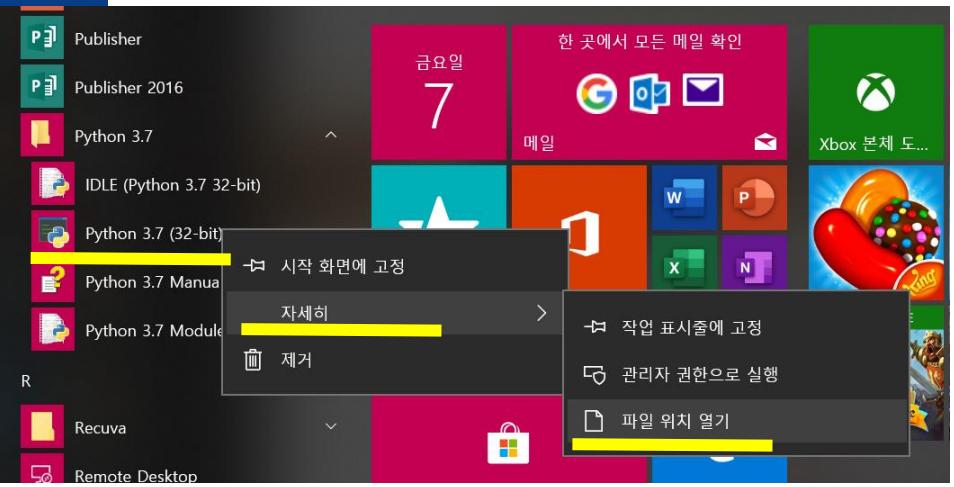
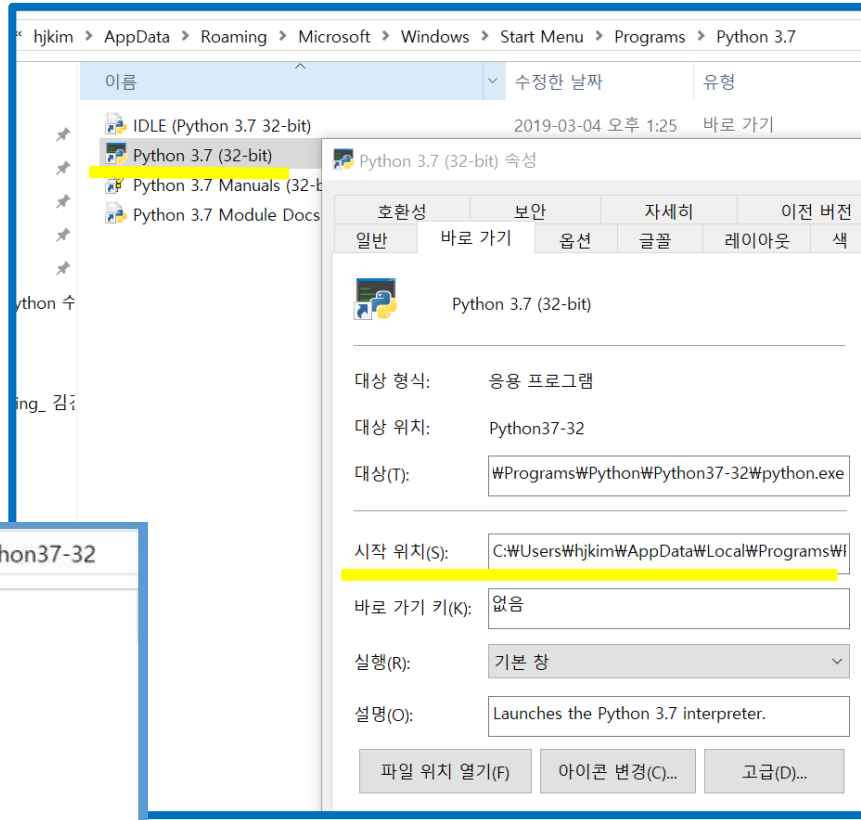
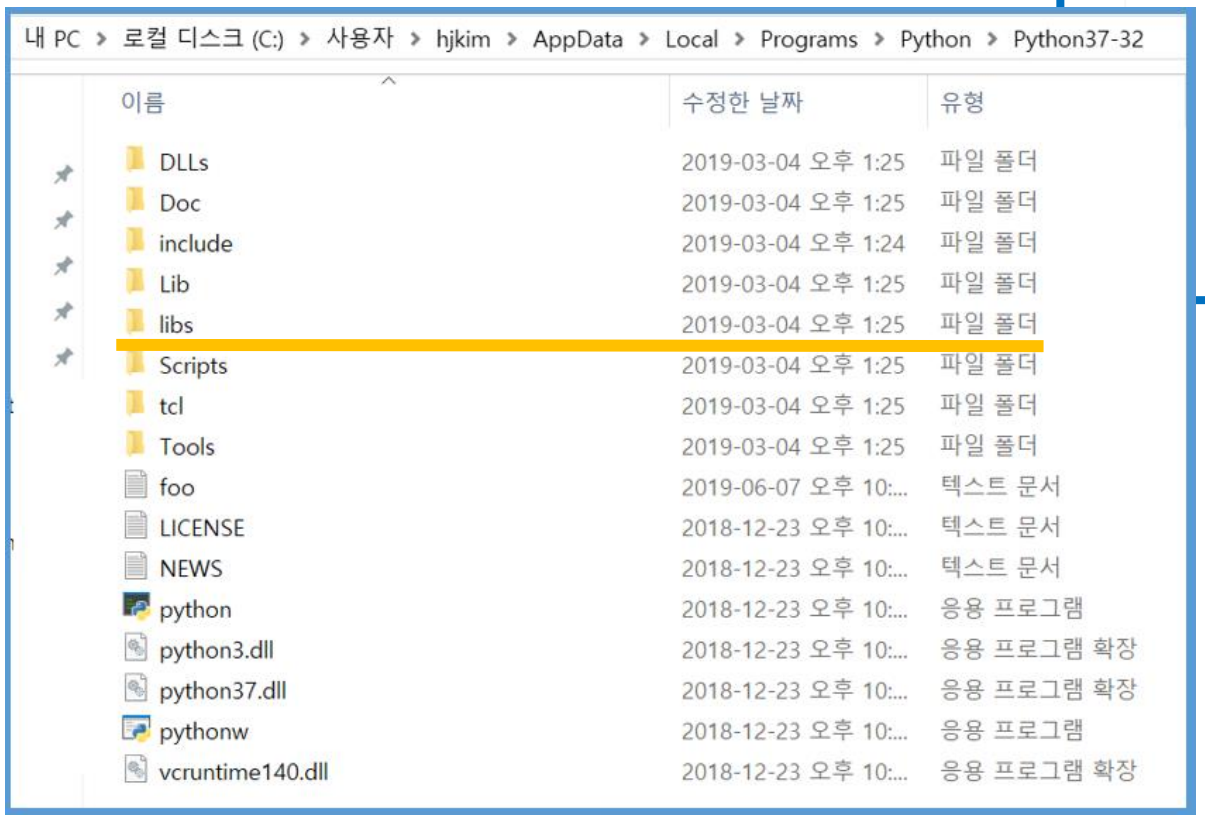
Ch 15: Math, Statistics, Random Modules

