

# Recitation 0

## Internet Technology (Section 01)

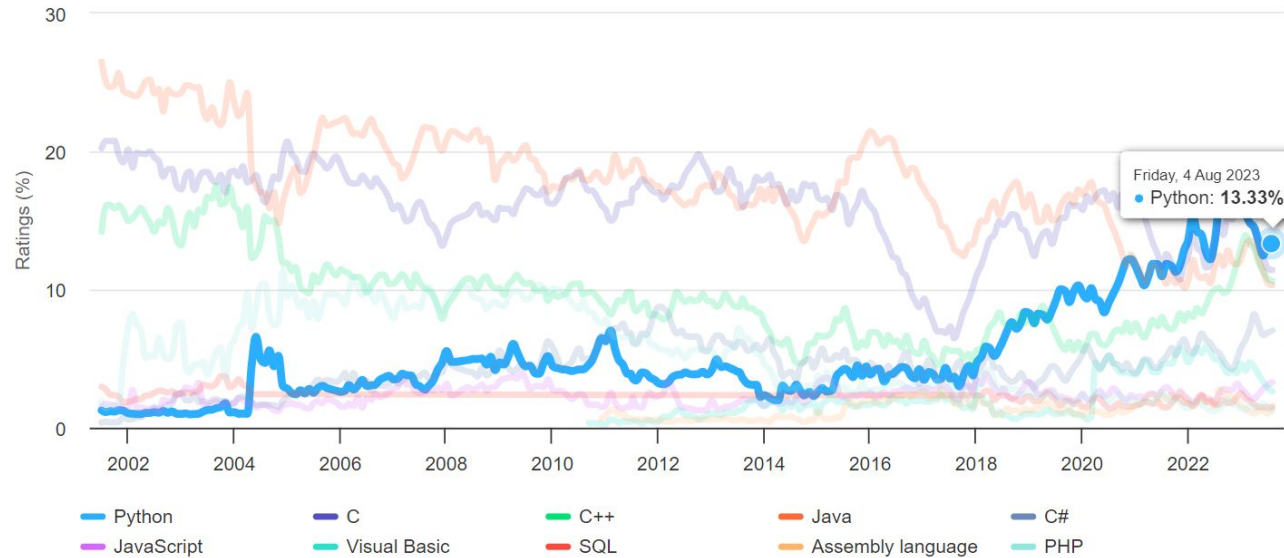
# About Me

- Name: Alborz Jelvani
- Email: [alborz.jelvani@rutgers.edu](mailto:alborz.jelvani@rutgers.edu)
- Office Hours: TBA

# Python3

## TIOBE Programming Community Index

Source: [www.tiobe.com](http://www.tiobe.com)



# Python3 Features

- Object oriented
- **Strongly** typed and **dynamically** typed
- Interpreted

```
jelvani@PC:~$ python3
Python 3.8.10 (default, Nov 14 2022,
12:59:47)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or
"license" for more information.
>>> print("hello!")
hello!
```

```
someVar = 100
print(type(someVar))
someVar = "a string"
print(type(someVar))
>> <class 'int'>
>> <class 'str'>
```

# Control Flow

**Indentation:** 4 spaces

```
While True:
    if cond1:
        doFunc()
    elif cond2:
        doThis()
    else:
        break
```

**Execution begins at top** (*main* not required)

```
#This is a comment
items = ["a", "b", "c"]
abc = ""
for letter in items:
    print(letter)
    abc+=letter
print(abc)
```

# Types

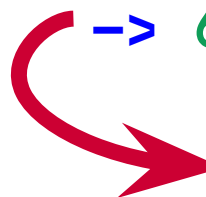
- Strings, integers, floats, bool
  - *str(),int(),float(), bool()*
- Lists
  - Collection of **mutable ordered** elements.
- Set
  - Collection of **mutable unordered** elements. 1 of each elements!
- Tuple
  - Collection of **immutable ordered** elements.
- Dictionary
  - Key-value pairs
- None

```
>>> "string"
>>> 27 #int
>>> 3.14 #float
>>> True #bool
>>> aList = [1,2,3,4,5]
>>> aSet = {1,2,3,4,5}
>>> aTuple = (1,2,3,4,5)
>>> aDict = {1: "one", 2:
             "two", 3: "three"}
>>> a = [None, None]
```

# Expressions

```
>>> 1==2 -> False
>>> "hi" == "hi" -> True
>>> True and False or True
-> True
>>> False or False and True
-> False
```

```
>>> 1+2+3**2//4+5/3
-> 6.666666666666667
```



$$1 + 2 + \left[ \frac{3^2}{4} \right] + \frac{5}{3}$$

```
>>> 10 % 2 == 1
-> False
```

# For Loops

- Iterate over *iterables*
  - Some objects implement iteration
    - `range(start, stop` (required), `step`)
    - `list()`, `str()`, `tuple()`, `dict()`, `set()`...

