-Type Conversion

# Definition of savings and result

savings = 100

result = 100 \* 1.10 \*\* 7

# Fix the printout

print("I started with $" + str(savings) + " and now have $" + str(result) + ". Awesome!")

# Definition of pi\_string

pi\_string = "3.1415926"

# Convert pi\_string into float: pi\_float

pi\_float = float(pi\_string)

pi\_float

\*str : mengubah type data menjadi string

\*float : mengubah type data menjadi float

-Create a List

# area variables (in square meters)

hall = 11.25

kit = 18.0

liv = 20.0

bed = 10.75

bath = 9.50

# Create list areas

areas = [hall,kit, liv, bed, bath]

# Print areas

print(areas)

-Create a List with Different Types

# area variables (in square meters)

hall = 11.25

kit = 18.0

liv = 20.0

bed = 10.75

bath = 9.50

# Adapt list areas

areas = ["hallway", hall, "kitchen", kit, "living room", liv, "bedroom", bed, "bathroom", bath]

# Print areas

print(areas)

-List of List

# area variables (in square meters)

hall = 11.25

kit = 18.0

liv = 20.0

bed = 10.75

bath = 9.50

# house information as list of lists

house = [["hallway", hall],

         ["kitchen", kit],

         ["living room", liv],

         ["bedroom", bed],

         ["bathroom", bath]]

# Print out house

print(house)

# Print out the type of house

print(type(house))

-Slicing List

\*fam[:4] : Slicing from first index until 3rd index

\*fam[5:] : Slicing 5th element until the end of element

-Subset and Conquer

# Create the areas list

areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]

# Print out second element from areas

print(areas[1])

# Print out last element from areas

print(areas[-1])

# Print out the area of the living room

print(areas[5])

-Subset and Calculate :

# Create the areas list

areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]

# Sum of kitchen and bedroom area: eat\_sleep\_area

eat\_sleep\_area = areas[3]+areas[7]

# Print the variable eat\_sleep\_area

print(eat\_sleep\_area)

-Slicing and Dicing :

# Create the areas list

areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]

# Use slicing to create downstairs

downstairs = areas[0:6]

# Use slicing to create upstairs

upstairs = areas[6:10]

# Print out downstairs and upstairs

print(downstairs)

print(upstairs)

-Slicing and Dicing (2) :

# Create the areas list

areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]

# Alternative slicing to create downstairs

downstairs = areas[:6]

# Alternative slicing to create upstairs

upstairs = areas[6:]

In [1]:

x = [["a", "b", "c"],  
     ["d", "e", "f"],  
     ["g", "h", "i"]]  
  
In [2]:

x[2][0]

Out[2]:

'g'

In [3]:

x[2][:2]

Out[3]:

['g', 'h']

**-Manipulating List**

-changing list element :

**\*fam[7] = 1.86** -> change indeks 7th to 1.86

**\*fam[0:2] = [“Lisa”, 1.74]** -> Change indeks 0 and 1

-Adding new element :

**\* fam\_add = fam + [“me”, 1.79]**

-Deleting List :

**\*del(fam[2])** : Deleting indeks ke 2

-giving the equals sign to the list :

**\*x = [“a”, “b”, “c”]**

**\* y = x**

**\*y[1] = “z”**

**\*y**

**\*[“a”, “z”, “c”]**

**\* x akan menghasilkan list yg sama**

-Replace List Element :

# Create the areas list

areas = ["hallway", 11.25, "kitchen", 18.0, "living room", 20.0, "bedroom", 10.75, "bathroom", 9.50]

# Correct the bathroom area

areas[-1]=10.50

# Change "living room" to "chill zone"

areas[4] = "chill zone"

-Extend a List :

# Create the areas list and make some changes

areas = ["hallway", 11.25, "kitchen", 18.0, "chill zone", 20.0,

         "bedroom", 10.75, "bathroom", 10.50]

# Add poolhouse data to areas, new list is areas\_1

areas\_1 = areas + ["poolhouse", 24.5]

# Add garage data to areas\_1, new list is areas\_2

areas\_2 = areas\_1 + ["garage", 15.45]

-Inner Working of List :

# Create list areas

areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Create areas\_copy

areas\_copy = list(areas)

# Change areas\_copy

areas\_copy[0] = 5.0

# Print areas

print(areas)

**#FUNCTIONS AND PACKAGES**

**\*max()** : untuk mencari nilai tertinggi

**\*round(1.68, 1)** : result = 1.7

\*round(1.68) : result = 2

-familiar functions

# Create variables var1 and var2

var1 = [1, 2, 3, 4]

var2 = True

# Print out type of var1

print(type(var1))

# Print out length of var1

print(len(var1))

# Convert var2 to an integer: out2

out2 = int(var2)

