[나도 코딩 파이썬 강의]

1. 파이썬 환경 설정

파이썬 다운받고, 폴더 설정해서 파일만들기. 그리고 run and debug 환경 만들기

2. 숫자 자료형, 문자 자료형, Boolean(참, 거짓), 논리 연산자

ctrl+s = 저장단축키

코드실행 = 상단 debug, start debugging, 오른쪽 맨 위의 초록 세모 클릭. 등등

\*\*: 제곱 %: 나머지 구하기, %%: 몫 구하기 3==3 -> true, 3+4==2 -> false

!= : 같지 않다. & -> and 조건 | -> or 조건

+= 2: 기존 변수에 2를 더한뒤에 저장해라.

“”” 문자열 “””” : 이렇게 하면 줄바꿈 문자열도 다 쓸 수 잇다.

3. 변수 할당하기

ex. 우리집 강아지 이름은 연탄이다. 연탄이는 4살이고 산책을 좋아한다. 연탄이는 어른이다.

이 경우 animal= “연탄” 이런식으로 할당해주면, 다른 사람이 코드를 가져가서 쓸 때, 변수만 바꿔서 쓸 수 있다!!

\* 단 이 때 문장에는 +animal+ 이렇게 넣어줘야하고 ,를 쓸 경우에는 띄어쓰기가 하나 들어가게 됨.

(+가 이어주는 거라고 생각하면 된다.)

코드 윗줄에서 hobby에 공놀이라고 할당해도 밑 문장에서 다시 낮잠으로 설정하면 출력 때 낮잠으로 출력됨.

# 주석 처리

‘’’ ~ ‘’’ 이렇게 하면됨. 혹은 드래그하고 나서 ctrl+/ 하면 됨.

\* font size 처리: file -> preference -> terminal font 들어가서 사이즈 바꾸면 됨.

4. 숫자 처리 함수

abs(-5) : 5 반환 pow(4,2) : 4의 2승 반환 max(1,2,4,10) : 10반환 ,min 마찬가지 round(3.14) : 반올림

5. 랜덤 함수 (무작위로 숫자를 뽑아줌)

from random import \*

print(random()) : 0~ 1.0 미만 임의 값 생성 randint(1,45) : 1부터 45이하 사이에서 랜덤추출

quiz.

월4회 스터디 하는데 3번은 온라인, 1번은 오프라인으로 약속/ 아래 조건에 맞는 오프라인 모임 날짜를 정해주는 프로그램을 짜라.

1. 랜덤으로 날짜 선택 2. 월 최대 28일로 정해라 3. 매월 1~3일은 스터디 준비를 해야하므로 제외

from random import\*

x=randint(4,28)

print("월별 스터디 모임 날짜는 매월 "+str(x)+" 일로 선정되었습니다.")

print는 string만 출력하기 때문에 숫자는 string화 시켜서 내보내줘야 한다!!

6. 슬라이싱: r에서 배웠듯이 필요한 것만 추출하는 법 (이 때 시작은 0부터 카운트)

[0:2] : 0부터 2직전까지 추출해라! (2번째 포함아닌 거 주의, 이 때 0은 생략가능함)

[7: ] : 8번째부터 끝까지 추출해라 [-7: ] : 맨 뒤에서 7번째부터 끝까지 추출해라

7. 문자열 처리 함수

\* variable, vari는 변수를 뜻함 \*

variable.lower() : 소문자전환 vari.upper(): 대문자전환 vari.[0].isupper() : 변수의 첫째자리가 대문자냐??

len(variable) : 문자열의 길이를 알려준다.

var.replace(“python”, “Java”) : var변수 안에 있는 python을 Java라는 단어로 바꿔라!

var.index(“n”) : var 변수에서 n이라는 문자를 찾아서 몇 번 째인지 알려줘라!

var.find(“n”): var 변수에서 n이라는 문자를 찾아서 알려줘라!

var.count(“n”): var 변수에서 n 이라는 문자의 개수를 알려줘라

8. 문자열 포맷

print(“나는 %d살 입니다” % 20) 이렇게 써주면 %에 20 이라는 수가 들어감.

1.

%s : 문자열을 삽입하겠다. ex. print(“나는 %s살 입니다” % ‘20’)

%c” character를 삽입하겠다. (character는 문자열과 달리 한 글자를 지칭함.)

\* %s를 쓰면, 숫자, 문자열, 문자 상관없이 모두 가능!!

print("나는 %s 색과 %s색을 좋아해요" % ("파란", "빨간"))

이러면 두개를 한꺼번에 넣기 가능!!

