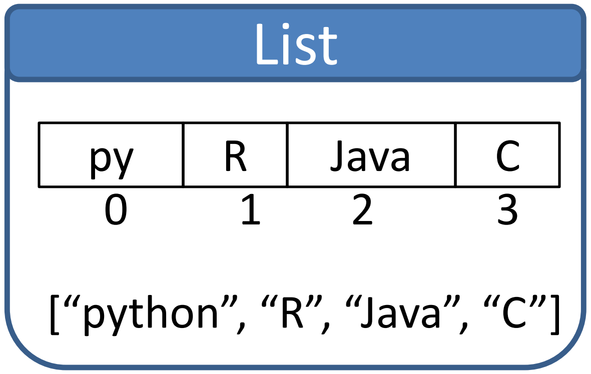
Types in Python

- Python is not a typed language
  - No type of variables declared
  - No fixed memory size is allocated for a variable declared
- What does it mean to you to program?

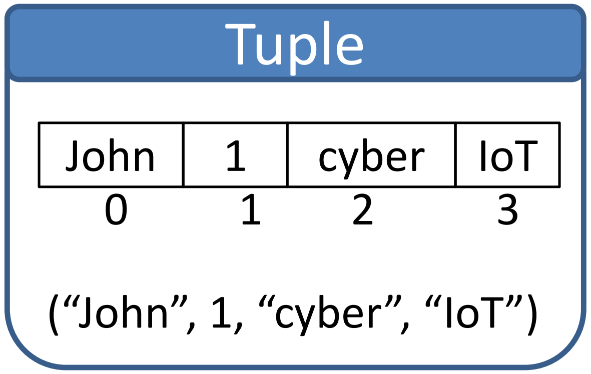
Motivation

- Consider a program that can compute the average of exams for each student. There are 100 students.
- How do you do it?
  - Create a variable name1 for the first student, name2 for the second student, etc
  - Create a variable mid1 for the name1's exam, mid2 for the name2's, etc.
  - Do this for final exam, and also the average for each student for 100 times, …
  - So many variables ;(;(;(
- Any other way to make it efficiently?
  - Think…

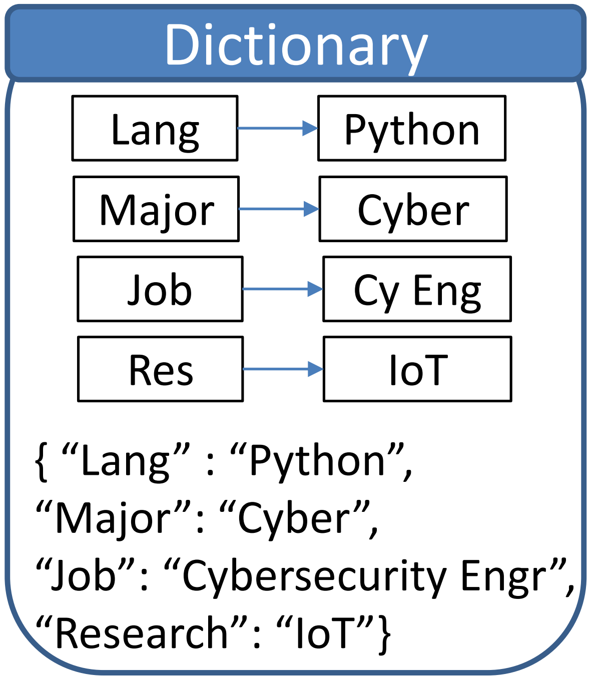
Data Types



["python", "R", "Java", "C"]



("John", 1, "cyber", "IoT")



{ "Lang" : "Python",
"Major": "Cyber",
"Job": "Cybersecurity Engr",
"Research": "IoT"}

- Observe the example very carefully
  - What can be in which data type

- Points of Consideration
  - How to define
  - How to access (use)

Examples of Data Types

- However, basic DT’s are available:
  - List [‘a’, 1, 3, ’mercy’]
  - Tuple (‘John’, ‘faculty’, 555)
  - Dictionary {“USA”: “Washington DC”, “Canada”: “Ottawa”, “Korea”: “Seoul”, “UK”: “London”}
- How are they different?
  - Not just symbols, [ ], ( ), { }, but compare those data elements in it.
- In what case, which DT needs to be applied?

