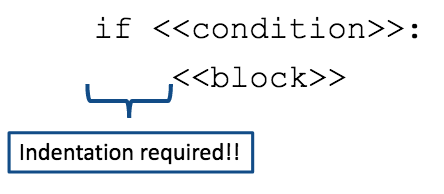
Week 4

* Conditional statement(조건문)



교재에 나와있는 if 조건문의 기본 형태이다

if 를 쓰고 옆에는 조건문(condition)을 넣는다 . 조건문에는 주로 불리언 자료형이 들어가거나  불리언자료형이 출력되는 문장을 넣는다.

그리고 해당 조건이 True 면 밑에 써있는 블록을 수행하는 것이다. 해당 조건이 False면 밑에 있는 블록을 수행하지 않고 넘어간다.

block 옆에는  공백과 indentation required!!라고 써있는데 조건문 안에 블록을 생성할 때 스페이스바 네번을 눌러 저런 공백을 맞추라는 것이다.

tab으로 하는 경우도 있는데 거의 대부분 스페이스바 네번으로 하니까 우리도 스페이스바 네번으로 하자.

indentation 도 중요한데  def 할 때도 마찬가지고 조건문을 만들면 조건문 맨 뒤에 반드시  : (콜론)을 잊지말고 써주자!

* Flow chart(순서도)

 if 조건문을 공부할 때는 Flow chart를 같이 그리거나 생각하면서 하는게 좋다. 만약 복잡하게 조건문을 써야한다면 (연속적인 if-elif 중첩 if-elif문 등) 순서도를 그려보고

순서도대로 코딩을 하는 것도 많이 도움이 되었다.

순서도 그리는 사이트는 일단  [순서도 그릴 수 있는 사이트(www.draw.io/)](https://www.draw.io/) 를 사용하면 유용하다 .

* if -elif -else

if-else는 조건문이 하나만 쓸 수있으므로 다양한 조건을 판단하기에는 어렵다. 그래서 elif를 써 다양한 조건을 설정할 수있다.

중요한건 if에서 거짓이어서 elif문으로 넘어갈 때 elif 조건에 대해 생각해보면 예를 들어  if에서 a 라는 조건이 있는데 a가 거짓이어서 elif  구문으로 넘어왔다.

그럼 elif  에서는 a가 거짓이라는 조건을 굳이 써줄 필요가 없다. if에서 걸러져 넘어온 상태이기 때문이다.

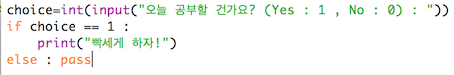
이 점을 생각하고 작성하면 더 쉽고 간단하게 작성할 수 있다.

* pass(if문에서만 쓸 수 있는 것 아님!)

if 문을 만들면 블록을 생성해야된다. if의 조건을 만족해 블록을 설정하는데 아무런 일을 하지 않게 설정하고 싶으면 pass를 쓴다

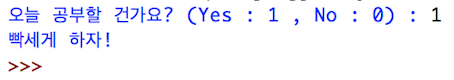
블록 안에 뭔가를 쓰긴 써야하므로  아무런 일을 하지 않게 설정하려면 블록에 pass를 쓰면 된다.

예를 들어



이런 식으로 간단하게 코드를 짜보고 실행을 해보자

1인 경우 : print("빡세게 하자!") 실행



0인 경우 : pass 실행 -> 출력되거나 실행되는 거 없이 바로 끝남

https://khlug.org/image/3535ac0c35d73e8d.png

* if문 한 줄로 작성하기

위에서 본 코드에서 else : pass라고 한 줄로 이어서 썼다. 원래라면 블록에다 해야되는데 그냥 이어서 쓴 것이다.

이것도 파이썬에서 어떻게 보면 편리한 기능이다. 만약 수행할 문장이 한 줄정도로 간단하게 쓸 수 있는 코드이면 이어서 써서 조금 더 간략하게 쓰고

코드 보기 좋게 할 수 있다.

Module(모듈)

* 모듈이란?

모듈은 함수나 변수, 아니면 클래스 등 을 모아 놓은 파일이다.

기본적으로 파이썬을 깔면 파이썬 라이브러리에 모듈 파일들이 있다. 이걸 기본으로 제공되어진 모듈이라고 생각하자.

파이썬 라이브러리에 내장되어있는 모듈은 종류가 많으므로 필요 한 것이 있으면 구글링으로 검색해보자

또 파이썬 라이브러리에 저장되어있는 모듈만을 사용할 수 있는 것이 아니다. 필요한 모듈을 다운 받거나 내가 직접 모듈을 만들 수 있다.

