■ 세상은 방향성이 있다

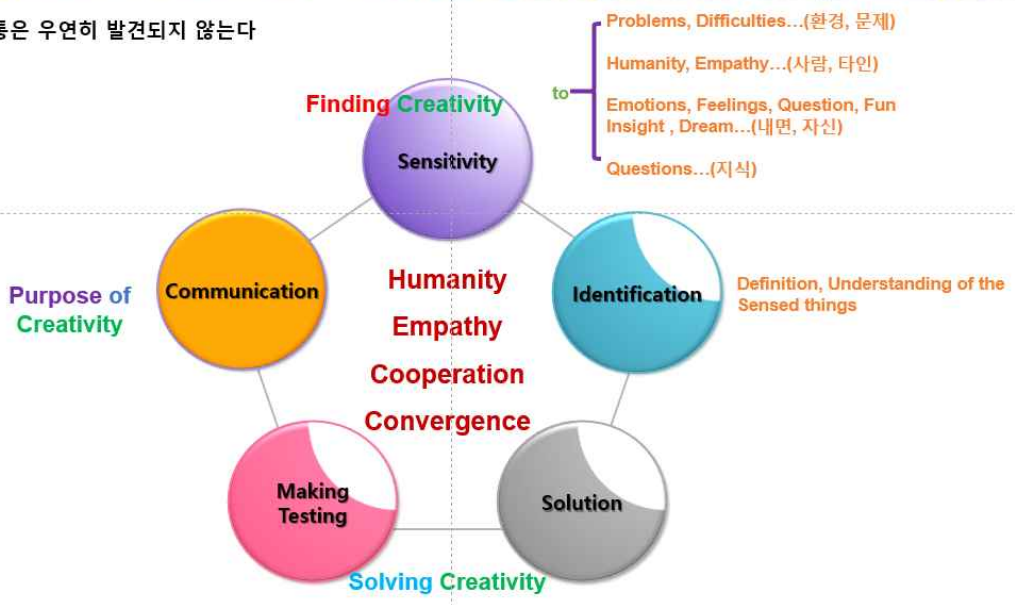
우주 팽창 - Vortex 운동

일반상대성이론 - 시간방향으로 빛의 속도로 이동중

연결(Connection)관점에서의 산업혁명과 공유 경제

	1차 산업혁명	2차 산업혁명	3차 산업혁명	4차 산업혁명
연결의 기술	증기기관, 철로	전기, 통신 인프라	컴퓨터, 인터넷, SW	모바일 인터넷, AI 블록체인
연결의 대상	제품(물질), 지역	에너지, 기업/가정	정보/데이터,컴퓨터	개인
	5차 산업혁명 = 연결의 방향(다음 연결 대상)에서 Hint			
핵심 연결 수단	철로	아날로그 통신	디지털 통신(인터넷)	모바일 인터넷
연결의 효과	기계에 의한 대량 생산	자동화 아날로그 플랫폼 경제	정보화 디지털 플랫폼 경제	개인화/지능화 공유 경제
연결의 가치	플랫폼에 의한 독점			공유
연결의 주체	국가	국가/대기업	글로벌기업	개인(P2P)
주요 경제 형태	생산자와 소비자가 분리된 독점적 상업 경제			Prosumer 로서의 개인간 경제
부작용	야수적 자본주의 -> 양극화			이익의 여전한 독점

창의성과 소통은 우연히 발견되지 않는다



■ 프로그래밍 교육은 시대의 변화에 맞추어 이뤄져야 한다

■ AI 시대에는 창의성, 공감, 소통이 핵심 가치이다

■ 프로그래밍이란

- 컴퓨터, 알고리즘, 데이터를 이용하여 문제를 해결하고 개인의 자아실현과 타인과의 공유가치를 실현하는 것
- 컴퓨터는 Narrow Problem Solver이고 사람은 범용 문제 해결 시스템 (General Problem Solver))

- 고전적 프로그래밍:
  사람이 정한 문제와 찾은 해결방법을 컴퓨터와 상호작용하며 문제를 해결하는 것
- 머신(딥)러닝 프로그래밍:
  사람이 정한 문제의 해결방법을 데이터와 범용 AI알고리즘을 이용하는 해결하는 것

■ 프로그래밍 학습(교육)의 목표

- 배경: 모바일인터넷으로 개인이 연결된 시대=>개인간의 연결로 개인의 권력이 커진 시대
  생산/소비자가 구분없는 메이커 시대, 인공지능이 중요도구로 사용되는 AI시대
  AI를 잘 활용하는 창의성 시대, 환경과 가치를 이웃과 공유하는 공유가치 시대
- 의미있는 문제의 발견
- 발견한 문제의 효율적(창의적) 해결
- 타인과 협력/융합하여 문제를 해결하는 경험
- 컴퓨터를 이용한 다양한 문제 해결 경험
- 보다 나은 공동체를 위한 공감과 소통 - 문제 선정(발견)의 이유와 중요성
- 알려진 유용한 문제해결 알고리즘과 자료구조의 이해
- 좋은 프로그래밍 습관과 프로그래밍 언어의 제반 특징
- AI의 문제 해결에의 적용 경험

■ 컴퓨터에게 일을 시킬 때

- 해당언어의 명령어들을 이용하여 좋은 프로그램을 작성/실행하므로써 원하는 결과 얻음
- 이 때, 사용할 자원(resource)은 컴퓨터, 프로그램소스, 데이터가 있다
- 풀고자 하는 문제와 운영환경에 따라 컴퓨터의 종류, 운영체제, 프로그래밍언어 등을 달리 선택해야 한다
  (처리속도, 휴대성, 개발기간, 운영공간, 운영비용, ……)

■ 좋은 프로그램이란

- 프로그램은 지능이 없는 컴퓨터가 처리하기에 문제가 없도록 문법적으로 정확해야 하고
- 수정이나 보완(업데이트), 타인의 이해가 용이하도록 주석(설명)과 Identifier(변수, 함수, 클래스 등)의 사용이 적절하여야 하고
- 문제의 해결 방식(알고리즘)이 복잡하지 않고 간결할수록 좋고
- 오류가 없이 작동하는지 논리적으로 엄밀해야 하고
- 자원 (메모리, CPU, 보조메모리)의 사용이 효율적이고 처리시간이 적절해야 한다