Understanding Data Types

In what case, which DT needs to be applied?

- Camera lenses: Canon lens, Nikon lens, Olympus lens, Sony lens, can be represented in _____________ data type.
- Camera lenses have features: shutter speed, ISO sensor, aperture, focus. These features can be represented in ____________ data type.
- There are 5 photos, p1, p2, p3, p4 and p5 which are taken by Canon lens, Nikon lens, Nikon lens, Sony lens, Sony lens, respectively. The data type _____ is appropriate to represent these photos.

Data Types

The data types, list, tuple or dictionary, are the structure of data collections.

- List: contains values of the same type
- Tuple: contains values that can constitute an object
- Dictionary: contains key and value pairs

List

- Try and feel the structure of lists

```
In [83]: lst = ['a', 1, 33, 'Mercy']

In [84]: lst[0]
Out[84]: 'a'

In [85]: lst[0:2]
Out[85]: ['a', 1]

In [86]: lst[-1]
Out[86]: 'Mercy'

In [87]: lst[-1:-2]
Out[87]: []

In [88]: lst[-2:-1]
Out[88]: [33]
```

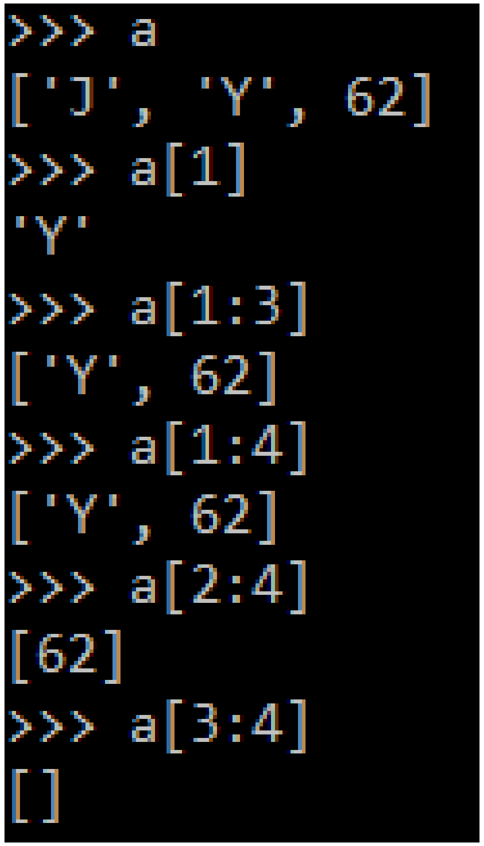
The range of list by a colon ":" notation. a :b means from the inclusive index a to the exclusive index b.

The index "-1" means the last element.

Why?
Why not?

In the previous slide

- Some returns as an element; some others in list.



```
>>> a
['J', 'Y', 62]
>>> a[1]
'Y'
>>> a[1:3]
['Y', 62]
>>> a[1:4]
['Y', 62]
>>> a[2:4]
[62]
>>> a[3:4]
[]
```

List

- Create a list of data as many as you can
- Try the following:
  - `lst[2]`
  - `lst[3]`
  - `lst[3][1]`
  - `lst[2][1]`
- Are all above legal?
- Then do this
  - `lst.append(‘College’)`
  - `lst.append([33,1,33])`
  - `lst.extend([33,1,33])`