- **math module**
- **statistics module**
- **random module**

Standard Library의 위치 [1/2]

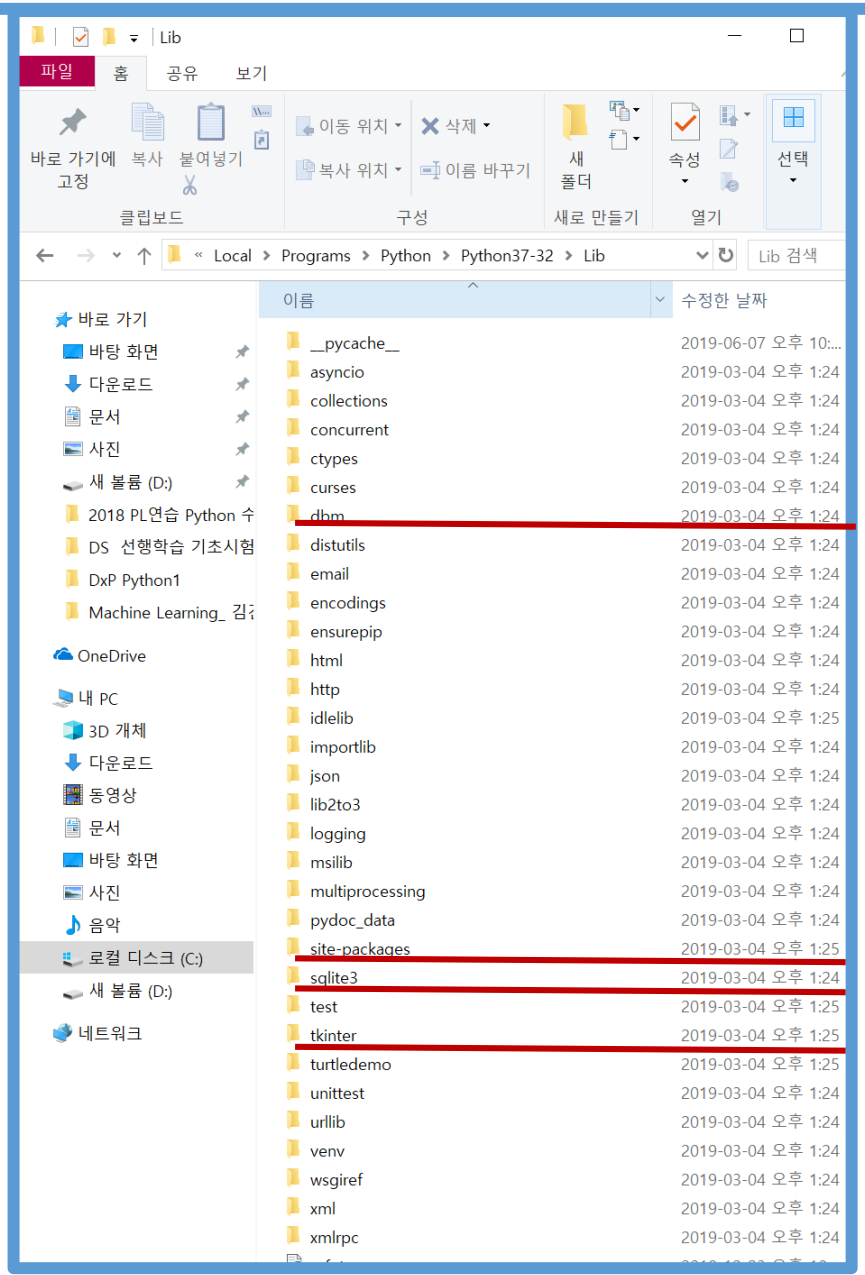






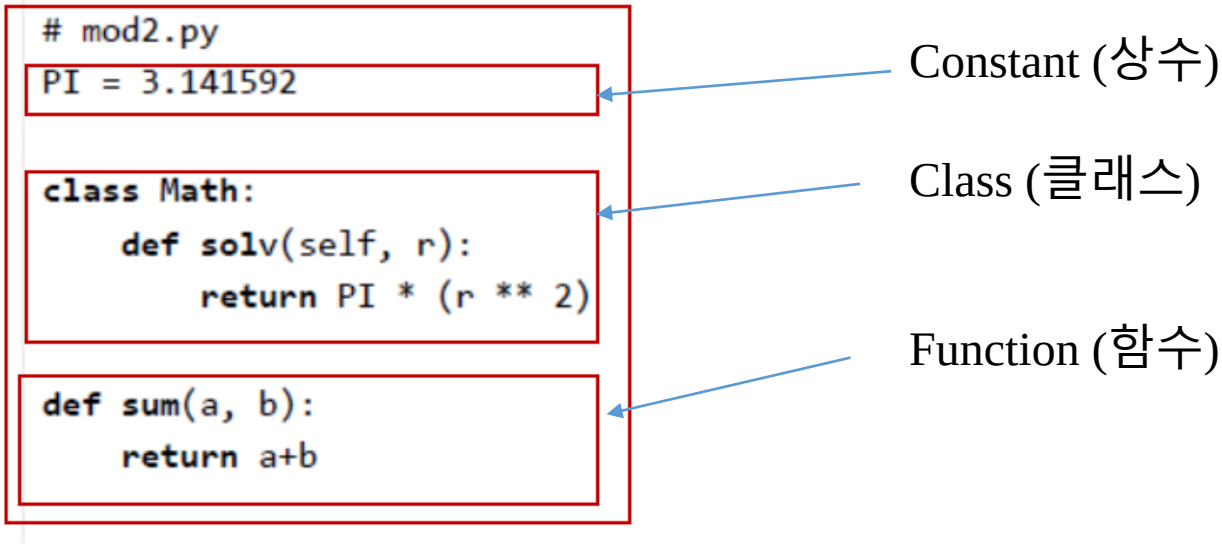
Standard Library의 위치 [2/2]

내 PC › 로컬 디스크 (C:) › 사용자 › hjkim › AppData › Local › Programs › Python › Python37-32 › Lib



Standard Library 들중에 많은것은 기존의 C library 를 이용하는것이 많이 있다!

Library (Module) 속의 구성원들



1. >>> import mod2
   >>> result = mod2.sum(2, 3) + mod2.PI

2. >>> from mod2 import sum
   >>> result = sum(2, 3) + mod2.PI

3. >>> from mod2 import *
   >>> result = sum(2, 3) + PI

“math” Module

- This module provides access to mathematical functions defined in C standard
  - These functions cannot be used with complex numbers (Use cmath)

- These are the categories of the functions
  - Constant
  - Number-Theoretic Functions
  - Power and Logarithmic Functions (지수 로그 함수)
  - Trigonometric Functions (삼각함수)
  - Angular Functions (도 함수)
  - Hyperbolic Functions (쌍곡선함수)

“math” Module : Constants

`math.pi`

The mathematical constant π = 3.141592…, to available precision.

`math.e`

The mathematical constant e = 2.718281…, to available precision.

`math.tau`