■ 타 언어대비 파이썬의 장점

- 영어에 가까운 쉬운 문법
  학습에 용이
- 다양한 자료구조,함수,라이브러리를 제공하여 다른 언어보다 상대적으로 길이가 짧은 코딩
  작성된 프로그램의 이해와 수정, 개발비용(시간) 면에서 유리
- 인터프리터 언어여서 동작을 바로바로 확인이 가능함
  개발시간 단축으로 프로토타이핑(초기버전 개발)에 유리

1. 문장(Statement)

- 프로그램은 문장(statement)으로 이루어져 있고, 파이썬 번역기는 문장단위로 번역 및 실행한다
- 프로그래머는 적절한 문장들로 컴퓨터에게 일을 시킨다.
  따라서 프로그래밍을 한다는 것은 적절한 명령어를 순서대로 나열하는 것이다
- 파이썬에서는 문장의 끝을 구분하는 기호를 별도로 가지지 않고, 기본적으로 한 줄 단위로 작성한다
  한 개의 줄에 여러개의 문장을 사용할 때는 각 문장의 끝에 세미콜론(;)을 추가한다

2. 파이썬 키워드

- 컴퓨터가 특정한 의미의 명령어로 사용하기 위하여 예약해 놓은 단어로서
  프로그램의 실행 흐름을 제어하는 등의 기본적인 처리의 기본 명령어 (33개)
- 주어진 의미 외에 다른 용도(변수명, 함수명 등)로 사용하면 안됨
  (※번역기가 혼동을 일으켜서 번역 오류를 일으키므로, 아예 번역시 문법적 오류로 처리함)
- 키워드로 처리할 수 없는 일들은 함수와 라이브러리를 이용함
  키워드 > 내장함수(모듈) > 사용자 함수(자체제작) 또는 외부 함수(외부모듈)
- 키워드의 사용법은 프로그래밍 언어의 학습 순서에 따라 점차적으로 학습하면 됨

```
import keyword
# print(keyword.kwlist)
kwlist = keyword.kwlist
for i in range(0,len(kwlist)):
    print("[{:10}]".format(kwlist[i]), end=" ")
    if (i+1)%5==0: print()

[False     ] [None      ] [True      ] [and       ] [as        ]
[assert    ] [break     ] [class     ] [continue  ] [def       ]
[del       ] [elif      ] [else      ] [except    ] [finally   ]
[for       ] [from      ] [global    ] [if        ] [import    ]
[in        ] [is        ] [lambda    ] [nonlocal  ] [not       ]
[or        ] [pass      ] [raise     ] [return    ] [try       ]
[while     ] [with      ] [yield     ]
```

3. 식별자(Identifier)

- 변수, 상수, 함수, 사용자 정의 타입 등에서 다른 것들과 구분하기 위해서 사용되는 객체의 이름,
- 변수의 이름, 함수의 이름, 사용자 정의 타입의 이름 등 '이름'을 일반화 해서 지칭하는 용어
- 변수명 작성
  - 카멜표기법 : 첫글자는 소문자, 단어의 첫글자만 대문자
  - 변수명만 보고도 뜻을 알 수 있도록 작성
- 함수명 작성
  - 파스칼표기법 : 단어의 모든 첫글자를 대문자로
  - 함수명만 보고도 뜻을 알 수 있도록 작성

4. 입출력 장치와 표준입출력 명령어

- 컴퓨터에게 일을 시키거나 시킨일의 결과를 확인하기 위하여 대부분 입출력 명령어를 사용하기 마련
- 컴퓨터 구성
  - 입력장치 (키보드, 마우스, 카메라, 스캐너, ……)
  - 출력장치 (모니터, (3D)프린터, 카메라, 플로터, ……)
  - 저장장치 (하드디스크, SSD, USB, 주기억장치(Ram)……)
  - 연산장치 (CPU, GPU)
- 표준입력장치 : 키보드
- 표준출력장치 : 모니터

- 표준입력 명령어 (input) : 문자열 객체(<str>)를 넘겨준다

  사용1) 변수 = input()

  ```
  name = input('');
  ```

  사용2) 변수 = input(‘문자열’)

  ```
  name = input('이름을 입력하세요 ');
  ```

  사용3) 변수 = int(input(‘문자열’))

  ```
  age = int(input("나이를 입력하시오 : "))
  ```

  사용4) 변수 = float(input(‘문자열’))

  ```
  radius = float(input("반지름값을 입력하시오 : "))
  ```

- 표준출력 명령어 (print) :

  사용1) print(‘문자열’)

  ```
  print(‘제 이름은 Kim입니다’)
  print(‘제 이름은 “Kim”입니다’)
  ```

  사용2) print(‘문자열1’, ‘문자열2’, sep=’:‘)

  ```
  print(‘Kim’, ‘Lee’, ‘Park’)
  ```

  사용3) print(‘문자열1’, ‘문자열2’, end=’‘)

  ```
  print(‘Kim’, ‘Lee’, ‘Park’, end=’‘)
  ```

- 출력시 기본 확장 문자(escape sequence)

  \' : 따옴표 문자
  \" : 쌍따옴표 문자
  \ : backslash 문자
  \a : bell 문자
  \b : backslash 문자
  \f : Formfeed 문자
  \n : newline 문 \r : carriage return 문자(\n와 동일하지 않다.)
  \t : tab 문자
  \v : vertical tab 문자

■ 출력시 길이와 정렬

print('Python is [{:15}]'.format('good'))	Python is [good]
print('Python is [{:<15}]'.format('good'))	Python is [good]
print('Python is [{:>15}]'.format('good'))	Python is [good]
print('Python is [{:^15}]'.format('good'))	Python is [good]
print('당신의 나이는 [{:15}]세'.format(22))	당신의 나이는 [22]세
print('당신의 나이는 [{:<15}]세'.format(22))	당신의 나이는 [22]세
print('당신의 나이는 [{:>15}]세'.format(22))	당신의 나이는 [22]세
print('당신의 나이는 [{:<15}]세'.format(22))	당신의 나이는 [22]세
print('-' * 40)	----------

5. 간단한 기본 산술연산자(+, -, *, /)

■ 아래 프로그램을 그대로 입력해서 실행하여 보세요

```
'''
        ============================
        *** 3과목 성적 처리 프로그램 ***
        ============================
'''
koreanScore = 88
englishScore = 90
mathScore = 92

sumScore = koreanScore + englishScore + mathScore
averageScore = sumScore / 3.0

print('성적합계: ', sumScore)
print('성적평균: ', averageScore)

# 평균점수를 10% 인상
averageScore = averageScore * 1.1
print('조정 후 성적평균: ', averageScore)
```

