|  |  |
| --- | --- |
| 교육 제목 | python 기초 92~159p |
| 교육 일시 | 2021.09.17 |
| 교육 장소 | 영우글로벌러닝 교육장 |
| **교육 내용** | |
| 오전 | format() 함수  "{}".format() 중괄호 포함하여 마침표를 찍고 사용  "{}".format(31)  "{}, {}".format(31, 5) -> 중괄호 개수와 포맷의 매개변수 개수가 같아야함  \*{}의 개수가 매개변수보다 많으면 index error 출력 반대는 상관없음  ex) "{}, {}, {}".fornat(34, 15) x, "{}, {}, {}".fornat(34, 15, 18, 21) o  format의 다양한 기능  정수를 특정칸에 출력 : {:d} / 기호 붙여 출력 {+d} / 부동소수점 출력 {:f}  소수점 제거 {:g}  upper() - 문자열을 모두 대문자로 만듦  예) a = "Hello world" / a.upper() -> HELLO WORLD  lower() - 문자열을 모두 소문자로 만듦  strip() - 문자열 양옆 공백 제거  lstrip() - 문자열 왼쪽 공백 제거  rstrip() - 문자열 오른쪽 공백 제거  find() - 왼쪽부터 처음 등장하는 위치 찾음  rfind() - 왼쪽부터 처음 등장하는 위치 찾음  예)"헬로헬로헬로우".find("헬로") -> 0 / "헬로헬로헬로우".rfind("헬로") -> 4  \*문자열의 첫번째 자리는 항상 0  in 연산자 - 문자열 내에 어떤 문자열이 있는지 확인 True/False로 나타남  ex) print("안녕" in "안녕하세요") -> True / print("잘가" in "안녕하세요") -> False  split() - 문자열을 특정한 문자로 자름 |
| 오후 | Bool 자료형 - True 혹은 False로만 표현됨  불 비교 연산자  == : 같다 / != : 다르다 / < : 작다 / > : 크다 / <= : 작거나 같다 / >= : 크거나 같다  불 논리 연산자  not : 아니다 - bool을 반대로 전환  and : 그리고 - 모두 참일때 True, 외에는 False  or : 또는 - 두개 중 하나만 참이라도 True, 모두 거짓이면 False  if 조건문 : 조건에 따라 코드를 실행거나 실행하지않게 하고 싶을때 사용  표현식 뒤에 콜론이 붙어야함, 다음 문장은 4칸 들여쓰기 후 입력  예) if 표현식 :  참일때 내용  \*\* 4칸 들여쓰기는 tap키로 쉽게 쓸수 있음, 없앨땐 shift+tap  if~else 구문 : if 문장으로 참일때 출력될 값과 else 문장으로 거일때 출력될 값을 표현할때 씀  예)  if 표현식 :  참일때 내용  else:  거짓일때 내용  elif 구문 : 조건 비교가 많아질때 사용 여러 조건을 나타낼수 있음  if 조건1 :  참일때 내용  elif 조건2:  참일때 내용  elif 조건3:  참일때 내용  else  거짓일 때 내용  날짜/시간 활용  datetime 패키지 사용  import 패키지명 as 약어 - 약어로 축약해서 쓰면 좋음  패키지명.클래스명.함수명 or 패키지명.함수명  예)import datetime as dt  now = dt.datetime.now()  print(now.hour, "시", now.minute, "분")  list 반복문 : 여러가지 자료를 저장할 수 있는 자료  식별자 = [요소1, 요소2, 요소3, 요소4]  대괄호 안에 여러 요소를 넣어 표현  문자열, 숫자로 여러 자료형으로 표현 가능  순서는 항상 왼쪽부터 번째로 시작함  음수로 오른쪽부터 셀때는 -1부터 시작  리스트 연산자  " + " =연결, " \* " = 반복, len() = 데이터 개수(길이)  요소추가  리스트명.append(요소) - 맨뒤에 요소 추가  리스트명.insert(위치, 요소) - 원하는 위치에 요소 추가  리스트명.extend() - 리스트 연산자 +와 같이 리스트와 리스트를 연결 시키는 기능  하지만 연산자 +는 비파괴적이고 extend 파괴적 함수임  \*\*\*파괴적 함수는 원본을 유지 시킬수 있어 원본과 수정본을 유연하게 사용할 수 있고 파괴적 함수는 원본이 없어지고 수정본만 남는 개념(대부분 비파괴적 함수가 쓰임이 좋지만 리스트처럼 크게 리소스를 잡아먹을 때 최적화를 위해 비파괴적 함수가 더 유리 할수 있다)  요소제거  인덱스로 제거  del 연산자 - del 리스트명[인덱스]  pop() - 리스트명.pop(인덱스) \*\*인덱스를 적지 않으면 맨뒤부터 제거(-1)  값으로 제거 - 리스트명.remove(값) \*\* 값이 중복될때는 맨 왼쪽만 제거  모두제거 - 리스트명.clear()  in/not in 연산자 - 리스트 내부에 값이 있는지 확인  값 in 리스트명 / 값 not in 리스트명  for 반복문  for 반복자 in 반복할 수 있는 것 :  -> for <요소 변수명> in 리스트명 : |
|  |  |