모듈을 불러오는 명령어는 import이다

import 모듈이름

입력하면 모듈을 불러와 모듈 안에 저장된 변수, 함수, 클래스를 쓸 수 있다.

파이썬 기본 모듈은 이렇게 그냥 import를 쓰면 되는데 위에서 언급한 내가 만들거나 다운 받은 모듈은 어떻게 사용하는지 알아야겠다.

import는 현재 디렉터리에 있는 파일이나 파이썬 라이브러리가 저장된 디렉터리에 있는 모듈 만 불러올 수 있다.

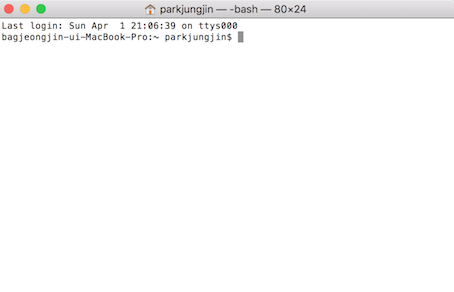
따라서 terminal창으로 내가 만든 모듈이나 다운받은 모듈이 지금 현재 디렉터리에 저장되어있는지 확인해야한다.

PycharmProjects라는 폴더에 module1.py를 만들었다.

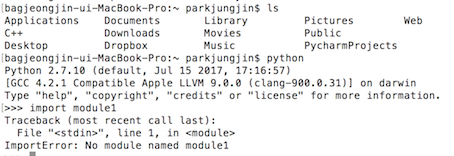
https://khlug.org/image/4325ac0cf67ac178.png

맥에서 terminal을 실행시켜보자.윈도우는 cmd를 실행시키면 된다. 윈도우는 명령어는 help를 치면 나오거나 구글링을 통해 알자

기억상에  cd(경로변경) dir(현재 디렉터리에 있는 파일 보여줌) 등등일 거다.



그 후 터미널에서 파이썬을 실행시키고 모듈을 사용해보았다.



terminal에서 python이라고 입력하면 바로 파이썬 언어인 대화형 인터프리터가  실행된다. (파이썬이 깔려있으면 어느 디렉터리에서든 실행이 된다)

그리고 창에 import module1을 실행시키면 에러가 난다

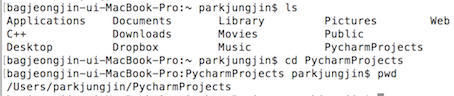
에러는 module1을 찾을 수가 없다는 것이다.

위에서 언급했듯이 import는 현재 디렉터리에 있는 파일이나 파이썬 라이브러리가 저장된 디렉터리에 있는 모듈 만 불러올 수 있다.

그럼 내가 PycharmProjects라는 폴더로 디렉터리를 이동하면 실행시킬수 있다는 것이므로 한번 디렉터리 이동을 해보자

https://khlug.org/image/7985ac0cdfdd67f1.png

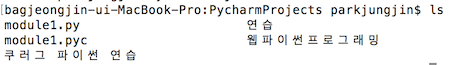
terminal 창에 pwd라고 입력하면 현재 경로를 표시해주는데 나같은 경우에는 /Users/parkjungjin이라는 경로에 있는 것이다.



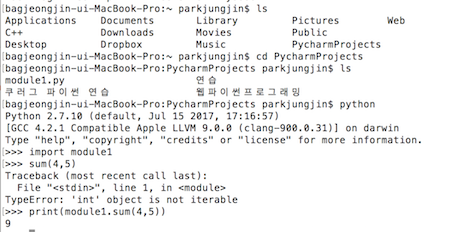
그 후로 ls 라고 입력하면 하위 디렉터리를 보여주는데 나는 내가 만든 모듈을 PycharmProjects에 저장했다.

cd는 cd 뒤에 입력된 디렉터리로 이동시켜주는 명령어다 cd PycharmProjects로 PycharmProjects디렉터리로 이동했다

그후 pwd로 현재경로를 확인해보니 /Users/parkjungjin/PycharmProjects 로 경로가 이동되었음을 확인 할 수있다



ls로 확인해보니 module1.py 가 있는 것이 확인 된다 이제 이 상태에서 python  을 입력해 파이썬 인터프리터를 실행해보자



처음 경우랑 다르게 import module1이 실행 된 것을 확인할 수 있다.

그래서 그냥 module1에 저장된 sum함수를 실행해 보니 에러가 난다.

import로 불러온 모듈에 있는 함수를 이용하려면 위 그림에서 보는 것과 같이

module1.sum(4,5)처럼

모듈이름.함수이름

이런식으로 사용해야한다.