The mathematical constant τ = 6.283185…, to available precision. Tau is a circle constant equal to 2π, the ratio of a circle’s circumference to its radius. To learn more about Tau, check out Vi Hart’s video Pi is (still) Wrong, and start celebrating Tau day by eating twice as much pie!

New in version 3.6.

`math.inf`

A floating-point positive infinity. (For negative infinity, use `-math.inf`.) Equivalent to the output of `float('inf')`.

New in version 3.5.

`math.nan`

A floating-point “not a number” (NaN) value. Equivalent to the output of `float('nan')`.

“math” Module : Number Theoretic Functions

- math.**ceil**(x) : Return the smallest integer greater than or equal to x
- math.**floor**(x) : Return the largest integer less than or equal to x
- math.**fabs**(x) : Return the absolute value of x
- math.**factorial**(x) : Return the x factorial
- math.**fmod**(x,y) : Return the remainder of x divided by y
- math.**gcd**(x, y) : Return the greatest common denominator of x and y

```
import math

a = -3.123
b = 8
c = 3

print("ceil of a : ", math.ceil(a))
print("floor of a : ", math.floor(a))
print("fabs of a : ", math.fabs(a))
print("factorial of b : ", math.factorial(b))
print("fmod of b,c : ", math.fmod(b,c))
```

Result

```
>>>
ceil of a :  -3
floor of a :  -4
fabs of a :  3.123
factorial of b :  40320
fmod of b,c :  2.0
```