-Multiple Arguments :

# Create lists first and second

first = [11.25, 18.0, 20.0]

second = [10.75, 9.50]

# Paste together first and second: full

full = first + second

full

# Sort full in descending order: full\_sorted

full\_sorted = sorted(full, reverse = True) -> reverse = True, maka descending

# Print out full\_sorted

print(full\_sorted)

-Methods : function to belongs with object/variable

-contoh method : capitalize(), replace(), bit\_length(), conjugate(), index(), count()

-penggunaan : object.methods()

**-fam.index(“mom”)** :untuk melihat “mom” ada di indeks ke brp

**-fam.count(1.73)** = 1 -> menghitung variable 1.73 ada brp

**-sister.capitalize()** : untuk merubah huruf pertama menjadi capital

**-sister.replace(“z”, “sa”)** : merubah “z” menjadi “sa”

-setiap type data punya methods yg berbeda-beda. Tidak bisa dipaksakan semua

-string methods :

# string to experiment with: place

place = "poolhouse"

# Use upper() on place: place\_up

place\_up = place.upper()

# Print out place and place\_up

print(place)

print(place\_up)

# Print out the number of o's in place

print(place.count("o"))

-List methods :

# Create list areas

areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Print out the index of the element 20.0

print(areas.index(20.0))

# Print out how often 9.50 appears in areas

print(areas.count(9.50))

-list methods : 1.remove() to remove first element of a list that matches the input. 2.reverse() to reverse the order of the elements in the list (bukan mengurutkan)

-List Methods(2) :

# Create list areas

areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Use append twice to add poolhouse and garage size

areas.append(24.5)

areas.append(15.45)

# Print out areas

print(areas)

# Reverse the orders of the elements in areas

areas.reverse()

# Print out areas

print(areas)

-Install Package :

\*pip.readthedocs.org/en/stable/installing/

\*download get-pip.py

\*execute in cmd : -python3 get-pip.py, -pip3 install numpy

-Import Package :

# Definition of radius

r = 0.43

# Import the math package

import math

# Calculate C

C = 2\*math.pi\*r

# Calculate A

A = math.pi\*r\*r

# Build printout

print("Circumference: " + str(C))

print("Area: " + str(A))

-Selective Import

# Definition of radius

r = 192500

# Import radians function of math package

from math import radians -> selective import. Just import radians

# Travel distance of Moon over 12 degrees. Store in dist.

dist = r \* radians(12)

# Print out dist

print(dist)

**#NUMPY**

-in python, list can not be included in arithmetical

-solution is using numpy

-contoh :

**\*import numpy as np**

**\*np\_height = np.array(height)**

**Result : array([list])**

**\*bmi = np\_weight/np\_height\*\*2**

- Numpy array hanya berisi 1 tipe. Jika ada banyak tipe dalam 1 array, maka akan di convert menjadi string.

- Numpy Subsetting :

\*bmi[1] : result = 21

\*bmi[bmi >23] : result = nilai yg lebih dari 23 dalam bentuk array

-First Numpy Array :

# Create list baseball

baseball = [180, 215, 210, 210, 188, 176, 209, 200]

# Import the numpy package as np

import numpy as np

# Create a numpy array from baseball: np\_baseball

np\_baseball = np.array(baseball)

# Print out type of np\_baseball

print(type(np\_baseball))

-Baseball players height :

# height is available as a regular list

# Import numpy

import numpy as np

# Create a numpy array from height\_in: np\_height\_in

np\_height\_in = np.array(height\_in)

# Print out np\_height\_in

print(np\_height\_in)

# Convert np\_height\_in to m: np\_height\_m

np\_height\_m = np\_height\_in \* 0.0254

# Print np\_height\_m

print(np\_height\_m)

-Baseball player BMI

# height and weight are available as regular lists

# Import numpy

import numpy as np

# Create array from height\_in with metric units: np\_height\_m

np\_height\_m = np.array(height\_in) \* 0.0254

# Create array from weight\_lb with metric units: np\_weight\_kg

np\_weight\_kg = np.array(weight\_lb)\*0.453592

# Calculate the BMI: bmi

bmi = np\_weight\_kg/np\_height\_m\*\*2

# Print out bmi

print(bmi)

-Lightweight Baseball player

# height and weight are available as a regular lists

# Import numpy

import numpy as np

# Calculate the BMI: bmi

np\_height\_m = np.array(height\_in) \* 0.0254

np\_weight\_kg = np.array(weight\_lb) \* 0.453592

bmi = np\_weight\_kg / np\_height\_m \*\* 2

# Create the light array

light = bmi <21

# Print out light

print(light)

# Print out BMIs of all baseball players whose BMI is below 21

print(bmi[light])