위에 terminal에서 쓴 명령어는 파이썬이 아니니 헷갈리지 말자

맥os terminal 명령어

| **terminal 명령어** | **의미** |
| --- | --- |
| **ls** | 하위 디렉터리를 보여준다  (- a를 붙여주면 모든 파일을 보여준다) |
| **pwd** | 현재 디렉터리(경로)를 표시해준다. |
| **cd 내용** | 내용에 입력된 디렉터리로 이동한다. |
| **python** | 파이썬 대화형 인터프리터를 실행시킨다. |

모듈 사용방법 :

1. 모듈 자체를 불러오는 경우(모듈 안에 존재하는 모든 변수, 함수, 클래스를 사용할 수있다)

import 모듈이름

모듈이름.함수,변수,클래스 이름

ex)

>>>import module1

>>>print(module1.sum(4,5))

9

2. from 모듈이름 import 모듈함수,변수,클래스

모듈 안에서 원하는 함수만 불러 올 수있다.

ex)

>>>from module1 import sum

>>>print(sum(4,5))

9

(전공자가 아니라서 잘 모르겠지만 아마 메모리 사용량을 줄이는게 좋기 때문에 따로 불러와 사용하는 것 같다.

다 불러오면 변수나 함수이름 이 겹칠 경우가 늘어나서 복잡하기 때문일 것 같다.

from ~import~ 를 쓰면 번거롭게 모듈이름.모듈함수이름 이런 식으로 작성을 안해도 되고 편해서 그런 것 같다.)

번외로 from 모듈이름 import \* ->모듈 안에 있는 모든 것을 다 불러서 사용 (\*은 모든 것이라는 뜻)

from~import~를 쓰면 위에서 본 것 처럼 그냥 .을 안붙이고도 사용 할 수있는데

모듈 안에 있는 모든 것을 다 불러서 사용하면 변수,함수 이름 부터 메모리 사용량 까지 정말 복잡.

(모듈안에 있는 변수

특히 상수(constant)로 규정되어진 변수는 그 값(value)을 바꾸는 것은 좋은 생각이 아니라고 하셨다.

파이썬은 바꾸는게 가능해서 바꿀 수 있지만 지양.)

파이썬을 깔면 라이브러리에 있는 모듈들은 내장 함수 처럼 그 수가 많고 다양한 종류가 있다. 모듈을 불러와 쓰는 외장함수라고 한다.

 다운 받은 모듈이나 내가 만든 모듈은  terminal창을 열고 해당 모듈이 있는 디렉터리로 이동한 후에 모듈을 쓸 수있다.

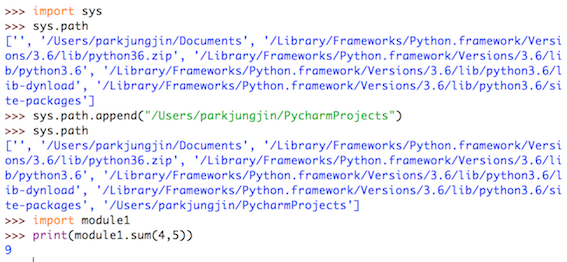
이 방법은 매우 귀찮고 제한적이다.

따라서 저장한 디렉터리로 이동하지 않고 모듈을 불러와 사용하는 방법이 있다.(참고 : Do it 점프 투 파이썬)

1. sys 모듈 이용하기
2. PYTHONPATH 환경 변수 사용하기

1.sys 모듈 이용하기

sys모듈은 파이썬을 설치하면 기본으로 있는 라이브러리 모듈이다



shell창을 열어서 했다.

방법은

1. import sys로 sys 모듈을 불러왔다.

2.sys.path를 입력했는데

sys.path는 파이썬 라이브러리가 설치되어있는 디렉터리를 보여준다.

"import는 현재 디렉터리에 있는 파일이나 파이썬 라이브러리가 저장된 디렉터리에 있는 모듈 만 불러올 수 있다. "라고 알고있다.

따라서 파이썬 라이브러리가 설치되어있는 디렉터리에 내 모듈이 저장된 디렉터리를 추가하면 모듈이 저장된 디렉터리로 이동하지 않고 모듈을 불러서 사용할 수 있는 것이다.

sys.path는 출력되는 자료형이 list이다 . 후에 Week6 Lists에서 더 자세하게 공부할 거지만 리스트에는 append라는 함수로 요소 한 개를 리스트안에 추가할 수있다.

리스트.append(추가할 요소)를 하면 리스트에 추가할 요소가 들어가진다.