“math” Module : Power & Log Fuctions

- math.**log**(a, b) : Return the value of $\log_b a$
- math.**pow**(a, b) : Return the a raised to the power of b
- math.**sqrt**(a) : Return the square root of a

```
a = 2
b = 4
c = 25

print("log a b : ", math.log(b, a))
print("pow of a,b : ", math.pow(a,b))
print("sqrt of a : ", math.sqrt(c))
```

Result

```
>>>
log a b :  2.0
pow of a,b :  16.0
sqrt of a :  5.0
```

- math.**log2**(*x*) : Return the base-2 logarithm of *x*.
  This is more accurate than log(x, 2)

- math.**log10**(*x*) : Return the base-2 logarithm of *x*.
  This is more accurate than log(x, 10)

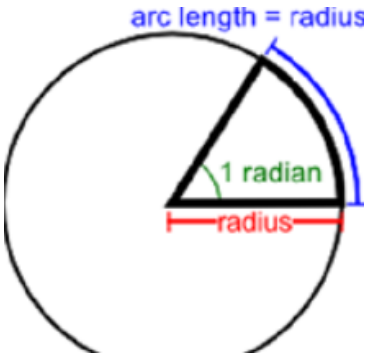
“math” Module : Angular Conversion

- math.**degrees**(x) : Convert angle x from radians to degrees
- math.**radians**(x) : Convert angle x from degrees to radians

```
print("degrees of pi : ", math.degrees(math.pi))
print("radians of 180 : ", math.radians(180))
```

Result

```
>>>
degrees of pi :  180.0
radians of 180 :  3.14159265
```



90^0 ➔ 0.5π = 0.5 * math.pi = 1.57 radian

180^0 ➔ π = math.pi = 3.14 radian

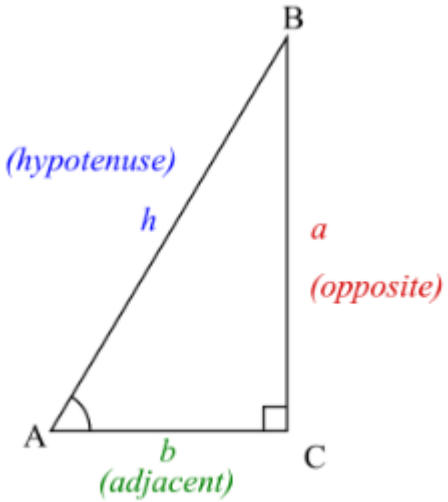
360^0 ➔ 2π = 2* math.pi = 6.28 radian

삼각함수의 parameter는 전부 radian으로 통일!

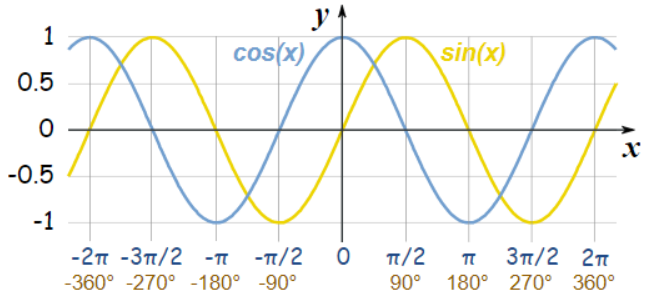
sin(90^0) ➔ sin(0.5 * math.pi)
sin(45^0) ➔ sin(0.25 * math.pi)

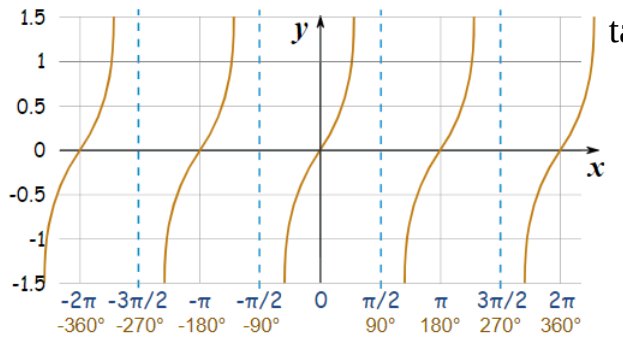
“math” Module : Trigonometric Functions [1/3]

- math.**sin**(x) : Return the sine value of x radians
- math.**cos**(x) : Return the cosine value of x radians
- math.**tan**(x) : Return the tangent value of x radians



Function	Description
sine	$\frac{\text{opposite}}{\text{hypotenuse}}$
cosine	$\frac{\text{adjacent}}{\text{hypotenuse}}$
tangent	$\frac{\text{opposite}}{\text{adjacent}}$





tan(x)

```
print("sin(pi) : ", math.sin( math.pi ))
print("cos(pi) : ", math.cos( math.pi ))
print("tan(pi) : ", math.tan( math.pi ))
```

Close to 0

Result

```
sin(pi) :  1.2246467991473532e-16
cos(pi) :  -1.0
tan(pi) :  -1.2246467991473532e-16
```

