```
---------------------------------------list--------------------------------
------

courses = ['math', 'physics', 'computersci', 'history']
courses_2 = ['geometry', 'phychology']
nums = [4, 6, 2, 3, 1, 12]

print(courses[2])
print(courses[2:])

courses.append('drawing')
print('new list is', courses)

courses.insert(0, 'geography')
print('new list with defining indexes', courses)

courses.extend(courses_2)  # courses.append(courses_2)  for appended list
directly

print('appended list for individual items', courses)

courses.remove('history')
print('list after removing values', courses)

popped = courses.pop()  # to remove the last values
print('popped values', popped)
print('after removing the last values', courses)

courses.reverse()
print('list after reverse list', courses)

courses.sort()
print('list after sorting', courses)

courses.sort(reverse=True)
print('list after reverse sorting', courses)

sorted_courses = sorted(courses)
print('sorted version of the list', sorted_courses)

print('minimum number of the list', min(nums))
print('maximum number of the list', max(nums))
print('summation of the numbers of the list', sum(nums))
print('Art' in courses)

for item in courses:
    print(item)
```

```
for index,course in enumerate(courses,start=1):
    print(index,course)

course_str=' - '.join(courses)
print(course_str)

new_str = course_str.split(' - ')
print(new_str)


------------------------------------named
tuple-------------------------------------

from collections import namedtuple


def merge(*records):
    """
    :param records: (varargs list of namedtuple) The patient details.
    :returns: (namedtuple) named Patient, containing details from all
records, in entry order.
    """


PersonalDetails = namedtuple('PersonalDetails', ['date_of_birth'])
personal_details = PersonalDetails('06-04-1972')

Complexion = namedtuple('Complexion', ['eye_color', 'hair_color'])
complexion = Complexion('Blue','Black')



print('date_of_birth=',personal_details.date_of_birth,'eye_color=',complex
ion.eye_color,'hair_color=',complexion.hair_color)


print(merge(personal_details.date_of_birth, complexion.eye_color)) #
returns null

-----------------------tuple-----------------------------------------

coordinates=(4,5)
sub1 = ('math','history','geography')
sub2 = sub1

print(coordinates[0])
print(sub2)



-------------------------------dictionary------------------------------------

student = {'name':'john', 'address':'helsinki','sub':['Math','Physics']}

print(student['sub'])
print(student['name'])

# get always returns something instead of error

print(student.get('address'))
print(student.get('phone'))

# If don't found returns not found
print(student.get('phone', 'not found'))

student['phone'] = '4444-2222'
student['name'] = 'bob'   # dictionary will be updated

print(student)

student.update({'name':'janne','age':'35'})
print(student)

del student['age']  # or age = student.pop('age')
print(student)

keys = student.keys()
print(keys)
values = student.values()
print(values)
items = student.items()
print(items)

for key,value in student.items():
    print(key,value)



--------------------Sets----------------------------

cs_subject = {'bengali','english','literature','math'}

# remove duplicates
```

```
print(cs_subject)
print('english' in cs_subject)

art_subject = {'art','english','literature','design'}


print(cs_subject.intersection(art_subject))
print(cs_subject.difference(art_subject))
print(cs_subject.union(art_subject))


-------------empty all-----------------------------

empty_list=[]
# or
empty_list=list()


empty_tuple=()
# or
empty_tuple=tuple()

empty_set = {} # this is incorrect , this will create a dictionary
empty_set = set()
```