```python
x = range(1, 21, 2)
y = ["hello", 3.14, 2020]

# convert x to a list
list(x)

# check if an object is a list
type(y) == list

# add element to a list
y.append("new")

# if appending a list to a list, it will be appended as a single elemen
# if you want to add all elements of a list to a new list, use extend
c = [1, 2, 3]
d1, d2 = c[:], c[:]
d1.append(["new", "year", 2020])
d2.extend(["new", "year", 2020])
print(d1)
print(d2)

# indexing, starts at 0
print(y[0])
print(y[0][4])

# indexing from the end, starts at -1
# y is now ['hello', 3.14, 2020, 'new']
print(y[-1])
print(y[-4][-1])

# modify list elements
y[0] = "hello there!"

# slicing, [including : excluding]
print(y[0:2])

# reverse elements of a list, in-place vs creating a new list
print(y)
print(y[::-1]) # created a new list with reverse order
print(y) # the original list is not changed
```

```
print(y.reverse()) # prints None
print(y) # the original list is reversed; in-place reversal

# in and not in operators
print(3.14 in y)
print(3.14 not in y)

# list concatenations
z = ["ok", 999] + y
z += [17] # can't do z += 17!

# length of a list is the number of elements
print(f"length of z is {len(z)}")

# nested lists: for example, element of a list is another list
nested = ["zero index", y, z]
print(nested)
print(f"length of nested is {len(nested)}")
print(nested[2][2][6:11])

# delete an element of a list using its index
# you can also using slicing to delete multiple elements
del nested[2]
print(nested)

# alternative approach to delete an element using its index
nested[0]  = [] # you can also use slicing
print(nested)

# insert an element (or elements) at a specific location
print(y)
y[2:2] = ["current year", 2019]
print(y)

# alternative approach to insert an element
y.insert(2, "this is new again")
print(y)

# remove an element using its VALUE (removes first element only!)
y.remove(2019)
```

```
print(y)

x = [1, 2, 4, 1, 6, 1, 1, 8, 1]
x.remove(1)
print(x)

# use 'while' and 'in' to remove all the elements equal to a give value
while 1 in x: x.remove(1)
print(x)

# pop() function, you can use it to remove a single element
print(y.pop(0))
print(y)
print(y.pop())
print(y)

# clear a list
x = [1, 2, 4, 1, 6, 1, 1, 8, 1]
x.clear() # returns empty list
# you can also use:
# del x[:]
# x[:] = []
# do NOT use x = [], which will be explained below

# get an index of an element from the list using its value
print(f"{y[0]} in {y} has an index of", end = " ")
print(y.index(3.14))
# if the element is not in the list you will get an error!

# list assignment vs .copy()
y2 = y
y3 = y.copy() # the values are the same but are they the same object?
print(y)
print(y2)
print(y3)

y[2] = "last year" # only affected y and y2, but y3
print(y)
print(y2)
print(y3)
```

```
y = [] # creates a new empty list and assigns to y; does not clear y an
print(y)
print(y2)
print(y3)

# for 201
print()
print("simple copy of a list")
g = [[1, 2], [3, 4]]
h = g.copy()
g[1][1] = 7
print(g)
print(h)

print()
print("deep copy of a list; relies on copy library")
g = [[1, 2], [3, 4]]
import copy
h = copy.deepcopy(g)
g[1][1] = 7
print(g)
print(h)

# below is an example of a simple copy but not a deep copy!
print()
g = [1, 2, 3, 4]
k = g[:]
g[1] = 7
print(g)
print(k)

print()
g = [[1, 2], [3, 4]]
k = g[:]
g[1][1] = 7
print(g)
print(k)
print()
```

```
# if all elements of the list are of the same type you can use sort()
x = [1, 6, 2, 8, 4, 9, 11]
x.sort()
print(x)
x.sort(reverse = True)
print(x)

x = ["March", "April", "May"]
x.sort()
print(x)

x = [1, 2, 4, 1, 6, 1, 1, 8, 1]
x2 = x.copy()
num = 1
locs = []
while num in x2:
    locs.append(x2.index(num))
    x2[x2.index(num)] = "dummy"
print("--------")
print(x)
print(x2)
print(locs)

# count number of times a value appears in a list
print(x.count(1)) # count 1's in x
```