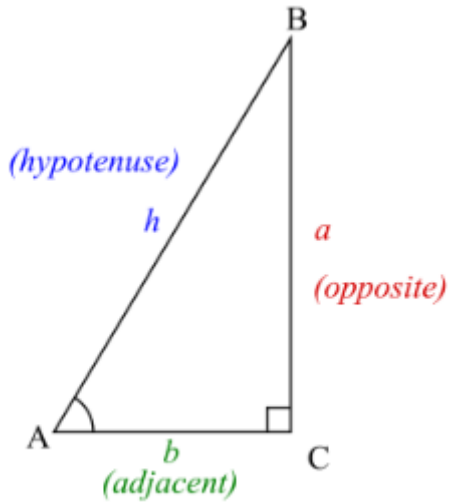
“math” Module : Trigonometric Functions [2/3]

- Reciprocals (역수) of Trigonometric Functions “math” module에는 없다!

secant(x) $$\sec A = \frac{1}{\cos A} = \frac{\text{hypotenuse}}{\text{adjacent}} = \frac{h}{b}$$

cotangent(x) $$\csc A = \frac{1}{\sin A} = \frac{\text{hypotenuse}}{\text{opposite}} = \frac{h}{a}$$

cosecant(x) $$\cot A = \frac{1}{\tan A} = \frac{\text{adjacent}}{\text{opposite}} = \frac{b}{a}$$



“math” Module : Trigonometric Functions [3/3]

- Inverse Trigonometric Functions (역삼각함수)

- math.**asin**(x) : Return the arc sine of x radians

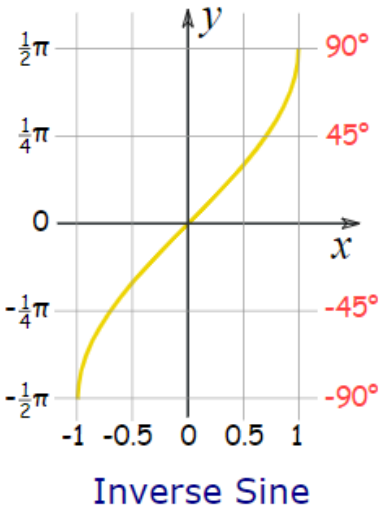
  y = sin(x) ➔ inverse function ➔ x = sin(y) ➔ y = arcsin(x)

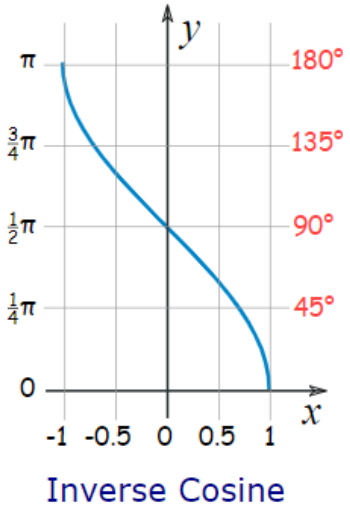
- math.**acos**(x) : Return the arc cosine of x radians

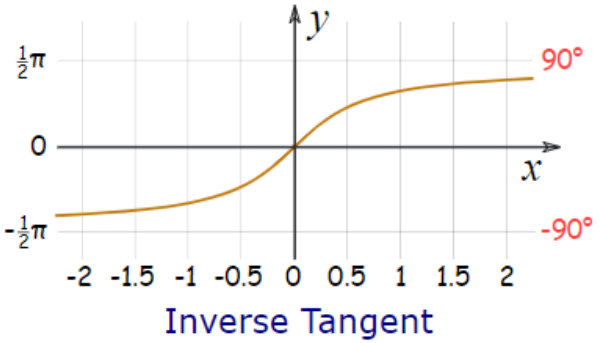
  y = cos(x) ➔ inverse function ➔ x = cos(y) ➔ y = arccos(x)

- math.**atan**(x) : Return the arc tangent of x radians

  y = tan(x) ➔ inverse function ➔ x = tan(y) ➔ y = arctan(x)







“math” Module : Hyperbolic Functions [1/2]

- Hyperbolic functions (쌍곡선 함수) are analogs of trigonometric functions that are based on hyperbolas instead of circles

삼각함수(*원함수*)의 사인, 코사인, 탄젠트 등에서 유추되어 각각에 대응되는 다음과 같은 함수가 있다.

- 쌍곡사인(hyperbolic sine)