- Want index?

- Use `enumerate()`
- ```
>>> list(enumerate([23, 56, 65, 36]))  
-> [(0, 23), (1, 56), (2, 65), (3, 36)]
```

```
for i in range(10):  
    print(i)
```

```
animals = ["moose", "marmot", "osprey"]  
for animal in animals:  
    print(animal)
```

```
for idx, val in enumerate(animals):  
    print(idx, val)
```



# Lists

```
>>> items = [] #or list()
>>> type(items) -> <class 'list'>
>>> items.append(1)
>>> items.append("1")
>>> items.append([3.14, 2.71])
>>> items -> [1, '1', [3.14, 2.71]]
>>> [type(item) for item in items]
-> [<class 'int'>, <class 'str'>, <class 'list'>]
>>> 3 in items -> False
```

# List Manipulation

- Couple ways

- Indexing
- Slicing
  - List[first include : last exclude : step]
- Add elements
  - List.append(item)
  - List.insert(index,item)
- Remove elements
  - List.remove(target)
  - List.pop(index)

```
>>> a = [2,1,5,3,1]
>>> a[2] -> 5
>>> a[-1] -> 1
>>> a[:3] -> [2,1,5]
>>> a[::2] -> [2,5,1]
>>> a[1:3:2] -> [1]
>>> a+[30,23]
      -> [2,1,5,3,1,30,23]
>>> a.pop(1) -> 1
>>> a.remove(3); a -> [2,5,1]
```

# List Comprehensions

- A more concise way to create lists

```
>>> a = [0]*10; a -> [0,0,0,0,0,0,0,0,0,0]
```

```
>>> [odd for odd in range(10) if odd%2==1] -> [1,3,5,7,9]
```

- Syntax

- `list2 = [expr(element) for element in list1 filter]`
- 'for' clause to iterate over iterable
- Expression to manipulate elements. The output will populate the new list
- An optional filter composed of 'if' clauses

# Lists (useful methods)

- Get length of list
- Reverse list
- Min/max
- *count(target)*
  - Number of occurrences for *target*
- Sort
- Clear

```
>>> a = [2,1,5,3,1]
>>> len(a) -> 5
>>> a.reverse(); a
      -> [1, 3, 5, 1, 2]
>>> max(a) -> 5
>>> a.count(1) -> 2
>>> a.sort(); a ->
      [1, 1, 2, 3, 5]
>>> a.clear(); a -> []
```

# Binary Data

- *bytes* type

- **immutable** sequence of bytes

```
>>> bite = b"alborz" #or bytes()
>>> bite[1] -> 108
>>> bite.hex() -> '616c626f727a'
>>> bite[3] = 23
```

  
**Nope!**

- *bytearray* type

- **mutable** sequence of bytes

```
>>> bite = bytearray('alborz',
'utf-8')
>>> bite[1] -> 108
>>> bite.hex() -> '616c626f727a'
>>> bite[3] = 75
>>> bite.decode() -> 'albKrz'
```

# *bytearray* Manipulation

- Similar to strings and lists

```
>>> packet = bytearray('payload', 'utf-8')
>>> len(packet) -> 7
>>> packet.append(5)
>>> packet.decode() -> 'payload\x05'
>>> packet.reverse()
>>> packet.decode() -> '\x05dao1yap'
```

# Functions

- Polymorphism

```
def add(a,b):  
    return a + b  
print(add(1,2))  
print(add("a ", "string"))
```

- Default args and keyword args

```
def add(a=4,b=5):  
    return a + b  
print(add(b=1))
```

- Arbitrary args

```
def merge(*lists):  
    a=[]  
    for l in lists:  
        a+=l  
    return a  
print(merge([1],[2],[3]))
```

# How to make a Rutgers CS account

- <https://services.cs.rutgers.edu/accounts/>



## Account Management Tools for Computer Science Systems

Click this

**Note:** Your computer science user ID is your **University NetID**. You must have a University NetID before you can activate a computer science account. Here's the University's tool for activating your NetID: <https://netid.rutgers.edu>

Computer science systems use passwords that are separate from your University password. You can make them the same, but it is somewhat better from a security point of view if you use different passwords.