2.

print("나는 {}살입니다".format(20))

format으로 처리하기!!

print("나는{1}색과 {0}색을 좋아해요" .format("파란", "빨간"))

print("나는{age}살이며 {color}색을 좋아해요" .format(age=20, color="빨간"))

age=20

color="빨간"

print(f"나는 {age}살이며, {color}색을 좋아해요.")

이런식으로도 응용이 가능하다!!

\* 줄바꿈, 탈출문자 팁

# \\: 문장 내에서 \ 구현

# \r: 커서를 맨 앞으로 이동

# \b: 백스페이스 (한 글자 삭제 기능)

# \t: 탭

print("백문이 불여일견, \n 백견이 불여일타")

print("백문이 \"불여일견\", \n 백견이 불여일타")

역슬러시로 큰따옴표 포함 문자를 출력할 수 있게 된다. ex. print(“c:\\:users”)

print("red apple \rPine")

\*\r 을사용해서 뒤에 오는 문자들을 문자열 맨앞으로 뺄 수 있다.

print("Redd\bApple")

print("Red\tApple")

quiz!

사이트 별로 비밀번호를 만들어 주는 프로그램 만들기

ex. <http://naver.com>

규칙1 : //까지 부분 제외

규칙2: 처음 만나는 점 이후 부분은 제외

규칙3: 남은 글자 중 처음 세자리 + 글자 개수 + 글자 내 ‘e’ 개수 + “!” 출력

9. 리스트: 순서를 가지는 객체의 집합, 자료형은 아무거나 다 가능함!!

ex. subway=[10,20,30]

subway.index(20) -> 1 ( 20은 두 번째니까 1이 출력)

subway.append(“하하”) -> 리스트에 추가하는 방법!

subway.insert(1, “정형돈”) -> 리스트 두번째 요소에 정형돈을 추가해라!

subway.pop -> 리스트 맨 마지막에 있는 요소를 제거해서 저장해라

ex. subway.pop() 후에 subway 출력하면 맨 마지막 요소가 없어지고 출력됨.

var.sort() -> var에 할당된 리스트 내 요소들을 순서대로 정렬해라 (오름차순)

var.reverse() -> 내림차순으로 리스트 요소들을 정리해라

var.clear() -> 리스트 요소들 모두 지우기

10. 사전(딕셔너리)

ex. cabinet = {3:” 유재석”} key : value 형태로!

print(cabinet[key]) print(cabinet.get(3))

\*get 함수 경우, key값이 잘못되면 none이 출력, 그리고 그 다음 문장이 실행, 근데 [key]형태는 그거로 바로 종료

\*key 값에는 string도 할 수 있다!

cabinet.get(5, “사용가능”): key 5 값이 있으면 가져오고, 없으면 “사용가능”을 가져와라.

print(3 in cabinet) -> cabinet 딕셔너리에 3이라는 키값에 값이 할당되어있니?? T or Fu 로 출력

#cabinet[“key”] = “ value” : 기존 key에 할당되어있는 값에 value를 넣어서 업데이트 해라 (value 바꾸기!)

# del cabinet[“key”] : 이런식으로 key 삭제 가능. #print(cabinet.key()) : key들만 출력해라! (value도 가능)

#cabinet.items() : key, value 모두 출력 cabinet.clear() : 모두 삭제

11. 튜플: 리스트와 다르게 내용 변경을 할 수 없음. 그래서 변경되지 않는 목록을 할 때 튜플 사용.

menu= ( “value1” , “value2”) : 튜플은 소괄호를 사용해서 할당해줘야됨.

print(menu[0], menu[1]) : 이런식으로 리스트랑 같은 방식으로 출력.

(name, age , hobby) = “ 장지수 “ , 25 , 코딩 “ 이런식으로 할당도 가능!

12. 세트 {집합} : 중복x, 순서없는 집단

my\_set={1,2,3,3,3} : 이 경우 출력하면 1,2,3 만 나옴! 왜냐면 중복되는 문자는 인식하지 않으니까…

# 교집합: print(java & python) -> 두 집합의 공통원소가 출력됨 | java.intersect(python) 이렇게 해도 가능

# 합집합: print(java | python) , print(java.union(python))

#차집합: java-print // java.difference(python) 해도됨

#원소 추가하기 : java.add( )로 추가 #원소 제거하기 java.remove

\*\*\*

menu={1,2,3} 으로 처음에 집합으로 지정해줘도 menu=list(menu) , tuple(menu), set(menu) 이런식으로 자료 구조를 바꿀 수가 있다!!

quiz. 파이썬 코딩대회를 주최하고 이를 위해 댓글 이벤트를 진행

댓글 작성자들 중 추첨을 통해 1명은 치킨, 3명은 커피 쿠폰을 받는다. 추첨 프로그램을 작성하자.

조건1: 댓글은 20명이 작성햇고 아이디는 1~20 이다.