$$\sinh x = \frac{e^x - e^{-x}}{2} = -i \sin ix$$

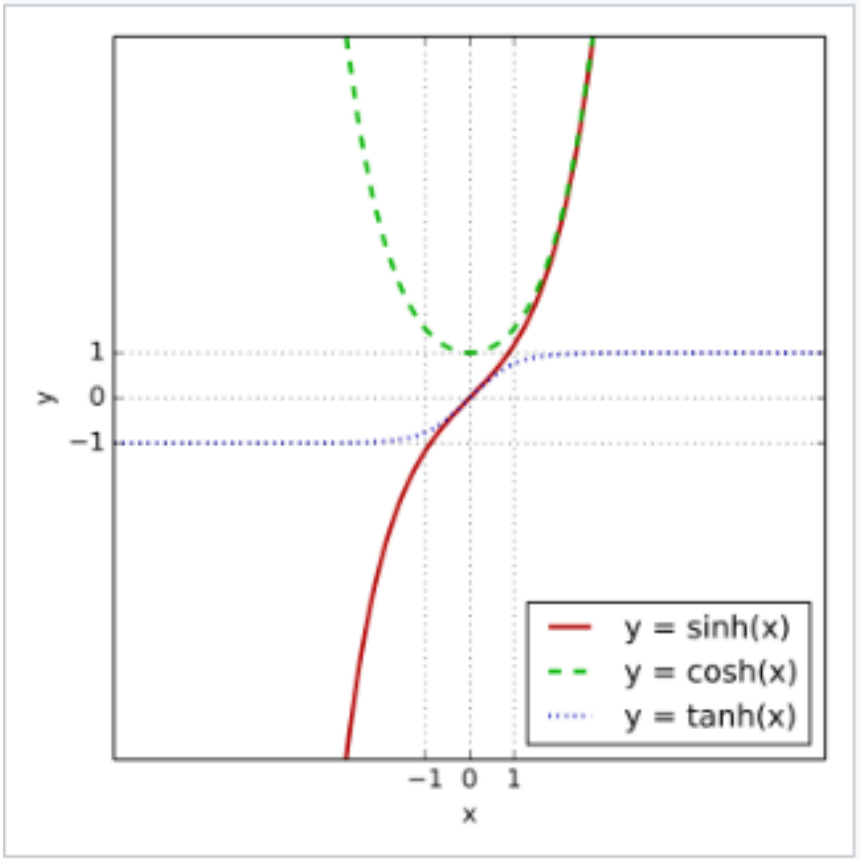
- 쌍곡코사인(hyperbolic cosine)

$$\cosh x = \frac{e^x + e^{-x}}{2} = \cos ix$$

- 쌍곡탄젠트(hyperbolic tangent)

$$\tanh x = \frac{\sinh x}{\cosh x}$$

$$= \frac{\frac{e^x - e^{-x}}{2}}{\frac{e^x + e^{-x}}{2}} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1} = -i \tan ix$$



sinh, cosh, tanh

“math” Module : Hyperbolic Functions [2/2]

- math.**cosh**(x) : Return the hyperbolic cosine of x
- math.**sinh**(x) : Return the hyperbolic sine of x
- math.**tanh**(x) : Return the hyperbolic tangent of x

```
print("cosh of pi : ", math.sinh( math.pi ))
print("sinh of pi : ", math.cosh( math.pi ))
print("tanh of pi : ", math.tanh( math.pi ))
```

Result

```
cosh of pi :  11.5487393572577
sinh of pi :  11.5919532755215
tanh of pi :  0.99627207622075
```

(Ch 28) Math, Statistics, Random Modules

- **math module**
- **statistics module**
- **random module**

"statistics" Module - Functions [1/3]

- statistics.**mean( )**: return the sample arithmetic mean of data, a sequence or iterator of real-valued numbers
- statistics.**median( )**: return the median of numeric data, using the common "mean of middle two" method
  - median_high( ), median_low( ), median_grouped( )
- statistics.**mode( )**: return the most common data point from discrete or nominal data. The mode is the most typical value, and is a robust measure of central location

```
from statistics import *

print(mean([1, 2, 3, 4, 4]))
print(mean([-1.0, 2.5, 3.25, 5.75]))
print(median([1, 3, 5]))
print(median([1, 3, 5, 7]))
print(median_low([1, 3, 5, 7]))
print(median_high([1, 3, 5, 7]))
print(median_grouped([52, 52, 53, 54]))
```

```
2.8
2.625
3
4.0
3
5
52.5
```

```
print(mode([1, 1, 2, 3, 3, 3, 3, 4]))
print(mode(["red", "blue", "blue", "red", "green", "red", "red"]))
```

```
3
red
```

참고사항

Population Data vs Sample Data

모평균, 모표준편차, 모분산

The **population standard deviation** formula is:

$$\sigma = \sqrt{\frac{\sum(X-\mu)^2}{n}}$$

where,

σ = population standard deviation
$\sum$ = sum of...
μ = population mean
n = number of scores in sample.

표본평균, 표본표준편차, 표본분산

The **sample standard deviation formula** is:

$$s = \sqrt{\frac{\sum(X-\bar{X})^2}{n-1}}$$

where,

s = sample standard deviation
$\sum$ = sum of...
$\bar{X}$ = sample mean
n = number of scores in sample.

Population Data 예: 동아리 학생 10명의 키

참고사항

	수현	수지	효주	세영	진구	호준	서진	지우	재석	준하
키 (cm)	177	167	160	162	174	180	176	158	172	184

X를 회원의 키라고 하면 X의 평균 $\mu = \mathrm{E}(X)$는

$$\mu = \frac{1}{10}(177 + 167 + 160 + 162 + 174 + 180 + 176 + 158 + 172 + 184) = 171$$

이다. $(X - \mu)$와 $(X - \mu)^2$를 구하면 다음 표와 같다.