■ 부산, 광주, 대구, 인천, 제주 다섯 개 도시의 2019년 년평균 강우량을 찾고,
키보드에서 5개 강우량 값을 입력받아 5개도시의 평균을 출력하는 파이썬 프로그램을 작성하시오

6. 모듈의 사용

- 모듈
  - 키워드나 함수보다 논리적으로 큰 단위의 프로그램 소스의 모음으로 소스(코드) 재사용에 유리
  - 자주 사용되는 코드나 유용한 코드를 논리적으로 묶어서 관리하고 사용하는 단위
  - *.py 형식으로 파이썬 파일로 저장
  - 함수, 클래스, 변수선언과 실행코드로 구성
- 내장모듈
  - 파이썬이 제공하는 표준 라이브러리 모듈
- 외부모듈
  - 사용자가 만든 모듈

- 모듈 사용 기본

  import 모듈
  import 모듈1, 모듈2, 모듈3 ...
  import 모듈명 as 별명

- 모듈이름 생략하여 사용

  from 모듈 import 함수
  from 모듈 import 함수1, 함수2, 함수3 ...
  from 모듈 import *
  from 모듈 import 함수 as 별명

import datetime
import math
import math as m
from math import pi
from math import *
 	
# import datetime 	
now = datetime.datetime.now()
print(now)
 	
# import math 	
print("The value of pi is", math.pi)
 	
# import math as m 	
print("The value of pi is", m.pi)
 	
# from math import pi 	
print("The value of pi is", pi)
 	
# from math import * 	
print("The value of e is", e)
	2018-07-06 17:27:36.859288
	
	The value of pi is 3.141592653589793
	
	The value of pi is 3.141592653589793
	
	The value of pi is 3.141592653589793
	
	The value of e is 2.718281828459045

7. 변수와 입출력문 실습문제 - 최저임금

- 아래 표는 2010년부터 2019년까지의 10년간 년도별 최저임금이다

2019~2010 연도별 최저임금액								
적용년도	적용기간	시간급	일급 (8시간 기준)	월급 (209시간 기준, 고시기준)	인상액	인상률	심의 의결일	결정 고시일
2019	19.1~19.12	8,350	66,800	1,745,150	820	10.9%	18.7.14	18.8.3
2018	18.1~18.12	7,530	60,240	1,573,770	1060	16.4%	17.7.15	17.8.4
2017	17.1~17.12	6,470	51,760	1,352,230	440	7.3%	16.7.16	16.8.5
2016	16.1~16.12	6,030	48,240	1,260,270	450	8.1%	15.7.9	15.8.5
2015	15.1~15.12	5,580	44,640	1,166,220	370	7.1%	14.6.27	14.8.4
2014	14.1~14.12	5,210	41,680	1,088,890	350	7.2%	13.7.5	13.8.2
2013	13.1~13.12	4,860	38,880	1,015,740	280	6.1%	12.6.30	12.8.1
2012	12.1~12.12	4,580	36,640	957,220	260	6.0%	11.7.13	11.8.1
2011	11.1~11.12	4,320	34,560	902,880	210	5.1%	10.7.3	10.8.3
2010	10.1.~10.12	4,110	32,880	858,990	110	2.8%	09.6.30	09.8.3

지난 10년간(2019-2010) 최저임금 인상률

(1) 년도별 시간당 임금(시간급) 10개만 프로그램에서 미리 변수 10개에 저장하시오
(변수명을 만들 때 어떤 이유로 그 변수명을 만들지 생각하고 만드세요)
(2) 2011년에서 2019년까지 9개 년도의 매년 상승률을 계산하는 문장들을 쓰고,
print 출력문을 사용하여 모든 상승률을 %로 출력하시오
(3) 최근 10년간 평균 인상률을 계산하여 출력하시오
(4) 위 3가지 문제를 해결하면서 주석문(Comment)을 적절히 추가하세요

(5) 년도별 시간당 임금을 미리 프로그램에 입력하지 말고 input 명령어를 이용하여
키보드를 이용하여 10개의 값을 입력 하시오
(6) 위 (1)번 (5)번 입력 방식의 장단점을 설명하시오
(7) 위 (5)번에서처럼 input 명령어를 사용하여 값을 입력할 때 어 떤 주의할 점이 있는지 설명하고,
필요하면 문제가 생기지 않도록 프로그램을 변경하시오

(8) 위 (2)번 방식으로 상승률을 계산하면 어떤 점이 불편한지 설명하시오
(9) 가장 인상률이 높은 해(년도)와 그 때의 인상률 구하려면 어떤 명령어가 필요할까요?
(10) 앞으로 20년이 더 지나서 30년간의 자료를 처리할 때 어떤 불편한 점이 있나요?
(11) 한국의 최저임금이 다른 OECD 국가의 최저임금에 비하여 높은지 아니면 낮은지를
어떻게 알 수 있는지 적절한 비교 방법을 말로 설명하세요
(12) 위 (11)번에서 찾은 방법을 10개 나라에 대하여 비교할려면, 어떤 자료(데이터)가 필요할까요?
(13) 위 (11)번에서 찾은 방법을 10개 나라에 대하여 비교할려면, 파이썬 프로그램에서
어떤 기능의 명령어가 필요할까요?

(14) 최저임금제도가 무엇이고 왜 필요한지 설명하시오
(15) 최저임금제의 문제점이 있다면 문제점과 이유를 설명하시오

8. 여러개의 자료 다루기 - (1) 리스트(list)와 메쏘드/함수

- 자료구조
  - 다양한 종류의 자료를 쉽고 효율적으로 다루기 위하여 프로그래밍 언어에서 제공(허용)하는 자료형태
  - 정수, 실수, 문자열, 리스트, 튜플, 집합, 사전
- 리스트
  - 여러개의 데이터 처리를 편하게 만들기 위하여, 여러개의 자료를 하나의 이름(변수명)으로 표현한 것
  - 리스트 생성 기본: 리스트명 = [요소1, 요소2, 요소3, ...]
  - 리스트를 사용하면 파이썬이 기본적으로 제공하는 다양한 리스트 메쏘드(method)를 사용하여 코딩량을 줄이고 코딩을 정확하고 편하게 할 수 있다

  ※ 메쏘드: 특정 객체에 적용되는 함수로 *객체이름.메쏘드명(parameter)* 형태로 사용 (예: a.sort())
  함수 : 특정 객체와 무관하게 사용. 따라서 사용시 객체이름이 앞에 오지 않음 (예: max(a))
- 리스트 메쏘드와 함수