조건2: 무작위 추출이며 중복 당첨은 안됨.

조건3: random module의 shuffle 과 sample을 사용한다.

ex. from random impornt \*

1st= [1,2,3,4,5]

print(1st), shuffle(1st) , print(sample(1st,1)): 셔플이라는 변수에서 랜덤으로 원소한개를 뽑아라!

13. if문

weather= “비”

if weather == “비” :

print(“우산 챙겨라”)

elif weather == “미세먼지” :

print(“마스크를 챙겨라”)

else:

print(“준비물은 필요없어용”)

temp=int(input("기온은 어때요??")) **#인풋은 스트링만 받기 때문에, 사용자 인풋값을 정수화 시켜야됨**

if 30<=temp:

print("너무 더워요. 나가지 마세여")

elif 10 <= temp and temp<30 :

print("좋은 날씨에요")

elif 0<= temp and temp <10 :

print("외투를 챙기세요")

else:

print("너무춰요. 나가지마세요 ")

14. 반복문, for & while

# for 문 for 이하의 코드를 반복해라

# for waiting\_no in range(5): ( 다음 변수에 대하여 아래 문장들을 반복해라 )

# print("대기번호: {0}". format(waiting\_no))

# starbucks= ["아이언맨", "토르", "아이엠 그루트"]

# for customer in starbucks:

# print("{0}님, 커피가 준비되었습니다". format(customer))

# while 문 : 어떤 조건을 만족할 때 까지 명령문을 반복해라

# customer="토르"

# index=5

# while index>=1 : (다음 조건을 만족하는한, 아래 명령문을 반복해라)

# print("{0}, 커피가 준비되었습니다. {1}번 남았어요".format(customer, index))

# index-=1

# if index==0:

# print("커피는 폐기 처분 되었습니다.")

# customer= "아이언맨"

# while True: 아래를 영원히 반복해라!

# print("{0}, 커피가 준비되었습니다. {1}번 남았어요".format(customer, index))

# index+=1

# if index==0:

# print("커피는 폐기처분되었습니다.")

customer="지수"

person= "Unknown"

#스타벅스 상황 연출…

while person != customer :

print("{0}, 커피가 준비되었습니다".format(customer))

person= input("What is your name?")

print("아직 준비가 안되었습니다")

15. continue 와 break

absent=[2,5]

for student in range(1,11):

    if student in absent:

        continue

    print("{0}, 책 읽어~". format(student))

\* continue : 특정 조건에 대해서 for 문이 반복할 명령문을 수행하지 않고, 제일 처음으로 돌아가라!

absent=[2,5]

nobook = [7]

for student in range(1,11):

    if student in absent:

        continue

    elif student in nobook:

        print("오늘 수업 여까지~. {0}는 교무실로 따라오니라".format(student))

        break

    print("{0}, 책을 읽어봐".format(student))

\* break는 for 명령문을 빠져나올 때 사용한다!

16. 한 줄로 끝내는 for 문

리스트 각 요소에 100씩 더하기

students= [1,2,3,4,5]

print(students)

students=[i+100 for i in students]

print(students)

학생 이름을 길이로 변환하기

students=["Iron man", "Thor", "I am groot"]

students=[len(i) for i in students]

print(students)

학생 이름을 대문자로 변환하기

students=[i.upper() for i in students]

print(students)

# quiz4!

당신은 택시 기사다. 50명의 승객과 매칭 기회가 있을 때 총 탑승 승객수를 구하는 프로그램을 만들어라.

조건1: 승객별 운행 소요 시간은 5~50분 사이 난수

조건2: 당신은 소요 시간 5~15분 사이 승객만 매칭해야함

#

from random import\*

passenger\_no=0

for i in range(1,51):

    driving\_time1=randrange(5,51)

    if 5<= driving\_time1 <=15:

        print("[O], {0} 번째 손님 (소요시간 : {1}분) 입니다.".format(i,driving\_time1))

        passenger\_no+=1

    else:

        print("[x], {0} 번째 손님 (소요시간 : {1}분) 입니다.".format(i,driving\_time1))

print("총 승객수는 "+str(passenger\_no)+"입니다.")

\*이게 모범답안

from random import\*

driving\_time=range(5,51)

taking\_time=range(5,16)

passenger\_no=0

for i in range(1,51):

    driving\_time12=(sample(driving\_time,1))

    driving\_time1=driving\_time[0:1]

    if 5<= driving\_time1 <15:

        print("[O], {0} 번째 손님 (소요시간 : {1}분) 입니다.".format(i,driving\_time1))

        passenger\_no+=1

    elif driving\_time1<5 or driving\_time1>=15:

        print("[x], {0} 번째 손님 (소요시간 : {1}분) 입니다.".format(i,driving\_time1))

        continue

print(str(passenger\_no))

\* 내 답안

일단 elif 라고 써서 너무 길어 보임. else: 를 쓰면 바로 다른 조건으로 퉁치기 가능.

