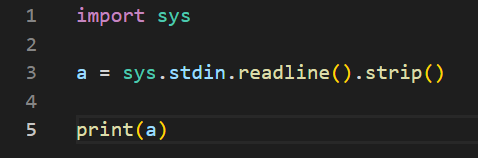
for i in range(input\_value2,0,-1):

for j in range(i) :

print(i, end = '\t')

print()

Range() 안의 숫자 만큼 반복



Input 대신 사용하는 sys.stdin.readline!!!

<https://mgyo.tistory.com/166>

2023년02월17일

딕셔너리 자료형

Key: value -> key-value pair

중괄호 사용 {}

Dic\_var = { “key” : “value”}

Print(type(dic\_var)) -> <class ‘dict’>

Print(dic\_var) -> {‘key’ : ‘value’}

Key-value pair는 각각 콤마로 구분자어 요소로 등록할 수 있습니다

Dictionary\_variable = {key : value, key:value, …}

() -> tuple 시퀀스 (인덱스 O)

[] -> list 시퀀스 (인덱스 O)

{} -> dictionary(key:value),set(key) 비시퀀스(인덱스 X)

key값이 고유하게끔 만든 것을 딕셔너리라고 한다.

딕셔너리 자료형에서 키는 문자열, 정수,실수,불,튜플을 사용

키에 이 자료형을 섞어서 사용해도 된다.

단, 키에는 리스트, 딕셔너리, 집합을 사용할 수 없다

키는 고유한 값을 가져야 한다.(인덱스에 겹치는 숫자가 업듯)

Print( list( map( type, dic\_var.keys()))

* + [<class ‘str’>,<class ‘int’>,<class ‘float’>,<class ‘tuple’>]

Empty\_dict = {}

Empty\_dict = dict()

Set()

List[]

Tuple()

dict([("key","value"),("key2","value2")])

* + Zip

dict\_b = dict(zip(["key","key2","key3"],["val","val2","val3"]))

print(dict\_b)

* + {'key': 'val', 'key2': 'val2', 'key3': 'val3'}

[\_ for \_ in zip(["key","key2","key3"],["val","val2","val3"])]

* + [('key', 'val'), ('key2', 'val2'), ('key3', 'val3')]

dict\_b = dict(zip(["key","key2","key3"],["val","val2","val3"]))

print(dict\_b,end="\n\n")

dict\_b['key'] = "super value"

print(dict\_b)

특정키가 존재하는지 확인할 때 in연산자를 활용

Print(‘key’ in dict\_b)

* 딕셔너리 설정
  + dict([("key","value"),("key2","value2")])
  + [\_ for \_ in zip(["key","key2","key3"],["val","val2","val3"])]
  + dict\_b = dict(zip(["key","key2","key3"],["val","val2","val3"]))

keys = ("key1","key2","key3")

val = ("val1","val2","val3")

sum = []

for index in range(len(keys)) :

sum.append((keys[index],val[index]))

print(sum)

* 딕셔너리 추가

dict\_b['key'] = "super value"

* 딕셔너리 삭제

Del dict\_b[‘new one’]

* 딕셔너리 확인
  + print('key' in dict\_b)
  + print('val2' in dict\_b.values())
  + len(dict\_b)

iterable형태면 zip으로 묶어서 dict로 만들어 줘도 된다.

딕셔너리 자료형

* 비시퀀스 자료형 ( 순서가 없음 -> 인덱스 X)
* 고유한key를 갖는다.
* 문자,\_,숫자 -> 인자 키워드 부분

keys = input("이름을 기입해주세요").split(" ")

value = input("몸무게를 입력해주세요.").split(" ")

people = dict(zip(keys,list(map(float,value))))

print(people)

리스트 생성 방법 – 대괄호 사용

* 대괄호 안에 리스트를 구성할 요소 작성
* 콤마로 구분
* list클래스로 만들어진 객체이다.
* 리스트는 반복되는 객체인 iterable을 인자로 하는 list()함수를 통해 생성할 수도 있습니다.
* 문자열.split() -> list의 형태로 반환

리스트 컴프리헨션(comprehension)

Expression for I in iterable

* + List(range(10)) = iterable
  + [i\*\*2 for I in list(range(10))]
    1. [0,1,4,9,16,…,81] 제곱리스트
* 반복가능한 객체 (iterable)로부터 값을 하나 꺼내와서 변수
* 변수로 표현식 실행 -> 결과 리스트 요소에 넣음 ->반복
* chr(i) for i in range(ord('a'),ord('z')+1 -> a부터 z까지
* prac\_result2 = [chr(i) for i in range(97,123)]

id 내장함수

* 고유한 정수 값(메모리 주소)를 갖는다.
* Id()로 가져온 정수 값이 같다면 같은 객체
* For I,v in enumerate(prac\_result5) :
  + Print(“prac\_result5[“,I,”]:”, id(v),sep=””)

기존 리스트의 전체 범위의 슬라이싱 결과는 새로운 객체이다.

리스트 변수명[:]으로 전체를 동일하게 가져오는, 슬라이싱으로 반환받는 리스트 객체는 다른객체이다.

old\_list = list(range(10))

new\_list = old\_list[:]

print(id(old\_list))

print(id(new\_list))

print(old\_list is new\_list) #객체의 비교

print(old\_list == new\_list) #값의 비교

ord() : 문자의 아스키 코드값을 리턴하는 함수이다.

객체 간의 비교에서 is 연산자 #객체의 비교

객체 간의 비교에서 == 연산자 #값의 비교

차이

chr() : 아스키 코드값 입력으로 받아 그 코드에 해당하는 문자를 출력하는 함수이다.

(참고) tuple, str은 is연산자의 결과가 다를 수 있다.

* + Tuple은 불뭐…?
  + 뭐더라

리스트 컴프리헨션(comprehension)

대괄호 대신 list()함수를 사용한다.

* + 성능도 대괄호 방식이 더 좋다

리스트 복사 (copy() 메서드)

* + 리스트.copy()메서드는 리스트 객체를 새롭게 복사
  + 유의사항
    1. 객체 자체를 새로운 메모리 공간에 복사
    2. 슬라이싱과 결과가 같다.
* 얕은 복사(shallow copy)
  + 리스트 내에 있는 값을 복사
  + 메모리 주소를 담고 있는 참조 변수가 있으면 객체주소 복사
  + 리스트에 애스터리스크(\*)를 사용하면 얕은 복사
  + List\_i = [[0]]\*3
  + list\_i[0][0] = 999 , print(list\_i)

[[999], [999], [999]]

* + 리스트 내에 있는 모든 요소가 같은 객체
  + 단 하나의 요소만 바꿨을 뿐인데 모든 요소의 값이 변경
  + \*뿐만 아니라 슬라이싱, copy()메서드를 통한 복사도 얕복
* 깊은 복사 (Deep copy)
  + Import copy
    - Deepcopy()를 위해 copy 라이브러리를 필요로 합니다.
    - 얕은 복사와달리 객체도 새롭게 복사한다.