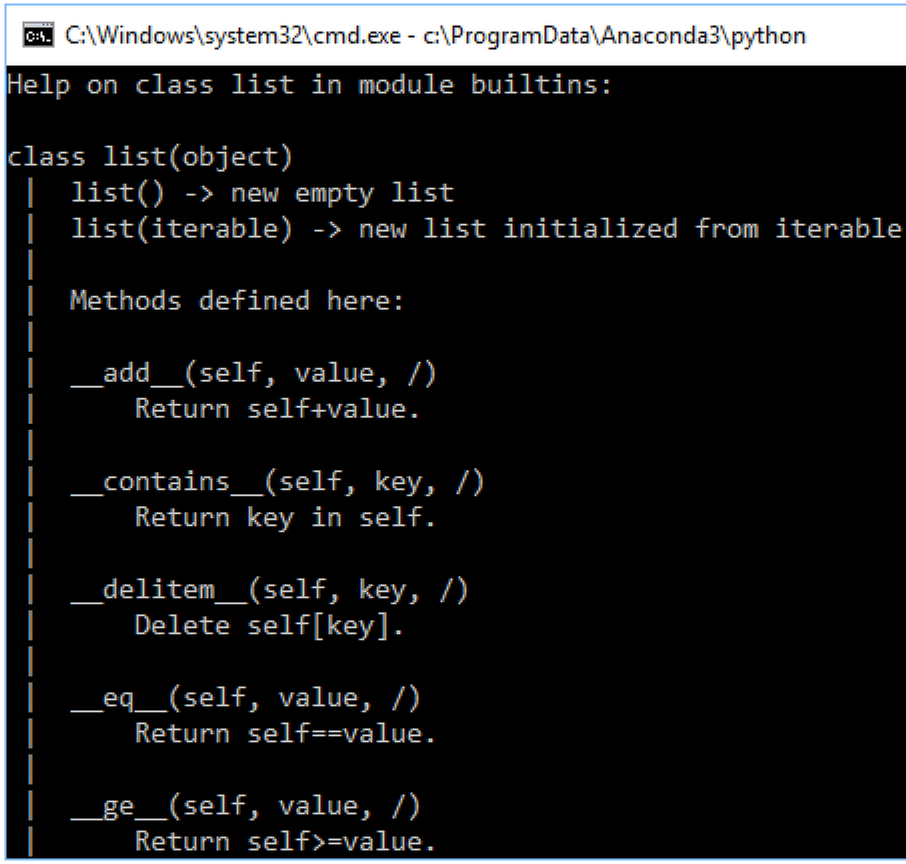
Difference?

- Are the following legal?
  - lst[3,1]
  - lst[3][1]
  - lst[4][1]
  - lst[:4][1]

List

More Methods

- Lookup the API
  - `len(lst)`
  - `max(lst)`
  - `lst.append('College')`
  - `lst.extend([33,1,33])`
  - `lst.count(33)`
  - `del lst[2]`
  - `lst.remove(33)`
- More
  - `insert()`
  - `reverse()`
  - `sort()`
  - `pop()`



C:\Windows\system32\cmd.exe - c:\ProgramData\Anaconda3\python

```
Help on class list in module builtins:

class list(object)
 |  list() -> new empty list
 |  list(iterable) -> new list initialized from iterable
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
```

Quiz on List

- Given any two arbitrary lists, lst1 and lst2
  - Try
  - lst1+lst2
  - lst1.extend(lst2)
- What is difference?

Tuple

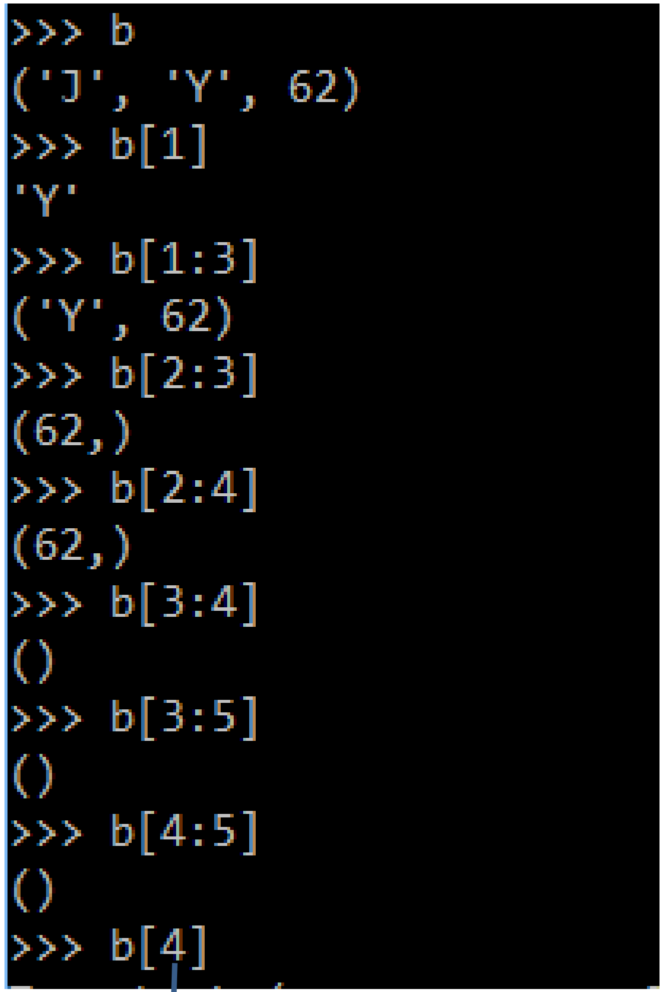
- Try and feel

```
In [91]: tup = ('John', 'faculty', 555)

In [92]: tup[0]
Out[92]: 'John'

In [93]: tup[:1]
Out[93]: ('John')

In [94]: tup[0:2]
Out[94]: ('John', 'faculty')

In [103]: tup[1][3]
Out[103]: 'u'
```