함수(메소드) * 가정 : a라는 리스트 변수임	설명
a.append(x)	리스트 a 의 맨 끝에 x변수 값을 추가
a.sort()	리스트 a를 오름차순으로 정렬함. a.sort(revese=True)하면 내림차순으로 정렬함
a.reverse()	a의 index값을 거꾸로 뒤집음.(반대로), 역 index를 이용해서, 우측(뒤쪽)에서 부터, 값을 가지고 오는 것과 동일한 결과
a.index(x)	리스트 a의 x index를 가져옴.
a.insert(i,x)	리스트 a의 i번째 인덱스에 x값을 추가
a.remove(x)	리스트 a의 처음 인덱스 x값을 삭제
a.pop()	리스트 a의 맨 마지막 index를 반환하고, 맨 마지막 요소(index)를 삭제함.
a.pop(x)	리스트 a의 x번재 요소를 출력하고, 해당 값을 삭제함.
a.count(x)	리스트 a안에 x가 몇개 있는지를 출력함.
a.extend(x)	리스트 a안에 리스트 x를 더함(리스트 확장)
len()	리스트 a의 index 갯수(배열 길이)를 출력함.
x in a	x값이 a라는 리스트에 있으면 true, 없으면 false
x not in a	x값이 a라는리스트에 없으면 true, 있으면 false
min(a)	리스트 a의 최소값이 있는 index를 표시
max(a)	리스트 a의 최대값이 있는 index를 표시

■ 기본 사용법과 리스트 관련 메쏘드/함수들

```
# 다양한 리스트의 생성
 a = []                          # 또는 a = list()        # 비어있는 리스트
 b = [1, 2, 3]                                          # 정수 데이터 리스트
 c = ['Life', 'is', 'too', 'short']                     # 문자열 데이터 리스트
 d = [1, 2, 'Life', 'is']                               # 정수, 문자열 데이터 리스트
 e = [1, 2, ['Life', 'is']]                             # 리스트가 원소인 리스트

# 리스트 요소의 접근과 계산
temp = b[1] + b[2]
print ( c[-1] )
f = [1, 2, 3, ['a', 'b', 'c']]
print( f[-1])                                           # ['a', 'b', 'c']
print( f[-1][1] )                                       # 'b'

# 리스트 슬라이싱
g = [1, 2, 3, 4, 5]
print( g[0:2] )                                         # [1, 2]
print( g[:3] )                                          # [1, 2, 3]
print( g[2:] )                                          # [3, 4, 5]

# 인터프리터에서 예제
>>> a = [1, 2, 3, ['a', 'b', 'c'], 4, 5]
>>> a[2:5]
[3, ['a', 'b', 'c'], 4]
>>> a[3][:2]
['a', 'b']

# 리스트 더하기
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> a + b
[1, 2, 3, 4, 5, 6]
```

```
# 리스트 반복하기
>>> a = [1, 2, 3]
>>> a * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]

# 리스트 길이 구하기
>>> a = [1, 2, 3]
>>> len(a)
3

# 리스트 값 수정하기
>>> a = [1, 2, 3]
>>> a[2] = 4
>>> a
[1, 2, 4]

# 리스트 특정 요소값 삭제하기
>>> a = [1, 2, 3, 4, 5]
>>> del a[1]
>>> a
[1, 3, 4, 5]
>>> del a[2:]
>>> a
[1, 2]

# 리스트에 값 추가하기(append)
>>> a = [1, 2, 3]
>>> a.append(5)
>>> a
[1, 2, 3, 5]
>>> a.append([6,7])
>>> a
[1, 2, 3, 5, [6, 7]]

# 리스트에서 요소 가져오고 리스트에서 삭제하기 - append()의 반대
>>> a = [1,2,3]
>>> a.pop()                          # 인덱스 생략시 마지막 요소 가져오고 삭제
3
>>> a
[1, 2]
>>> a.pop(0)
1
>>> a
[2]
```

```
# 리스트 값의 순서대로 '영구' 정렬하기          반면, sorted() 함수는 '일시적' 정렬
>>> a = [1, 4, 3, 2]
>>> a.sort()                                  >>> b = sorted(a)
>>> a
[1, 2, 3, 4]

# 위치(값이 아닌)를 역순으로 정렬
>>> a = ['a', 'c', 'b']
>>> a.reverse()            #  a.sort(reverse=True) : 값을 기준으로 역순으로 정렬
>>> a
['b', 'c', 'a']

# 위치(인덱스) 반환
>>> a = [1,2,3]
>>> a.index(3)
2
>>> a.index(1)
0

# 특정 위치에 값 삽입
>>> a = [1, 2, 3]
>>> a.insert(0, 4)
>>> a
[4, 1, 2, 3]

# 리스트에서 첫 번째 해당값 제거  - remove는 인덱스를 사용, 반면 remove는 값을 사용
>>> a = [1, 2, 3, 1, 2, 3]
>>> a.remove(3)
>>> a
[1, 2, 1, 2, 3]

# 리스트에서 특정값의 출현 빈도 세기
>>> a = [1,2,3,1]
>>> a.count(1)
2

# 리스트의 확장
>>> a = [1,2,3]
>>> a.extend([4,5])
>>> a
[1, 2, 3, 4, 5]
>>> b = [6, 7]
>>> a.extend(b)
>>> a
[1, 2, 3, 4, 5, 6, 7]
```

9. 리스트(list) 실습문제 - 최저임금

※ 아래 실습의 목표는 다음과 같다

[1] 리스트의 기본 사용법 이해

[2] 단순 변수를 사용할 때와 리스트와 같은 자료구조+메쏘드+함수를 사용할 때의 차이점의 이해

[3] 논리적 사고력과 프로그래밍 언어를 이용한 문제 해결 경험

2019~2010 연도별 최저임금액								
적용년도	적용기간	시간급	일급 (8시간 기준)	월급 (209시간 기준, 고시기준)	인상액	인상률	심의 의결일	결정 고시일
2019	19.1~19.12	8,350	66,800	1,745,150	820	10.9%	18.7.14	18.8.3
2018	18.1~18.12	7,530	60,240	1,573,770	1060	16.4%	17.7.15	17.8.4
2017	17.1~17.12	6,470	51,760	1,352,230	440	7.3%	16.7.16	16.8.5
2016	16.1~16.12	6,030	48,240	1,260,270	450	8.1%	15.7.9	15.8.5
2015	15.1~15.12	5,580	44,640	1,166,220	370	7.1%	14.6.27	14.8.4
2014	14.1~14.12	5,210	41,680	1,088,890	350	7.2%	13.7.5	13.8.2
2013	13.1~13.12	4,860	38,880	1,015,740	280	6.1%	12.6.30	12.8.1
2012	12.1~12.12	4,580	36,640	957,220	260	6.0%	11.7.13	11.8.1
2011	11.1~11.12	4,320	34,560	902,880	210	5.1%	10.7.3	10.8.3
2010	10.1.~10.12	4,110	32,880	858,990	110	2.8%	09.6.30	09.8.3