그리고 continue 생각해보면 굳이 쓸 필요가 없었다…

그리고 정수형으로 추출할 때는 randrage라는 함수를 써야되는데 이걸 몰랐다!!

17. 함수 생성과 응요

def open\_account():

    print("새로운 계좌가 생성되었습니다.")

open\_account()

\*\* def 함수이름(): => 이렇게 함수를 정의한다!!

def withdraw\_night(balance, money):

    commission = 100

    return commission, balance-money-commission

commission, balance4=withdraw\_night(balance3,500)

#커미션과 밸런스를 같이 받는데, 함수를 사용해서 값을 할당한다.

print("수수료 {0} 원이며, 잔액은 {1} 원입니다.".format(commission,balance4))

18. 함수 기본값 지정, 키워드값을 사용한 함수 호출

함수를 설정할 때, 이런식으로 기본값을 설정할 수 있다.

def profile(name, age=17, main\_lang="파이썬"):

    print("이름: {0}\t 나이: {1}    주 사용 언어: {2}" \

        .format(name,age,main\_lang))

profile("유재석")

profile("김태호")

def profile(name, age, main\_lang):

    print(name, age, main\_lang)

profile(name="장지수", main\_lang="파이썬", age=20)

19. 가변인자

ex. 장지수가 언어를 배워서 할 줄 아는 언어가 한 개 추가되었으면 함수도 언어를 추가해야되는 이런 번거로운 일이 발생한다. 이럴 때 사용할 수 있는 것이 가변인자이다.

가변인자의 예시는 다음과 같다.

def profile(name, age, \*language):

    print("이름: {0} \t 나이 : {1} \t" .format(name,age), end=" "  )

    for lang in language:

        print(lang, end=" ")

    print()

profile("장지수", 25, "파이썬","자바","영어","중국어","말레이시아","한국어")

profile("김태호",212, "코틀린", "파이썬")

20. 전역변수(프로그램 어디서든 할 수 있는 변수)와

지역변수(함수가 호출될 때만 만들어지고 호출 끝나면 사라지는 변수)

gun=10 #이게 전역변수

def checkpoint(soliders):

    gun=20 #이게 지역변수

    gun=gun- soliders

    print("[함수 내] 남은 총 : {0}".format(gun))

print("전체 총 : {0}" .format(gun))

checkpoint(2)

print("남은 총 : {0}".format(gun))

이 때 함수 내에서 전역변수를 사용하고 싶으면 앞에 global 을 붙이면 된다!!

하지만 일반적으로 전역변수를 많이하면 코드가 더러워지니까 아래 방법을 많이 쓴다!!

quiz.) 표준 체중을 구하는 프로그램 작성

\*표준 체중: 각 개인의 키에 적당한 체중

(성별에 따른 공식)

남자 : 키 x 키 x22

여자 : 키 x 키 x 21

조건 1ㅣ 표준 체중은 별도 함수 내에서 계산

\*함수명: std\_weight 전달값: 키, 성별

출력예제: 키 175cm 남자의 표준 체중은 67.38 kg 입니다.

21. 표준 입출력

print("python", "java", sep=" , ", end="?")

print("뭐가 더 재미있을까요?")

, 기준으로 두 문자를 분리해라, 그리고 끝에는 물음표를 넣고 끝내라 그래서 다음 문장이 연달아서 출력되는 것.

scores={"수학:":0, "영어:":50, "코딩:":100}

for subject, score in scores.items():

    #subject와 score라는 두개의 변수를 정의하고 얘네 값을 scores.items를 통해서 가져오겠다

    #scores.itmes 하면. scores 딕셔너리에 있는 키와 밸류들을 쌍으로 가져오기 때문!

    # print(subject, score)

    print(subject.ljust(8),str(score).rjust(4), sep=":")

    #subjects는8칸을 확보한 후 왼쪽으로 정렬해라.

    #score는 4칸을 확보한 후 오른쪽으로 정렬해라.

#은행 대기순번표 출력법!

#001, 002, 003, ...

for num in range(1,21):

    print("대기번호 : "+str(num).zfill(3))

\*\* str(number).zfill(3) 이 함수는 문자열을 3의 크기로 하고, 0을 채워넣으라는 것!

22. 다양한 출력 포맷

#빈 자리는 빈공간으로 두고, 오른쪽 정렬을 하되, 총 10자리 공간을 확보해라.

print("{0: >10}".format(5000))

print("{0:^<20,}".format(1000000))

\*\* {0}. format(변수) 형태로 출력할 건데, 그 뒤에 출력 조건을 어떻게 주냐를 이해해야됨!!