Access by indexing
Return in tuple

In the previous slide

- Some returns as an element; some others in tuple.



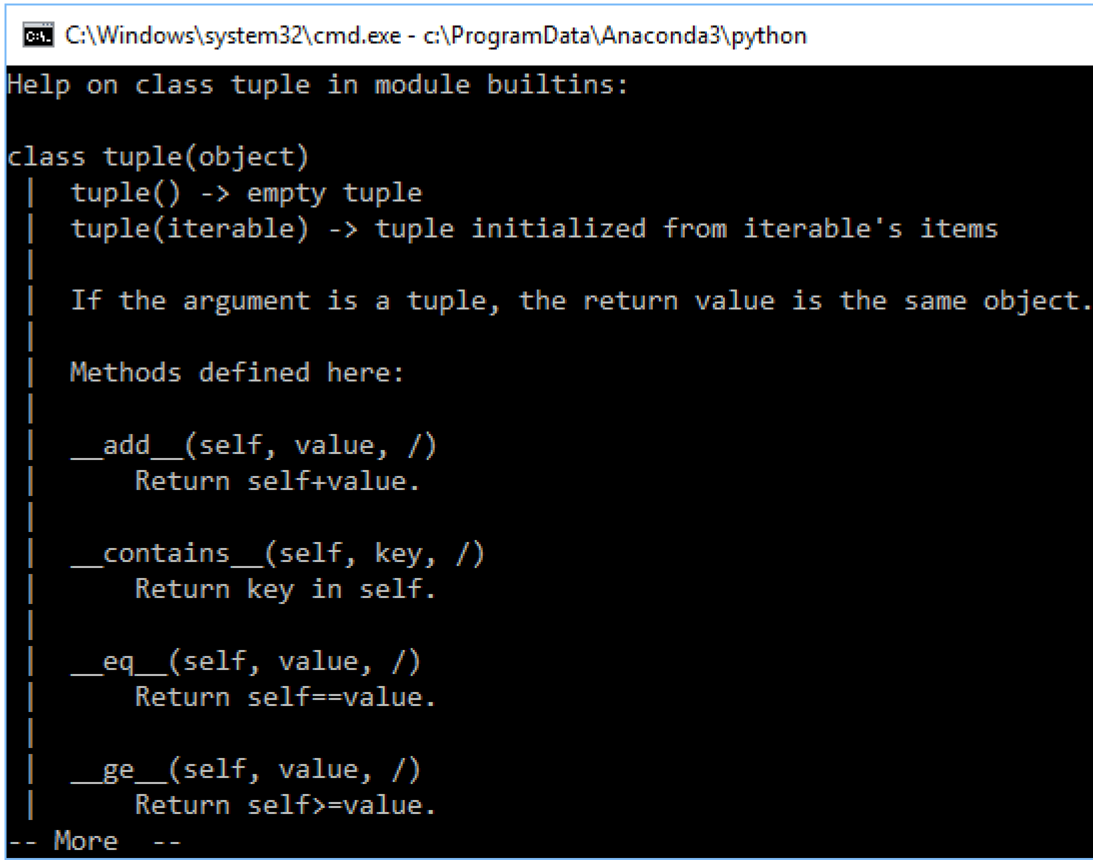
```
>>> b
('J', 'Y', 62)
>>> b[1]
'Y'
>>> b[1:3]
('Y', 62)
>>> b[2:3]
(62,)
>>> b[2:4]
(62,)
>>> b[3:4]
()
>>> b[3:5]
()
>>> b[4:5]
()
>>> b[4]
```

- What is the return?