지난 10년간(2019-2010) 최저임금 인상률

(1) 2010년부터 2019년 까지의 년도별 시간당 임금(시간급)을 리스트 변수(객체)를 선언하여 저장하시오
(변수명을 만들 때 camel형으로 하세요)

(2) 최저임금이 가장 작은 경우, 가장 큰 경우, 그리고 10년간 몇 % 인상되었는지 출력하시오

(3) 2020년도 최저임금을 검색하고, 위 (1)번에서 정의한 리스트에 메쏘드를 이용하여 추가하시오

(4) 파이썬에는 평균을 계산하는 함수가 있는지 검색해서 확인해보시오

(5) sum() 과 len() 함수를 이용하여 최저임금의 11년간 평균을 계산하는 코드를 작성하시오

(6) 위 (5)번에서 len()함수를 이용하면 어떤 좋은점이 있는지 생각해보시오

(7) 함수/메쏘드를 사용하지 않았던 지난번 코딩 방식과, 이들을 사용한 코딩 방식의 차이점에 대하여 설명하시오

(8) minWages = [4110, 4320, ... 8350]
yearWages = [2010, 2011, ... 2019] 와 같이 년도와 최저임금값을 리스트에 각각 저장하면,
키보드로부터 년도를 입력하면, 입력한 년도의 최저임금값을 출력하게 코딩하시오.
그리고, 저장해두지 않은 년도를 입력하면 어떤일이 일어나는지 확인하시오

(9) 인상률 10개를 저장하는 리스트 객체(변수)를 만들고, 리스트 메쏘드를 이용하여
인상률이 높은 값부터 낮은값 순으로 출력하시오

(10) 위 (1)~(9)번에서 제시한 내용외에 코딩하고싶은 내용이 없는지 생각해보시오

10. 프로그램 실행 흐름의 제어

- 프로그램의 실행흐름 제어
  - 프로그램에서 문장(들)의 실행여부, 실행순서, 실행횟수를 조절하여 원하는 결과를 얻도록 프로그램의 동작(흐름)을 제어하는 것
- 프로그램의 논리적 흐름 기본 3가지
  - 순차적(Sequential) : 문장이 쓰여진 순서대로 실행.
    한번씩 실행 (1)
  - 조건적(Conditional) : 조건의 만족(참,거짓) 여부에 따라 실행 문장(들)의 대상이 변경
    실행되지 않거나 실행 되거나 (0/1)
  - 반복적(Repeatetive) : 문장(들)을 일정 횟수나 조건을 만족하는 동안 반복적으로 실행
    여러번 실행 (0~n)

- 순차적 흐름 제어:
  - 먼저 실행되어야 하는 문장을 나중에 실행되어야 하는 문장보다 먼저(위에, 앞에) 기술
  - 조건적, 반복적 문장을 제외한 모든 문장에 해당. 위에서 아래로 실행
  - 어떤 문장을 먼저 실행해야 하는가는 논리적 사고로 결정

- 조건적 흐름 제어: if
  - 특정 조건이 맞을 때(혹은 맞지 않을 때)만 실행되게 끔
  - 조건문은, 값의 대소를 비교하거나 비교 결과를 논리적으로 연결한 결과 값이 참(True)인지 판단
  - 비교 연산자 : < > == != >= <=
    ex) if x <= y :
  - 논리 연산자 : and or not
    ex) if x < 0 **or** x > 100 :
    if **not** isHuman :
  - if 문장의 기본 구조 4가지

# [1] 특정 조건이 맞을 때의 한가지 경우에만 실행 	
if 조건문:
  수행할 문장1
  수행할 문장2
  …
	birthYear = int(input("출생년도: "))
	if birthYear < 1900 :
	  print("1900보다 큰 양수를 입력하세요")
	  birthYear = int(input("출생년도: "))
	
	# 입력값이 올해보다 큰 경우 오류

# [2] 특정 조건이 맞을 때와 그렇치 않은 모든 경우를 사용 	
if 조건문:
  수행할 문장1
  수행할 문장2
  …
else :
  수행할 문장3
  수행할 문장4
  …
	subjectNo = 5
	score = int(input("총 점수: "))
	if score >= 0 and score <= 100 :
	  print("평균 = ",score/subjectNo)
	else :
	  print("0~100 사이의 총점을 입력하세요")
	  score = int(input("총 점수: "))
	
	# 어떤 문제가 있을까요?

```
# [3] 서로다른 조건이 여러개일 때
if 조건문:
    수행할 문장1
    수행할 문장2
    ...
elif :
    수행할 문장3
    수행할 문장4
    ...
elif :
    수행할 문장5
    수행할 문장6
    ...
```

```
subjectNo = 5
score = int(input("총 점수: "))
average = score/subjectNo
if average >= 90 :
    hakjum = 'A'
elif average >= 80 :
    hakjum = 'B'
elif average >= 70 :
    hakjum = 'C'
elif average >= 60 :
    hakjum = 'D'
elif average >= 0 :
    hakjum = 'F'

# 어떤 문제가 있을까요?
```

```
# [4] 서로다른 조건이 여러개이고 예외의 경우도 고려할 경우
if 조건문:
    수행할 문장1
    수행할 문장2
    ...
elif :
    수행할 문장3
    수행할 문장4
    ...
elif :
    수행할 문장5
    수행할 문장6
    ...
else :
    수행할 문장7
    수행할 문장8
    ...
```

```
subjectNo = 5
score = int(input("총 점수: "))
average = score/subjectNo

if average > 100 :
    print("학점이 100점 초과 !!!")
elif average >= 90 :
    hakjum = 'A'
elif average >= 80 :
    hakjum = 'B'
elif average >= 70 :
    hakjum = 'C'
elif average >= 60 :
    hakjum = 'D'
else :
    hakjum = 'F'
```