빈칸은 뭐로 채우고, 왼쪽정렬이고 몇칸을 확보해야되며, 숫자를 ,를 사용해서 보기 좋게 할 것인지 뭐 이런것들

# 소수점 출력

print("{0:.2f}".format(5/3))

\*\* format(숫자)를 소수점 둘째 자리까지 출력해줘라!!

23. 파일입출력

코드로 파일을 만들고, 안에 글자를 쓰고, 그리고 안에 내용을 불러오고 뭐 이런 것들 하는거인데… 과연 이게 많이 쓰일까? 생각보다 비중있게 다루네 …

score\_file=open("score.txt", "w", encoding="utf8") #w=write 이파일을 쓰기 위한 용도로 사용하겟다고 정의해줌.

print("수학 : 0", file=score\_file) # 이 파일을 만들고, 다음의 내용들을 작성한다.

print("영어 : 50", file=score\_file)

score\_file.close()

score\_file=open("score.txt", "a", encoding="utf8") #a는 append의 약자, 즉 추가해서 쓰겠다.

score\_file.write("과학 : 80") #write 함수는 이어서 쓰기 때문에 줄바꿈 문자를 넣어줘야된다.

score\_file.write("\n코딩 : 100")

score\_file.close()

score\_file=open("score.txt", "r", encoding='utf8')

print(score\_file.read())

score\_file.close()

score\_file=open("score.txt", "r", encoding='utf8')

print(score\_file.readline(), end="") #줄 별로 읽기, 한 줄 읽고 커서는 다음 줄로 이동

print(score\_file.readline())

print(score\_file.readline())

print(score\_file.readline())

score\_file.close()

파일이 몇줄일 줄 모를때 반복 문을 통해서

while True:

    line=score\_file.readline()

    if not line:

        break

    print(line)

score\_file.close()

lines= score\_file.readlines() #line을 리스트 형태로 저장해라

# for line in lines: #리스트 요소를 각각에 대해서 다음을 수행하라.

#   print(line, end="")

# score\_file.close()

24. Pickle: 프로그램상에 사용하는 데이터를 파일 형태로 내보내는 것.

file\_name=open(“file\_name.pickle”, “wb”) 뭐 이런식으로 피클형식의 파일을 만들고

import pickle, pickle.dump(a,b) => a에 있는 정보를 b라는 파일에 넣어라

profile=pickle.load(profile\_file) => profile\_file에 피클 형태로 저장되어있는 정보를 불러와라

25.\*with 문\*\*

import pickle

with open("profile.pickle", "rb") as profile\_file:

    print(pickle.load(profile\_file))

이렇게 with를 사용해서 한 줄로 읽을 수 있다!!

with open("study.txt", "w", encoding="utf8") as study\_file:

    study\_file.write("파이썬 생각보다 어려워요~")

with open("study.txt", "r", encoding="utf8") as study\_file:

    print(study\_file.read())

이렇게 하면 파일을 만들고, 그안에 내용을 적고 visual studio에서 확인하는 방법이 단 두줄의 코드로 가능해진다!!

#Quiz

회사에서 매주 1회 작성해야하는 보고서가 있다. 그 보고서의 형태는

x주차 주간보고

부서:

이름:

업무 요약:

1주차부터 50주차까지 보고서 파일을 만드는 프로그램을 작성하자

조건: 파일명은 1주차, 2주차, 처럼 만들고 형식은 txt 파일이다.

for i in range(1,51):

    with open("{0}주차.txt".format(i), 'w', encoding="utf8") as report\_file:

        report\_file.write("- {0} 주차 주간 보고 - \n부서: \n이름: \n업무요약: ".format(i))

so eazy

26. “클래스” – 매우 중요!!

클래스는 붕어빵 틀로 비유됨. 틀은 한 개인데, 그에 맞춰서 붕어빵을 계속 만들 수 있듯이…

서로 연관이 있는 함수와 변수들의 집합??

\*아래와 같은 형식으로 적는다.

class unit:

    def \_\_init\_\_(self, name, hp, damage):

        self.name=name

        self.hp=hp

        self.damage=damage

        print("{0} 유닛이 생성되었습니다.".format(self.name))

        print("체력 {0}, 공격력 {1}".format(self.hp, self.damage))

marine1=unit("marine", 40, 5)

tank=unit("tank",100,35)

\*\* 마린, 탱크는 유닛 클래스의 ‘인스턴스’ 라고 한다.

\_\_init\_\_ : 이 부분은 파이썬에서 쓰이는 생성자임. 마린, 탱크와 같은 객체를 만들 때 필요한 것.