What about in a list?

Tuple

- Elements in a tuple are accessed in the same notation of lists.
  - To create, use ( )
  - To access, use [ ] notation to indicate with indexes
- Very similar to list
  - Only two methods
    - count()
    - index()
- Methods
  - `cmp(tup,tu2)`
  - `len(tup)`
  - `list(tup)`
  - `max(tup)`
  - `Tup1 + (1,2)`


C:\Windows\system32\cmd.exe - c:\ProgramData\Anaconda3\python

```
Help on class tuple in module builtins:

class tuple(object)
 |  tuple() -> empty tuple
 |  tuple(iterable) -> tuple initialized from iterable's items
 |
 |  If the argument is a tuple, the return value is the same object.
 |
 |  Methods defined here:
 |
 |  __add__(self, value, /)
 |      Return self+value.
 |
 |  __contains__(self, key, /)
 |      Return key in self.
 |
 |  __eq__(self, value, /)
 |      Return self==value.
 |
 |  __ge__(self, value, /)
 |      Return self>=value.
-- More  --
```

Dictionary

- Try and feel

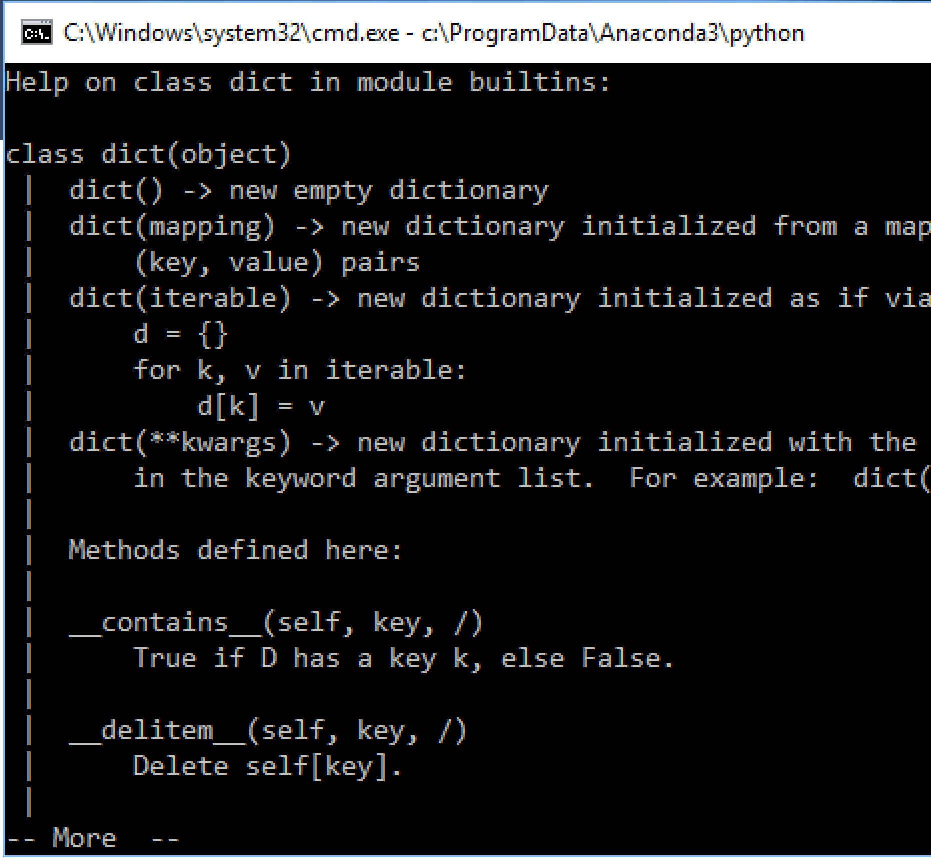
The same returns

```
In [109]: dct = {"USA":"DC",
"Canada":"Ottawa", "S.Korea":"Seoul",
"UK":"London"}


In [110]: dct["USA"]
Out[110]: 'DC'


In [111]: len(dct)
Out[111]: 4


In [112]: dct.get("USA")
Out[112]: 'DC'

In [114]: dct.items()
Out[114]: dict_items([('S.Korea', 'Seoul'),
('USA', 'DC'), ('Canada', 'Ottawa'), ('UK',
'London')])


In [115]: dct.keys()
Out[115]: dict_keys(['S.Korea', 'USA',
'Canada', 'UK'])
```