■ 반복적 흐름 제어 (1) : while

- 특정 조건이 맞을 때 문자(들)을 반복 실행

```
# [1] 특정 조건이 맞을 때 원하는 문장들을 실행

while 조건문:
    수행할 문장1
    수행할 문장2
    ...
```

```
# 1 + 2 + 3 + ..... + n = ?

lastNo = int(input("누적 합계할 수 입력: "))
number = 1; sum = 0
while number <= lastNo :
    sum = sum + number
    number = number + 1
    print("부분 합: ",sum)
```

```
# n + (n-1) + (n-2) + ..... + 2 + 1 = ?

lastNo = int(input("누적 합계할 수 입력: "))
if lastNo > 0 :
    number = lastNo; sum = 0
    while number > 0 :
        sum = sum + number
        number = number - 1
else :
    print ("0 보다 큰 수를 입력 하시오")
```

```
# 1 * 2 * 3 * ..... * n = ?

lastNo = int(input("누적 곱할 수 입력: "))
number = 1; sum = 1
while number <= lastNo :
    sum = sum * number
    number = number + 1
    print("부분 곱의 합: ",sum)
```

- 무한반복과 특정 조건이 맞을 때 반복을 벗어나기

```
# [1] 무한 반복

while True :
    수행할 문장1
    수행할 문장2
    ...
```

```
# 0을 입력 전까지 원의 면적 무한정 구하기
import math

#print(math.pi)
while True :
    radius = int(input("반지름 값 입력: "))
    if radius == 0 :
        break
    elif radius < 0 :
        print ("양의 반지름값을 입력하세요")
        continue
    circleArea = math.pi * (radius ** 2)
    print("원의 면적 = ", circleArea)
print ("종료")
```

※ 조건문, 반복문(while) 연습문제

(1) 정수 2개 (firstNo, secondNo)를 키보드로부터 입력 받아서, 둘 중 작은 수부터 큰 수 사이의 모든 자연수를 더하는 프로그램을 작성하시오
예) 1, 10을 입력하면 55가 출력되도록 하시오

(2) 위 (1)번 프로그램에서, 두 수 가운데 큰 수가 먼저 입력되어도 문제없이 동작하도록 작성하였는지 확인하고, 필요하면 문제없이 동작하도록 변경하시오

(3) 위 (2)번 프로그램에, 입력한 두 값 사이의 짝수 값만 모두 더하는 프로그램을 추가하시오.

(4) 2010년에서 2019년 까지 10년간 최저 시간급을 list에 저장한 후, while 명령어와 리스트의 append 메쏘드를 이용하여 매년 인상률을 구하여 또다른 리스트에 저장하시오

(5) max() 함수를 사용하지 말고, 위 (4)번에서 구한 인상률 리스트와 if, while문을 이용하여 인상률이 가장 컸던 해의 인상률을 출력하시오

(6) sum() 함수를 사용하지 말고, 위 (4)번에서 구한 인상률 리스트와 while문을 이용하여 평균 인상률을 구해서 출력하시오

(7) 위 (6)번에서 구한 평균인상률 보다 인상률이 큰 해의 최저시간급과 인상률을 출력하시오

■ 반복적 흐름 제어 (2) : for 문장은 나중 학습으로 미룸
(for 없이도 어떤 반복문도 while로 구현 가능. 반대의 경우도 마찬가지임)

11. 함수를 이용한 프로그래밍 간소화와 코드 재사용

- 함수란
  - 함수: 기능적/논리적으로 하나의 논리적 실행단위로 분리하여 생각할 수 있는 "논리적" 코드 블록
  - 파이썬에서 제공하는 "내장함수"와 사용자가 필요에 따라 만든 "사용자 함수"로 크게 구분
  - 함수 정의(Definition)를 통하여 실행할 코드를 기술하고, 함수 호출(Function Call)을 통하여 사용
- 함수 정의와 호출의 기본
  - argument(인수)로 함수에 넘길값을 지정하고, parameter(매개변수)로 함수가 값을 넘겨 받음
  - argument와 인수는 필요에 따라 생략 가능
  - return 문장의 의미와 실행 흐름
  - 변수의 유효범위(scope rule): 전역변수(Global Variable)와 지역변수 (local Variabe)
  - global 키워드 : 함수안에서 전역변수 지칭시 사용, 가급적 사용하지 않는 것이 좋으나 꼭 필요시 사용

```
# 함수 정의

def 함수명 (parameter) :
    수행할 문장1
    수행할 문장2
    ...
    return 리턴값     # 생략가능

함수명(argument)

#---------------------------------
# 아래 프로그램을 입력하여 실행하시오
#---------------------------------

#---------------------------------
# 함수 정의 및 호출의 예 (1)
import math
def is_odd_even (number) :
    if (number % 2) == 0 :
        return '짝수'
    else :
        return '홀수'
#---------------------------------
# 함수 정의 및 호출의 예 (2)
def factorial (n) :
    result = 1
    while n >= 1:
        result = result * n
        n = n - 1
    return result
```

```
#-------------------------------------
# 함수 정의 및 호출의 예 (3)
def area_volume(r) :
    circle_area = math.pi * (r **  2)
    sphere_volume = 4/3 * math.pi * (r **  3)
    return circle_area, sphere_volume

#-------------------------------------
# 함수 정의 및 호출의 예 (4) - return문장 없음
def no_return (number) :
    if (number % 2) == 0 :
        rtn_val = '짝수'
    else :
        rtn_val = '홀수'
    print (rtn_val)
    return

no = int(input('정수 입력'))

print (no, '는 ', is_odd_even(no), '입니다')
fact_val = factorial(no)

radius = no
area, volume =  area_volume (radius)
print ("원의 면적:", area, "구의 체적:", volume)

no_return(no)

cnt = 1
while cnt <= 20 :
    print (no,'! = ', factorial(no))
```

- 함수를 사용할 때의 장단점
  - 함수를 호출하는 부분에서는 전체 실행코드 블록이 한줄의 함수 호출문장으로 줄어들므로 코드가 간단해 지는 효과가 있다
  - 한번 잘 만들어 놓은 함수를 나중에 누군가 재사용할 수 있다
  - 함수 사용시 실행시간이 길어질 수 있으나, 일반적인 경우는 큰 이슈는 아님
- argument가 여러개일 때, 매개변수에 초깃값 설정, 키워드파라미터, lambda함수는 나중 학습으로 미룸

※ 함수 연습문제

(1) 정수 2개 (firstNo, secondNo)를 키보드로부터 입력 받아서, 둘 중 작은 수부터 큰 수 사이의 모든 자연수를 더한 후 결과값을 return 하는 함수 add_all()을 작성하시오