-subsetting numpy array

# height and weight are available as a regular lists

# Import numpy

import numpy as np

# Store weight and height lists as numpy arrays

np\_weight\_lb = np.array(weight\_lb)

np\_height\_in = np.array(height\_in)

# Print out the weight at index 50

print(np\_weight\_lb[50])

# Print out sub-array of np\_height\_in: index 100 up to and including index 110

print(np\_height\_in[100:111])

**#2D NUMPY ARRAY**

-2d numpy :

**\*np\_2d = np.array([[1, 2, 3, 4, 5],**

**[6, 7, 8, 9, 10]])**

**\*np\_2d.shape()** : result = 2 rows, 5 column

-Subsetting numpy 2d :

**\*np\_2d[0]** -> mengambil semua baris pertama

**\*np\_2d[0][3] or np\_2d[0,3]** -> mengambil baris pertama kolom ke-4

**\*np\_2d[:, 1:3]** -> mengambil semua baris pada kolom no.2 dan 3

-2D numpy array :

# Create baseball, a list of lists

baseball = [[180, 78.4],

            [215, 102.7],

            [210, 98.5],

            [188, 75.2]]

# Import numpy

import numpy as np

# Create a 2D numpy array from baseball: np\_baseball

np\_baseball = np.array(baseball)

# Print out the type of np\_baseball

print(type(np\_baseball))

# Print out the shape of np\_baseball

print(np\_baseball.shape) -> result (4,2)

-Baseball data in 2D form :

# baseball is available as a regular list of lists

# Import numpy package

import numpy as np

# Create a 2D numpy array from baseball: np\_baseball

np\_baseball = np.array(baseball)

# Print out the shape of np\_baseball

print(np\_baseball.shape)

-Subsetting 2D numpy array :

# baseball is available as a regular list of lists

# Import numpy package

import numpy as np

# Create np\_baseball (2 cols)

np\_baseball = np.array(baseball)

# Print out the 50th row of np\_baseball

print(np\_baseball[49])

# Select the entire second column of np\_baseball: np\_weight\_lb

np\_weight\_lb = np\_baseball[:,1]

# Print out height of 124th player

print(np\_baseball[123,0])

-2D Arithmatic :

# baseball is available as a regular list of lists

# updated is available as 2D numpy array

# Import numpy package

import numpy as np

# Create np\_baseball (3 cols)

np\_baseball = np.array(baseball)

# Print out addition of np\_baseball and updated

print(np\_baseball + updated)

# Create numpy array: conversion

conversion = np.array([0.0254, 0.453592, 1])

# Print out product of np\_baseball and conversion

print(np\_baseball \* conversion)

-Numpy for statistic:

-Mean : **np.mean(np\_city[:,0])**

-Median : **np.median(np\_city[:,0])**

-Correlation : **np.corrcoef(np\_city[:,0], np\_city[:,1])**

-standar deviasi : **np.std(np\_city[:,0])**

-Generate Data from numpy :

**\*np.random.normal(distribution mean, distribution standard deviation, number of samples)**

**\*np.round(np.random.normal(1.75, 0.2, 5000),2)**

-Average vs median :

# np\_baseball is available

# Import numpy

import numpy as np

# Create np\_height\_in from np\_baseball

np\_height\_in = np\_baseball[:,0]

# Print out the mean of np\_height\_in

print(np.mean(np\_height\_in))

# Print out the median of np\_height\_in

print(np.median(np\_height\_in))

-Explore baseball data

# np\_baseball is available

# Import numpy

import numpy as np

# Print mean height (first column)

avg = np.mean(np\_baseball[:,0])

print("Average: " + str(avg))

# Print median height. Replace 'None'

med = np.median(np\_baseball[:,0])

print("Median: " + str(med))

# Print out the standard deviation on height. Replace 'None'

stddev = np.std(np\_baseball[:,0])

print("Standard Deviation: " + str(stddev))

# Print out correlation between first and second column. Replace 'None'

corr = np.corrcoef(np\_baseball[:,0],np\_baseball[:,1])

print("Correlation: " + str(corr))

-Blend it all together

# heights and positions are available as lists

# Import numpy

import numpy as np

# Convert positions and heights to numpy arrays: np\_positions, np\_heights

np\_positions = np.array(positions)

np\_heights = np.array(heights)

# Heights of the goalkeepers: gk\_heights

gk\_heights = np\_heights[np\_positions=='GK']

# Heights of the other players: other\_heights

other\_heights = np\_heights[np\_positions != 'GK']

# Print out the median height of goalkeepers. Replace 'None'

print("Median height of goalkeepers: " + str(np.median(gk\_heights)))

# Print out the median height of other players. Replace 'None'

print("Median height of other players: " + str(np.median(other\_heights)))