Dictionary

- Look up the API

- Methods
  - `.get()`
  - `.items()`
  - `.keys()`
  - `.values()`
- Access
  - `.items()[index]`
    - **Error?**
  - `list(.items())`



C:\Windows\system32\cmd.exe - c:\ProgramData\Anaconda3\python

```
Help on class dict in module builtins:

class dict(object)
 |  dict() -> new empty dictionary
 |  dict(mapping) -> new dictionary initialized from a map
 |      (key, value) pairs
 |  dict(iterable) -> new dictionary initialized as if via
 |      d = {}
 |      for k, v in iterable:
 |          d[k] = v
 |  dict(**kwargs) -> new dictionary initialized with the
 |      in the keyword argument list.  For example:  dict(
 |
 |  Methods defined here:
 |
 |  __contains__(self, key, /)
 |      True if D has a key k, else False.
 |
 |  __delitem__(self, key, /)
 |      Delete self[key].
 |
-- More  --
```

Dictionary

- Methods
  - To add
    - `dct.update({"Japan":"Tokyo"})`
    - `dct.update({"U.Korea":["Seoul","Pyungyang"]})`
    - `dct.update({"USA":1})`
  - To remove
    - `dct.pop(“U.Korea”)`
- Example
  - Let `IEEE802std = {802.3: "Ethernet", 802.11: "Wireless LAN", 802.15: "Wireless PAN", "802.15.1": "Bluetooth", "802.15.4": "Low-Rate Wireless PAN"}`
  - How to get the value of 802.11
    - Can we express it using indexes only?
      - Hint: Use list() of dictionary items

HW4

- Consider
  - `IEEE802std = {802.3: "Ethernet", 802.11: "Wireless LAN", 802.15: "Wireless PAN", "802.15.1": "Bluetooth", "802.15.4": "Low-Rate Wireless PAN"}`
- Q1: What is returned when "`IEEE802std`" is issued?
  - Explain the returned value.
- Q2: How about from "`IEEE802std.items()`"? What about "`IEEE802std.iteritems()`"?
  - Hint: One of them is obsolete!
- Q3: Write a statement to list the items of IEEE802std.
  - Hint: use a built-in function

HW4 - continued

- Q4: Consider 802.11 and "802.15*", one in number another in string. Find out why should it be?
- Q5: Is it possible to redefine the structure that 802.15 consists of 802.15.1 which is Bluetooth, 802.15.4 which is low-rate wireless pan? If so, how?
  - Hint: nested dictionary
- Q6: It is possible to define a list of dictionaries and a dictionary of lists.
  - Show your extended examples of each, a list of dictionaries and a dictionary of lists.
  - Extend the domain of network protocols. Hint: https://www.webopedia.com/quick_ref/OSI_Layers.asp

Sample

```
{ "Protocols": [{"Layer_1": { "title" : "Physical layer”,
                              "Protocols" : ["Ethernet", "FDDI", "ATM"],
                              "Technology": ["electrical impulse", "light", "radio signal"]},
                "Layer_2":  # do more work… ,
                …
                "Layer_7": …
                }
               ]
}
```

- Read the above by pairing “{“ with “}” for dictionary, “[“ with “]” for lists.
  - For example,
  - {“Name” : [{“first_name” : “John”, “last_name” : “Yoon”}, {“first_name” : “Chris”, “last_name” : “Park”}], “Building”: “Maher Hall”}
  - Observe the above example for name/value pair-wise structure.
- Note that those pairs are nested as above