(2) 위 (1)번 프로그램에 이어, 두 자연수 중 하나라도 음수를 입력하면 종료하고 그렇지 않으면 무한정 위 add_all() 함수를 호출해서 더하는 일을 계속하도록 프로그램을 완성하시오

(3) 정수 2개 입력받는 부분을 add_all()함수 안에 넣는 것이 좋을지, 아니면 함수 호출하는 곳에 위치시키는지 좋을지에 대하여 자신의 생각을 말하시오

(4) 함수 이름은 어떻게 짓는지 좋을지 말하시오

(5) 함수를 실행하면 어떤 장점과 단점이 있는지 말하시오

(6) 함수를 사용하지 않고는 절대 코딩할 수 없는 문제가 있을지 생각해보시오

(7) 같은 이름의 함수가 있으면 어떤 문제가 있을지 답하시오

(8) 전역변수를 많이 사용하는게 좋을지 생각해보시오

(9) 지역변수와 전역변수를 구분짓는 기준이 무엇인지 설명하시오

(10) 전역변수와 지역변수의 이름이 같을 경우, 함수 안과 밖에서 어떤 변수가 사용될지 설명하시오

(11) 하나의 함수안에 넣을 코드의 적절한 내용/길이를 어떻게(무슨 기준으로) 정해야 할까요?

12. 파일을 이용한 보조기억장치에 데이터 저장 및 파일 읽기

- 파일
  - 보조기억장치에 저장된 데이터로, 전원공급에 관계없이 데이터를 저장할 때 사용
  - 사용할 때는 프로세스(Process)간의 자원(Resource) 동시사용에 따른 문제를 방지하기 위하여 반드시 열기(Open)를 먼저 해서 운영체제로부터 사용권한을 얻은 다음에 사용하고, 사용 후에는 다른 프로세스가 필요시 사용할 수 있게 운영체제에게 사용완료를 알리기 위해 닫기(Close) 로 종료해야함

- 파일 열기 모드(mode)
  - 파일을 열 때는 사용 목적에 따라 다음의 열기 모드를 알려 준다.

파일열기모드	설명
r	읽기모드 - 파일을 읽기만 할 때 사용
w	쓰기모드 - 파일에 내용을 쓸 때 사용
a	추가모드 - 파일의 마지막에 새로운 내용을 추가 시킬 때 사용

```
fp = open ("text.txt", "r")        # fp : 파일객체, "r" 텍스트 읽기 모드
```

  - 파일은 이진파일과 텍스트파일로 나뉜다. 이진 파일은 \n처럼 제어문자를 따로 인정하지 않고 모두 그냥 문자로 간주할 때 사용한다. 이진파일을 읽을 때는 "rb" 등과 같이 mode 기호에 b 문자를 붙여서 사용하면 된다.

```
fp = open ("text.txt", "rb")
```

  - 관련 기본 함수 : read(크기), realine(), readlines(), write()
    읽는 동작의 용도에 맞는 함수를 골라서 사용하면 된다
  - 파일을 읽고 쓸 때 운영체제는 파일 내에서 현재의 위치를 나타내는 값을 사용한다
    파일을 열면 위치가 파일 내용의 처음에 위치하고, 파일 끝에 닿았는지 여부를 필요한 관련함수가 체크하고 그에 따라 동작한다
  - 파일 닫기를 잊지 않으려면 with...as 구문을 이용하면 좋다

- 파일에 쓰기와 읽기의 예 :

```
# [1] 파일에 덮어쓰기 (w mode)
fp = open("C:Wdata1.txt", 'w')              # 기존 파일 내용은 삭제
age = input("나이: ")
content = "나이는 " + age + " 세 입니다.Wn" # 문자열 쓰기, 줄 구분할려면 Wn 추가
fp.write(content)                           # 파일에 쓰기
if int(age) <= 70 :
    fp.write("청춘이시네요^^")              # 파일에 쓰기
fp.close()

# [2] 파일로부터 한 줄 읽기 (r mode)

fp2 = open("C:Wdata1.txt", 'r')
text1 = fp2.readline()                      # 파일에서 한 줄 문자열로 읽기
print("파일에서 읽어들인 첫째 줄: ",text1)  # readline 실행후에는 파일포인터가 다음줄로 이동
text2 = fp2.readline()                      # 파일에서 한 줄 문자열로 읽기
print("파일에서 읽어들인 둘째 줄: ",text2)
fp2.close()

# [3] 파일로부터 모든 줄 읽기 (r mode)

fp2 = open("C:Wdata1.txt", 'r')
text = fp2.readlines()                      # 파일에서 모든 줄을 읽어 문자열 리스트로 반환
print("파일에서 readlines() 읽어들인 모든 줄: ",text)
fp2.close()

fp2 = open("C:Wdata1.txt", 'r')
text = fp2.read(2)                          # 파일에서 문자 2개만 읽어 문자열로 반환
print("파일에서 read(2)로 읽어들인 문자열: ",text)

text = fp2.read()                           # 파일에서 나머지 모든 줄을 읽어 문자열로 반환
                                            # 파일 끝에 다다르면 빈문자 리턴
print("파일에서 read()로 나머지 모두 읽어들인 줄: ",text)
fp2.close()
```

```
# [4] 기존 파일 끝에 추가하여 쓰기 (a mode)

fp2 = open("C:Wdata1.txt", 'a')
text = fp2.write("bye bye Wn")                # 파일끝에 추가하여 쓰기
fp2.close()

# [5] with as 로 close() 생략하기
#   with 블록이 끝나면 자동으로 close() 함수 호출해 줌
with open("C:Wdata1.txt", 'a') as fp2 :
    text = fp2.write("안녕")
```

- 생성된 파일 :

- 실행 결과 :

```
나이: 65
파일에서 읽어들인 첫째 줄:  나이는 65 세 입니다.

파일에서 읽어들인 둘째 줄:  청춘이시네요^^
파일에서 readlines() 읽어들인 모든 줄:  ['나이는 65 세 입니다.Wn', '청춘이시네요^^'
]
파일에서 read(2)로 읽어들인 문자열:  나이
파일에서 read()로 나머지 모두 읽어들인 줄:  는 65 세 입니다.
청춘이시네요^^
>>>
```

※ 파일 연습문제

(1) 아래 표는 2020년 12월 12일자의 한국내 코로나19 확진자 수의 8개 영역별 발생 숫자이다. 지역명, 총 확진자수, 신규확진자수 정보를 키보드로부터 읽어 들인 후, "corona19.txt" 이름의 텍스트 파일에 저장하시오.