\*\*멤버 변수: 위 코드에서는 self를 지칭함.

27. “메소드” : 클래스 안에 들어있는 함수를 지칭함.

class Attackunit:

    def \_\_init\_\_(self, name, hp, damage):

        self.name=name

        self.hp=hp

        self.damage=damage

    def attack(self, location):

        print("{0}:{1} 방향으로 적군을 공격 합니다. [공격력 {2}]"\

            .format(self.name, location, self.damage))

    def damaged(self, damage):

        print("{0} : {1} 데미지를 입었습니다.".format(self.name, damage))

        self.hp -=damage

        print("{0}: 현재 체력은 {1} 입니다.".format(self.name, self.hp))

        if self.hp <=0:

            print("{0} : 파괴되었습니다.".format(self.name))

#  클래스로 파이어뱃 만들기

firebat1 = Attackunit("파이어뱃", 50, 16)

firebat1.attack("5시")

firebat1.damaged(25)

firebat1.damaged(25)

\*\*\* 클래스와 메소드를 이용해서 스타크래프트 파이어뱃 유닛 생성 및 파괴 해보기!

=>메소드 지정에서 self는 반드시 지정해준다고 생각하면 된다.

\*\*\* 클래스는 필요한 함수들을 모두 모아놓은 것이며 클래스 안에서 정의된 함수들을 메소드라고 한다. 그런데 함수들에 공통적으로 사용되는 입력 객체가 있다면 그것을 미리 지정해주는 것이 편한데 이런 역할을 하는 것이 초기화 메소드 (\_\_init\_\_)이다.

firebat1이라는 객체에 클래스를 사용해서 이름, 체력, 공격력을 할당해준다. 그리고 그 객체에 클래스 내부의 함수를 사용해서 공격방향과 피해를 입은 후 체력들을 나타내는 것이다.

28. 상속과 다중상속

서로 다른 클래스간에 중복되는 것이 있을 경우, 앞의 클래스를 상속받아서 다른 클래스를 만들 수 있다.

\*\*상속

#일반유닛

class unit:

    def \_\_init\_\_(self, name, hp):

        self.name=name

        self.hp=hp

#유닛이라는 클래스 내용을 상속받아서 attackunit 클래스를 만들어보자

class Attackunit(unit):

    def \_\_init\_\_(self, name, hp, damage):

        unit.\_\_init\_\_(self, name, hp)

        self.damage=damage

\*\*다중상속

#드랍십 추가로 만들어보기

class flyable:

    def \_\_init\_\_(self, flying\_speed):

        self.flying\_speed = flying\_speed

    def fly(self, name, location):

        print("{0} : {1} 방향으로 날아갑니다. [속도 {2}]" \

            .format(name, location, self.flying\_speed))

class flyable\_attackunit(Attackunit, flyable):

    def \_\_init\_\_(self, name, hp, damage, flying\_speed):

        Attackunit.\_\_init\_\_(self, name, hp, damage)  #클래스 가져와서 초기화

        flyable.\_\_init\_\_(self, flying\_speed) #클래스 가져와서 초기화

#그냥 두개의 클래스를 상속받아서 새로운 유닛을 정의하기 편하게 한 것임.

valkyrie=flyable\_attackunit("발키리", 200, 6, 5)

valkyrie.fly("발키리", "3시")

\*\*날아다니는 공격유닛에 대해서 앞에서 만든 유닛들과 똑같이 정보를 입력받아서 생성하려면

29. 메소드 오버라이딩 : 자식클래스에서 새롭게 메소드를 정의해서 사용하기!

class flyable\_attackunit(Attackunit, flyable):

    def \_\_init\_\_(self, name, hp, damage, flying\_speed):

        Attackunit.\_\_init\_\_(self, name, hp, 0, damage)

        #attackunit 클래스에 있는 정보를 상속 받을 건데, 그 안에 스피드(지상속도)는 0으로 가져와라

        #왜냐면 공중 유닛이니까 지상 속도를 0으로 처리하는 것.

        flyable.\_\_init\_\_(self, flying\_speed)

    def move(self, location):

        print("[공중 유닛 이동]")

        self.fly(self.name, location) 이 부분 메소드 오버라이딩!

#메소드 내부에서 상속받은 클래스의 메소드가 또 사용된다.

        #flyable 클래스를 상속받았으니까 그 안에 있는 fly 함수를 사용할 수 있다.

vulture = Attackunit("vulture", 80, 10, 20) #vulture 가 self가 되는 건가?? ㅇㅇ

battlecruiser = flyable\_attackunit("battlecruiser", 500, 30, 5)

vulture.move("1시")

battlecruiser.fly("배틀크루저", "9시")

battlecruiser.move("6시")   #위에 fly 함수를 쓸 때는 name 변수를 받아 줘야했지만

#move 함수를 사용하게 되면 self 자리에 battlecruiser가 오니까 자동으로 이름을 받게 되서 함수 사용시 입력 값 한개만 해주면됨.