따라서 sys.path는 리스트 이므로 sys.path.append("내 모듈이 저장되어져있는 디렉터리")를 하면

내 모듈이 저장되어져있는 디렉터리도 파이썬 라이브러리가 저장된 디렉터리가 되는 것이다.

그래서 내 경우에는 PycharmProjects에 저장되어있으니까 sys.path.append("/Users/parkjungjin/PycharmProjects") 를 한 것이다.

(Tip : 윈도우, 맥에서 디렉터리 경로 값이 다르니 참고-> 나는 맥os이다.

  Tip : terminal에서는 /(slash) , \(back slash)든 상관 없지만, 소스 코드안에서는 반드시 /  또는 \\ 를 사용해야한다. -> 나는 / 를 사용했다)

후에 sys.path를 다시 입력해 확인해 보면 "/Users/parkjungjin/PycharmProjects" 가 추가되어진 것을 확인 할 수있다.

그래서 module1을 불러와 만든 sum함수를 사용해보면 에러없이 잘 되는 것을 확인 할 수 있다.(전 처럼 terminal에서 디렉터리 바꿔서 안해도 됨!)

2.PYTHONPATH 환경 변수 사용하기

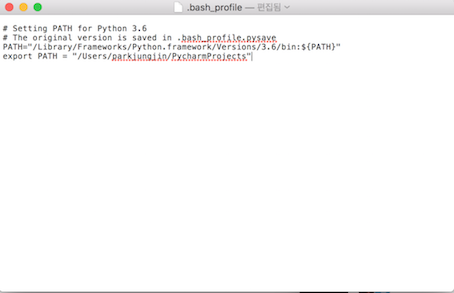
PYTHONPATH 환경 변수에 모듈이 저장된 /Users/parkjungjin/PycharmProjects 디렉터리를 설정하는 것이다.

맥os 기준이다.

terminal 창을 킨다.

https://khlug.org/image/4705ac0ecca626ff.png

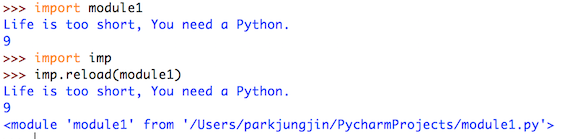
$ open -e.bash\_profile을 입력한다. 그러면 bash\_profile 창이 뜨는데



export path에 export path="디렉터리 경로"를 하면 된다. 나같은 경우에는 export path="/Users/parkjungjin/PycharmProjects"로 했다.

이제 모듈을 저 디렉터리에 저장을 하면 디렉터리 이동이나 별도의 추가 작업 없이 module1 모듈을 불러와 사용할 수있다.

* imp모듈



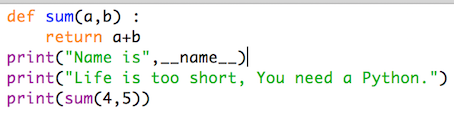
imp모듈중 reload 를 배웠다.

a모듈을 import로 부르고 a모듈의 코드를 변경했을 때 변경된 a모듈로 부르고 싶을때 imp.reload를 쓴다.

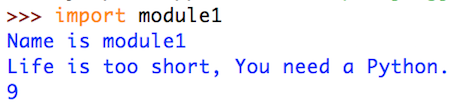
Python에서는 모듈을 변경하는 것은 별로 좋지 않은 생각이므로 자기가 만든 모듈을 import하고 모듈 변경을 한 후 변경된 모듈을 쓰고 싶을 때 쓰라는 교수님의 조언이 있으셨다.

* \_\_name\_\_

module1파일을 수정했다.



그리고 shell창에서 module  을 불러와봤다.



이번에는 직접 module1.py파일을 실행 시켰다.

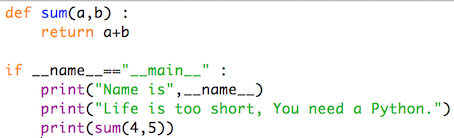
https://khlug.org/image/7755ac0f67feb6b8.png

두 경우 \_\_name\_\_  이 차이나는 것을 볼 수있다.