※ 지역별 데이터를 한 줄에 모두 저장 하는게 좋을지, 아니면 지역명, 총확진자수, 신규확진자수를 각각 별도의 줄에 저장하는 게 좋을지 생각하시오

지역	총확진자	신규확진자
서울	12,187	399 ▲
경기	9,737	331 ▲
대구	7,349	28 ▲
검역	2,273	14 ▲
인천	1,841	62 ▲
경북	1,834	18 ▲
부산	1,235	57 ▲
충남	1,064	10 ▲

(2) 위 (1)번에서 저장한 "corona19.txt" 파일을 연 후 읽어 들여서, 각 지역별 확진자 관련 데이터를 숫자로 변환하여 다음을 계산하여 출력하시오

a. 신규 확진자 수의 총합, 가장 신규확진자가 많은 도시명과 신규확진자 수를 출력하시오

b. 총 확진자 대비 신규확진자 수의 비중이 가장 큰 도시명을 출력 하시오

(3) 위 2번에서 출력한 a, b 문제의 답을 파일 "analysis.txt" 파일에 저장하시오

(4) 위 (3)번에서 저장한 "analysis.txt" 파일을 읽어서 그 내용을 화면에 출력하시오

(5) 파일에 저장을 하면 어떤 좋은 점이 있는지 설명하시오

(6) 파일에 저장을 할 때 저장되는 데이터에 대하여, "어떤 형태나 순서로 저장하는 것이 좋은지"를 결정하는 기준에 대하여 설명하시오.

(7) 위 (1)~(6)번의 답을 아래 한글 파일이나 pdf로 저장하여 아래 제출방법대로 제출하시오

- 제출 방법 : pkkim@silla.ac.kr 메일로 제출
- 메일 제목 : AI융합-파일과제-근무학교명-이름
- 제출 기한 : 12월 24일 정오

13. for 반복문

- while과 더불어 반복을 수행하는 명령어
  - 리스트와 같은 자료구조에 대하여 반복을 실행할 때 편리하기 때문에 자주 사용됨

```
# 기본 사용 구조
for 반환받을 변수 in 리스트(또는 튜플, 문자열):
    수행할 문장1
    수행할 문장2
    ...
#-----------------------------------------------------------------
# 기본사용 예 (1)
dayList = ['sun', 'mon', 'tue', 'wed', 'thur', 'fri', 'sat']
for day in dayList:
    print(day)
# 결과
sun
mon
# 중간 생략
sat
#-----------------------------------------------------------------
# 기본사용 예 (2), range()와 같이 사용
sum = 0
for i in range(1, 11):
    sum = sum + i
print(sum)
# 결과
55
#-----------------------------------------------------------------
# 기본사용 예 (3)
dataList = [(1,2), (3,4), (5,6)]
for (first, last) in dataList:
    print(first + last)
# 출력
3
7
11
#-----------------------------------------------------------------
# for 문에서 continue
scores = [90, 50, 70, 40, 80]

no = 0
for score in scores:
    no = no + 1
    if score < 60:
        continue
    print("%d번 학생 축하합니다. 합격입니다. " %no)
```

14. 더 공부해야 할 주제들

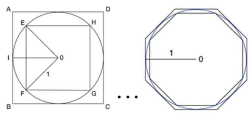
- 튜플, 딕셔너리, 세트(집합), 문자열
- 클래스
- 모듈과 패키지
- 예외처리

15. 프로그래밍 (언어) 학습 시, 중요한 것은

- 문법이 아니라 무엇을(what) 왜(why) 만드는지를 찾아서,
- 직접 코딩하면서 얻는 지식이 중요하다!

16. 겨울 방학 때 학습할 내용 추천

- 14장의 내용을 wikidocs 사이트에서 공부하고
- CNN을 이용한 딥러닝 기법을 사용하여 마스크 착용 여부 인식 프로그램을 작성 및 수행 해보기
  (1) 인터넷 검색을 통하여 anaconda 개발환경 설치
  (2) http://ailab.silla.ac.kr/lec/python/ 사이트의 “6. 딥러닝 마스크 착용여부 탐지 (Source)” 학습

	내용	질문
(1) 주제	반복문을 이용한 원주율 π 계산과 오차	
(2) 기본 코딩 주제	언어 지정 상수(π)의 사용법 무한 반복문 사용법 반복문에서 빠져 나오는 방법	
(3) 응용 코딩 주제	소수점 정밀도와 오차의 중요성 소수점 이하 정밀도 사용 기준은?	
(2) 관련 교과목	수학	
(3) 관련 학년	중학교 * 학년	
(4) 기본 정의	지름과 원둘레의 비율 (원둘레/지름) '둘레'를 뜻하는 그리스어 'περιμετροζ'의 머리글자로 18세기 스위스의 수학자 오일러가 처음 사용했다.	
(5) 사용 및 특징	반지름 r인 원둘레(2π r), 원의 넓이 (π r^2), 구의 부피(4/3π r^3), 구의 겉넓이 (4π r^2) e의 계산에 사용 무리수, 초월수, 아르키메데스 상수, 비순환 무한소수	
(6) 관련 자료	https://ko.wikipedia.org/wiki/원주율	
(7) 고전적 방법/ 언플러그드	(1) 실을 이용한 방법 (2) 줄자를 이용한 방법 (3) 원통형 바퀴를 이용한 방법 (4) 아르키메데스의 정96각형 근사법 :  3.1408..... < π < 3.1428.....	-측정 오차의 문제 -오차의 종류
(8) 코딩 방법들	(1) 파이썬 지정 상수 PI 사용 (2) 라이프니쯔 공식 π = 4 (1/1 - 1/3 + 1/5 - 1/7 + 1/9 .....)	- 코딩을 이용한 π 계산의 장점
(9) 응용 코딩	(1) 파이썬에서 정의한 π 상수의 소수점 이하 정밀도는 몇자리 (2) 나노과학에서 π 값 정밀도(오차)가 미치는 영향 문제 (3) 천문/물리학에서 π 값 정밀도(오차)가 미치는 영향 문제 (4) 소수점 이하 n자리까지 계산법	
(10) 참고	(1) 2016년 11월 11일 스위스의 입자 물리학자인 페터 트뤼프(Peter Trüb)는 105일 동안 계산하여, 원주율을 소수점 이하 22조 4591억 5771만 8361자리(π ^{e}조 자리) 까지 계산 : 3.1415926535..... (2) 기상예보, 인공위성 계산 등에는 소수점 아래 30자리 사용	