\*\*\*그래서 메소드 오버라이딩이라는 것은 빨간색 글자와 같다!!

\*\*\* pass : 함수나 메소드에서 아무것도 안하고 끝나는 것. (그냥 빈 코드의 함수를 만들 때 사용하는 것

#건물

class buildingunit(unit):

    def \_\_init\_\_(self, name, hp, location):

        pass

def game\_stop():

    pass

#클래스에서는 pass를 사용해서 종료함.(아무것도 안하는 것)

30. super

class unit:

    def \_\_init\_\_(self):

        print("unit 생성자")

class flyable:

    def \_\_init\_\_(self):

        print("flyable 생성자")

class flyableunit(unit, flyable):

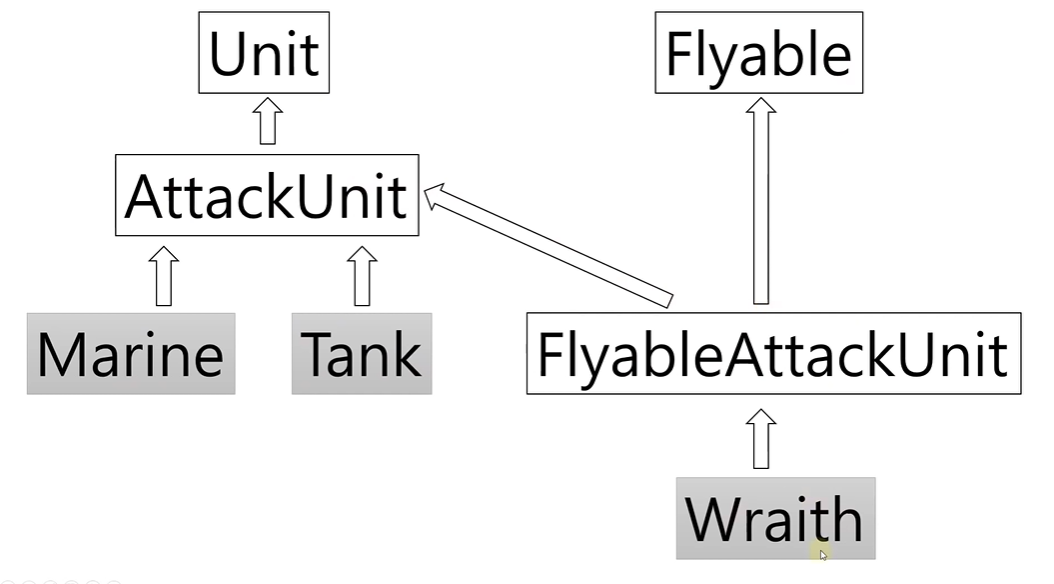
    def \_\_init\_\_(self):

        super().\_\_init\_\_()

# 드랍쉽

dropship = flyableunit()

\*\*flyableunit 클래스를 사용하는데 self인자에는 dropship이 온다. 그럼 super를 사용해서 상속받을 클래스들의 \_\_init\_\_ 함수를 가져오려고 하는데 \_\_init\_\_함수는 unit과 flyable 클래스 둘다에 포함되어 있다. 그럼 상식적으로 두개 모두 가져와야한다고 생각하지만, 맨 처음오는 클래스의 \_\_init\_\_만 가져와서 unit의 생성자 메소드를 가져오게 되고 결과적으로 unit 생성자 가 출력된다.



\*\* 계략도

Quiz.: 부동산 프로그램 작성하기

ex.

총 3대의 매물이 있습니다.

강남 아파트 매매 10억 2010년

마포 오피스텔 전세 5억 2007년

송파 빌라 월세 500/50 2000년

class House:

    def \_\_init\_\_(self, location, house\_type, deal\_type, price, completion\_year):

        self.location=location

        self.house\_type=house\_type

        self.deal\_type=deal\_type

        self.price=price

        self.completion\_year=completion\_year

    def show\_detail(self):

        print(self.location, self.house\_type, self.deal\_type, self.price, self.completion\_year)

product1=House("강남", "아파트", "매매", "10억", "2010년")

product2=House("마포", "오피스텔","전세", "5억", "2007년")

product3=House("송파", "빌라", "월세", "500/50" , "2000년")

list\_name=[]

list\_name.append(product1)

list\_name.append(product2)

list\_name.append(product3)

print("총 {0} 대의 매물이 있습니다".format(len(list\_name)))

for i in list\_name:

    i.show\_detail()