표준편차

	수현	수지	효주	세영	진구	호준	서진	지우	재석	준
$X - \mu$	6	-4	-11	-9	3	9	5	-13	1	1
$(X - \mu)^2$	36	16	121	81	9	81	25	169	1	1(

따라서 분산은

$$\mathrm{V}(X) = \mathrm{E}((X - \mu)^2) = \frac{708}{10} = 70.8$$

이고 표준편차는

$$\sigma_X = \sqrt{\mathrm{V}(X)} = \sqrt{70.8} \approx 8.4$$

μ = population mean

σ = population standard deviation

$$\sigma = \sqrt{\frac{\sum(X - \mu)^2}{n}}$$

Sample Data의 예

참고사항

- 전교학생이 200명이다
- 그중에 10명을 sampling해서 키를 재었다
- { 155, 167, 170, 175, 180, 159, 160, 177, 182, 165}

- 이 Sample의 Standard Deviation을 구하라

$\bar{X}$ = sample mean

s = sample standard deviation

$$s = \sqrt{\frac{\sum(X - \bar{X})^2}{n - 1}}$$

“statistics” Module - Functions [2/3]

- statistics.**pstdev( )**: return the population standard deviation
- statistics.**pvariance( )**: return the population variance of data, a non-empty iterable of real-valued numbers

Assume Population Data

```
print(pstdev([1.5, 2.5, 2.5, 2.75, 3.25, 4.75]))

data = [0.0, 0.25, 0.25, 1.25, 1.5, 1.75, 2.75, 3.25]
print(pvariance(data))

mu = mean(data)
print(pvariance(data, mu))
```

```
0.9868932735272 51
1.25
1.25
```

“statistics” Module - Functions [3/3]

- statistics.**stdev()**: return the sample standard deviation
- statistics.**variance()**: return the sample variance of data, an iterable of at least two real-valued numbers

Assume Sample Data

```
from statistics import *

print(stdev([1.5, 2.5, 2.5, 2.75, 3.25, 4.75]))

data = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]
print(variance(data))

mu = mean(data)
print(variance(data, mu))
```

```
1.0810874155219827
1.1760204081632655
1.3720238095238095
```

Statistics in Various Python Places [1/6]

Python Standard Library: Statistics

- Averages and measures of central location

These functions calculate an average or typical value from a population or sample.

`mean()`	Arithmetic mean (“average”) of data.
`harmonic_mean()`	Harmonic mean of data.
`median()`	Median (middle value) of data.
`median_low()`	Low median of data.
`median_high()`	High median of data.
`median_grouped()`	Median, or 50th percentile, of grouped data.
`mode()`	Mode (most common value) of discrete data.

- Measures of spread

These functions calculate a measure of how much the population or sample tends to deviate from the typical or average values.

`pstdev()`	Population standard deviation of data.
`pvariance()`	Population variance of data.
`stdev()`	Sample standard deviation of data.
`variance()`	Sample variance of data.

Statistics in Various Python Places [2/6]

Statistics Functions in Numpy (A)

Order statistics¶

`amin` (a[, axis, out, keepdims])	Return the minimum of an array or minimum along an axis.
`amax` (a[, axis, out, keepdims])	Return the maximum of an array or maximum along an axis.
`nanmin` (a[, axis, out, keepdims])	Return minimum of an array or minimum along an axis, ignoring any NaNs.
`nanmax` (a[, axis, out, keepdims])	Return the maximum of an array or maximum along an axis, ignoring any NaNs.
`ptp` (a[, axis, out])	Range of values (maximum - minimum) along an axis.
`percentile` (a, q[, axis, out, ...])	Compute the qth percentile of the data along the specified axis.
`nanpercentile` (a, q[, axis, out, ...])	Compute the qth percentile of the data along the specified axis, while ignoring nan values.

Statistics in Various Python Places [3/6]

Statistics Functions in Numpy (B)

Averages and variances

`median`(a[, axis, out, overwrite_input, keepdims])	Compute the median along the specified axis.
`average`(a[, axis, weights, returned])	Compute the weighted average along the specified axis.
`mean`(a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis.
`std`(a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis.
`var`(a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis.
`nanmedian`(a[, axis, out, overwrite_input, ...])	Compute the median along the specified axis, while ignoring NaNs.
`nanmean`(a[, axis, dtype, out, keepdims])	Compute the arithmetic mean along the specified axis, ignoring NaNs.
`nanstd`(a[, axis, dtype, out, ddof, keepdims])	Compute the standard deviation along the specified axis, while ignoring NaNs.
`nanvar`(a[, axis, dtype, out, ddof, keepdims])	Compute the variance along the specified axis, while ignoring NaNs.

Statistics in Various Python Places [4/6]

Statistics Functions in Numpy (C)

Correlating

`corrcoef` (x[, y, rowvar, bias, ddof])	Return Pearson product-moment correlation coefficients.
`correlate` (a, v[, mode])	Cross-correlation of two 1-dimensional sequences.
`cov` (m[, y, rowvar, bias, ddof, fweights, ...])	Estimate a covariance matrix, given data and weights.

Histograms

`histogram` (a[, bins, range, normed, weights, ...])	Compute the histogram of a set of data.
`histogram2d` (x, y[, bins, range, normed, weights])	Compute the bi-dimensional histogram of two data samples.
`histogramdd` (sample[, bins, range, normed, ...])	Compute the multidimensional histogram of some data.
`bincount` (x[, weights, minlength])	Count number of occurrences of each value in array of non-negative ints.
`digitize` (x, bins[, right])	Return the indices of the bins to which each value in input array belongs.