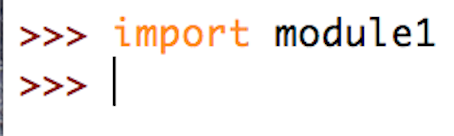
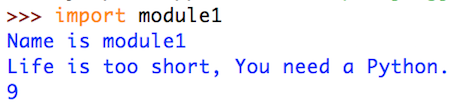
|  |  |
| --- | --- |
| 경우 | \_\_name\_\_ |
| import한 경우 | 파일이름 |
| 직접 실행한 경우 | \_\_main\_\_ |

여기서 if \_\_name\_\_=="\_\_main\_\_" : 을 알 수있다

module1  을 다시 수정해보자



이렇게 수정하고 모듈을 import module1으로 불러와보자

                                          if \_\_name\_\_ == "\_\_main\_\_"  수정 후                                                                                      if \_\_name\_\_ == "\_\_main\_\_"  수정 전

위의 예제들을 통해 알 수 있듯이  if \_\_name\_\_ == "\_\_main\_\_"    을 사용하면 직접  module1.py를 실행해야  if \_\_name\_\_ == "\_\_main\_\_"     이 참이 되어

 if 문 아래에 있는 블록들이 실행된다.  하지만 import  를 이용해 모듈을 불러오는 경우면  \_\_name\_\_이 module1이 되어 if문이 거짓이 되어서 if문 블록이 실행 되지 않는다

-> 파이썬 모듈을 만들고 그 모듈을 테스트하기 위해 보통 이 방법을 사용한다고 한다.

배운 모듈, 파이썬 라이브러리에 있는 모듈

정확하게는 모듈 안 에있는 함수 즉 외장 함수다.

* math

영어 뜻 처럼 수학 기능을 갖고 있는 모듈이다 . 거의 공학용 계산기 인데 계산기가 없다면 잘 활용하면 계산기보다 쓰기편할 것 같다.

import math를 하고 help(math)를 하면 math모듈이 갖고있는 함수 종류를 볼 수있다. 정말 다양한 함수가 있으니 보고 잘 쓰면 좋을 것 같다.

많이 쓰이는게

sqrt(x) : x의 제곱근을 리턴한다

pi : 변수다. 파이 값을 갖고있다. 자료형은 float

sin(x) : 사인(x) 리턴한다

cos(x) :  코사인(x) 리턴한다

tan(x) :  .....

흔히 우리가 알고 있는 공학용 계산기와 비슷하다 .

* random

random  모듈은 난수(랜덤값)을 발생시키는 모듈이다

random() : [0,1) 사이에 소수점이 있는 난수를 리턴한다. -> 출력값은 float, 범위에 0은 포함,1은 포함하지 않는다.

uniform(start,stop) : [start,stop) 사이에 소수점이 있는 난수를 리턴한다. -> 출력값은 float , 범위에 start는 포함 stop는 포함하지 않는다.

randrange(start, stop, step) : [start, stop)  사이에 step값으로 건너뛰는 정수인 난수를 리턴한다.

-> 출력값은 int

     범위는 start는 포함 stop 은 포함하지 않는다

     start , step값은 입력을 안해도 된다

     start 값을 입력 안하면 [0,stop)으로 계산된다.

     step값을 입력하면 start, start+step, start+2\*step, start+3\*step, .... , stop-step 값 중에서 출력이 된다. 슬라이싱에 step 이랑 개념이 같다.

randint(start,stop) : [start, stop] 사이에서 정수인 난수를 리턴한다. -> 출력값은 int 범위는 start을 포함, stop도 포함한다.

choice(s) : 나열형 자료형(s)(list, tuple, str, ....) 의 요소 중 무작위로 하나를 선택하여 리턴한다. ->나열형 자료형이라고하는데 그냥 반복가능한 자료형에서 딕셔너리만 안 되는 것 같다.

shuffle(l) : list(l)의 요소들을 무작위로 섞은 후 섞인 리스트를 리턴한다. -> list만 되는 것 같다.

* turtle(자세히는 잘 모르겠다)

그림을 그려주는 모듈이다

Pen() :  그림판과 펜을 생성한다.

color("색깔") : 펜의 색깔을 정해준다 ex) "red", "blue", " green"

shape("모양") : 펜의 모양을 정해준다 ex) "turtle", "arrow", "triangle"

up() : 펜의 방향을 위로 한다.

down() : 펜의 방향을 아래로 한다.

right(angle)  :  펜의 방향을 오른쪽으로 angle만큼 돌린다.

left(angle)  :  펜의 방향을 왼쪽으로 angle만큼 돌린다.

clear() : 그림판에 그려진 모든 걸 지운다(펜의 위치는 마지막 상태 그대로 유지됨)

reset() : 그림판에 그려진 모든 걸 지우기 + 펜의 위치까지 맨 처음 상태로 만든다.

done() : 프로세스가 종료가 안됨. 그림판에 그린 그림을 유지할 수 있음 done() 을 안쓰면 그림을 그리다가 다 그리면 프로세스가 종료되어서 그림판도 없어진다.

forward(x) : 펜을 x 픽셀 만큼 앞으로 그린다.

back(x) : 펜을 x 픽셀 만큼 뒤로 그린다.