31. 예외처리

사용자가 잘못된 값을 입력하거나, 코드가 잘못되었을 때 사용할 수 있는 부분.

try:

    print("나누기 전용 계산기입니다.")

    nums = []

    nums.append(int(input("첫 번째 숫자를 입력하세요 : ")))

    nums.append(int(input("두 번째 숫자를 입력하세요 : ")))

    nums.append(int(nums[0] / nums[1]))

    print("{0} / {1} = {2}".format(nums[0], nums[1], nums[2]))

except ValueError:

    print("Error! you entered the wrong data")

except ZeroDivisionError as err:

    print(err)

\*\*\* try:

진행하는 코드

except errorname:

print(~~) => 특정 코드를 실행했을 때, 안되는 걸 알 수 있다.

# try 구문은 마지막에 finally를 쓰면 항상 그 밑에 코드가 실행된다!

finally:

print(“~해주셔서 감사합니다”)

Quiz!

다음 시스템 코드를 보고 적절한 예외처리 구문을 넣으시오.

1보다 작거나 숫자가 아닌 입력값이 들어오면 valuerror로 처리해라 : 잘못된 값을 입력햇다

대기 손님이 주문할 수 있는 총 치킨량은 10마리로 한정한다.

치킨 소진 시 사용자 정의 에러 [soldouterror]를 발생시키고 프로그램 종료

: 재고가 소진되서 더 이상 주문을 받지 않습니다.

32. 모듈: 필요한 것들 끼리 부품처럼 잘 만들어진 파일

모듈  필요한 것들 끼리 부품처럼 잘 만드는것

클래스와 함수 등의 코드를 담고 있는 파일을 모듈이라고 한다.

file.py

import moduel

moduel.price(3)

import moduel as mdd #불러오려는 모듈에 별명을 붙여서 줄여서 사용할 수 있다.

mdd.price\_morning(4)

#

from moduel import price\_soldier

price\_soldier(2)

\*\*\* 모듈을 가져오면 모듈.함수 => 내장함수를 쓸 수 있다, 모듈이름이 길면 별명으로 줄여서 쓸 수 있다. 그리고 모듈에서 가져오는 함수도 별명을 붙일 수 있다. 모듈안에 있는 특정 함수만 불러올 수 있다.

33. 패키지: 모듈들의 집합 (하나의 디렉토리에 여러 모듈들을 모아놓음)

#모듈에 있는 코드는 실행을 시켜서 정보를 저장해줘야한다!

import countries.korea   #countries 패키지에 korea 모듈을 가져온다.

#import는 클래스나 함수를 할 수 없고 모듈이나 패키지만 가능하다!!

trip\_to = countries.korea.koreapackage() #korea 모듈에 있는 koreapacakage를 쓴다

trip\_to.detail()

from countries import japan #countries 패키지에서 japan 모듈만을 가져오겠다

travel\_to = japan.japanpackage() #japan 모듈에 있는 japanpackage 클래스를 가져와서 지정하겠다

travel\_to.detail

34. 모듈 패키지

# 실제 패키지에서는 각 모듈이 잘 작동하는지 확인할 필요가 있다.

class japanpackage:

    def detail(self):

        print("[일본 패키지 3박 5일] 도쿄, 신주쿠 여행 (온천 투어) 55만원")

if \_\_name\_\_ == "\_\_main\_\_": #name이 main이면

    print("japan 모듈을 직접 실행")

    print("이 문장은 모듈을 직접 실행할 때만 실행돼요")

    trip\_to = japanpackage()

    trip\_to.detail()

else:

    print("japan 외부에서 모듈 호출")

japan 파일 모듈 내부에서 if\_\_name\_\_ == 구문을 통해 외부에서 호출되어 출력된 코드인지 알 수 있다!

35. pip 설치하기

구글에서 pypi 들어가서 원하는 패키지 검색~

그리고 터미널에 복사해서 붙여넣기 하면 다운로드 가능!

pip install ~~

pip show ~~ => 다운로드한 패키지의 정보를 알 수 있다

36. 내장함수

그냥 파이썬 안에 내장되어있는 함수,

build in functions python 검색하고 사용하면됨.

input, dir, print, tupe, type 뭐 이러것들이 있음!

37. 외장 함수

그냥 import로 불러와서 사용하면 됨…

#glob: 경로 내 폴더/ 파일 목록 조회

#os: 운영체제에서 제공하는 기본 기능

#time : 시간 관련함수

#timedelta: 두 날짜 사이의 간격

quiz!

프로젝트 내에 나만의 시그니처를 남기는 모듈 만들기

모듈 파일명은 byme.py