Statistics in Various Python Places [5/6]

Statistics in Pandas

Function	Description
count	Number of non-NA observations
sum	Sum of values
mean	Mean of values
mad	Mean absolute deviation
median	Arithmetic median of values
min	Minimum
max	Maximum
mode	Mode
abs	Absolute Value
prod	Product of values
std	Bessel-corrected sample standard deviation
var	Unbiased variance
sem	Standard error of the mean
skew	Sample skewness (3rd moment)
kurt	Sample kurtosis (4th moment)
quantile	Sample quantile (value at %)
cumsum	Cumulative sum
cumprod	Cumulative product
cummax	Cumulative maximum
cummin	Cumulative minimum

Statistics in Various Python Places [6/6]

Statistics in Scipy

scipy.stats submodule

- supports various distribution objects
- contains various statistical hypothesis test functions

- Statistical functions (`scipy.stats`)
  - Continuous distributions
  - Multivariate distributions
  - Discrete distributions
  - Statistical functions
  - Circular statistical functions
  - Contingency table functions
  - Plot-tests
  - Masked statistics functions
  - Univariate and multivariate kernel density estimation (`scipy.stats.kde`)

(Ch 28) Math, Statistics, Random Modules

- **math module**
- **statistics module**
- **random module**

“random” Module

- Random module implements pseudo-random number generators
- Python uses the Mersenne Twister (**메르센 트위스터**) as the core generator
  - It produces 53-bit precision floats and has a period of $2^{19937} - 1$
  - 1997년에 마츠모토 마코토 와 니시무라 다쿠지가 개발한 유사난수 생성기
- Even though it is extensive, it is also completely deterministic
  - Therefore we can somehow predict the values

```
import random

random.seed(2)
a = random.random()

print(a)
```

Result

```
>>>
0.9560342718892494
```

“random” Module – Functions [1/3]

- random.**random**( ) : Return the next random floating point number in the range [0.0, 1.0) which is greater or equal to 0.0, less than 1.0
- random.**seed**(a): Initialize the random number generator by a
- random.**seed**( ) : Initialize the random number generator with current system time

```
import random

random.seed(3)
a = random.random()
random.seed(3)
b = random.random()

print("seed(3)")
print("a : ", a)
print("b : ", b)

random.seed()
c = random.random()
random.seed()
d = random.random()

print("seed()")
print("c : ", c)
print("d : ", d)
```

Result

```
>>>
seed(3)
a :  0.2379646270918913 7
b :  0.2379646270918913 7
seed()
c :  0.1006005554769392 5
d :  0.3473399591181498 5
```

“random” Module – Functions [2/3]

- random.**uniform**(a, b) : Return a random floating point number in [a, b]
- random.**randint**(a, b) : Return a random integer N in [a,b]
- random.**randrange**(a) : Return a random integer N in [0, a)
- random.**randrange**(a, b, c) : Return a random integer N in [a, b) where N = a + kc (k is constant)

```
import random

a = random.uniform(3, 10)
b = random.randint(3, 10)
c = random.randrange(10)
d = random.randrange(0, 15, 5)

print("uniform(3,10) : ", a)
print("randint(3,10) : ", b)
print("randrange(10) : ", c)
print("randrange(0,15,5) : ", d)
```

Result

```
>>>
uniform(3,10) :  8.085661898057056
randint(3,10) :  5
randrange(10) :  8
randrange(0,15,5) :  10
```

“random” Module – Functions [3/3]

- random.**choice**(temp) : Return a random element from any sequence ‘temp’
- random.**shuffle**(a) : Shuffle the sequence a. The original data a is modified.
- random.**sample**(a, b) : Return b length list of randomly chosen elements from sequence a

```
import random

a = 'abcdefghij'
b = [1, 2, 3, 4, 5, 6, 7]
random.shuffle(b)
c = [1, 2, 3, 4, 5]

print("choice(a) : ", random.choice(a))
print("shuffle(b) : ", b)
print("sample(c, 3) : ", random.sample(c, 3))
```

Result

```
>>>
choice(a) :  c
shuffle(b) :  [2, 3, 4, 6, 5, 1, 7]
sample(c, 3) :  [4, 5, 2]
```

Example code with “random” Module

```
# random_pop.py
import random
def random_pop(data):
    number = random.randint(0, len(data)-1)
    return data.pop(number)

if __name__ == "__main__":
    data = [1, 2, 3, 4, 5]
    while data: print(random_pop(data))
```

||

```
def random_pop(data):
    number = random.choice(data)
    data.remove(number)
    return number
```

random_pop()은 리스트의 요소중에서 무작위로 하나를 선택하여 그값을 리턴한다.

꺼내진 item은 pop( ), remove( )에 의해서 list에서사라진다

list.pop(number) 는 list에서 number를 제거하고 number를 return

list.remove(number) 는 list에서 number만 제거

Major Functions in "random" Module

중요

- randint(n1, n2) ➔ n1 ~ n2사이의 random integer
- randrange(n1, n2) ➔ n1 ~ n2사이의 random integer
  - n2는 포함이 안됨, randrange(n1, n2, step)

- random( ) ➔ 0 ~ 1사이의 random float number
- uniform(n1, n2) ➔ n1 ~ n2사이의 random float number

- choice(L) ➔ L중에서 1개 선택, no modification on L
- sample(L, n) ➔ L중에서 n개 선택, no modification on L

- shuffle(L) ➔ L을 shuffling, L is modified