- **Lifetime of accounts and files**

When users leave the University (or computer science), their accounts are closed. Files are archived. They will be deleted after a year, except for faculty files. Shared directories will be deleted or archived based on the user that owns them.

Sometimes users will have a continuing association with the department even after leaving. Accounts may be continued as guest or retiree accounts. Any faculty member can sponsor a guest. The Computer Science Department handles retirees.

- [Activate an account on a Computer Science system](#)

This will let you create an account for Computer Science Department systems. Computer science faculty, majors, grad students, and students enrolled in computer science courses other than 110, 170 and 494 are eligible for accounts.

- [Set or reset your Computer Science password.](#)

This will show you a University login screen. So as long as you remember your University password, you set or reset your CS password. If you need more security than a simple password can afford, please see [two factor authentication](#).

- [Group and Guest management.](#) This will you create and manage groups. This is useful for three purposes:

- If you want to share files with a group of people, you can create a group that lists the people and then change the permissions of the file so they can access it. For more help with this, see [Sharing files](#).
- If you are authorized (normally faculty) and want to allow grad students or collaborators to access research or instructional systems, you can create a guest group. For help with this, see [Managing your guest users](#).
- If you are running your own computer cluster, you can use groups to control what users are allowed to login.

- Consider giving us contact information, in case we need to reach you for support reasons.

You can set additional information using "ipa user-mod" on any of our systems, e.g. "ipa user-mod YOURNETID --phone=8484452001". "ipa user-mod --help" will give a list of what can be set. "ipa user-show YOURNETID" will show your current information. You only need to do this once. The same information applies to all of our systems. Note that changing first name, lastname, and GECOS won't do anything useful, since we reset them to the official University names nightly.



# Weblogin for Rutgers CS

- <https://weblogin.cs.rutgers.edu/>
  - Login to Rutgers CS account
  - Pick a machine
  
- <https://report.cs.rutgers.edu/nagiosnotes/iLab-machines.html>
  - View machine status

## RECENT CONNECTIONS



kill.cs.rutgers.edu



prolog.cs.rutgers.edu



i-lab1.cs.rutgers.edu

## ALL CONNECTIONS

Choose a host to connect to from the menu, or type a hostname in this box:

Enter Connection URI:

- geneva
- ilab1.cs.rutgers.edu
- ilab2.cs.rutgers.edu
- ilab3.cs.rutgers.edu
- ilab4.cs.rutgers.edu
- kill-all-my-sessions
- assembly.cs.rutgers.edu
- basic.cs.rutgers.edu
- batch.cs.rutgers.edu
- boole.cs.rutgers.edu
- butter.cs.rutgers.edu
- c211-1.cs.rutgers.edu
- c211-2.cs.rutgers.edu
- c211-3.cs.rutgers.edu
- c211-11.cs.rutgers.edu
- c211-12.cs.rutgers.edu
- c211-13.cs.rutgers.edu
- c246-1.cs.rutgers.edu
- c246-2.cs.rutgers.edu
- c329-1.cs.rutgers.edu
- c331-1.cs.rutgers.edu

- cheese.cs.rutgers.edu
- cp.cs.rutgers.edu
- cpp.cs.rutgers.edu
- crayon.cs.rutgers.edu
- data7.cs.rutgers.edu
- dogmatk.rutgers.edu
- frost.cs.rutgers.edu
- grep.cs.rutgers.edu
- h206-2.cs.rutgers.edu
- h257-1.cs.rutgers.edu
- h266-1.cs.rutgers.edu
- h273-1.cs.rutgers.edu
- h275-1.cs.rutgers.edu
- h275-2.cs.rutgers.edu
- h275-g4.cs.rutgers.edu
- h405-2.cs.rutgers.edu
- h410-1.cs.rutgers.edu
- h410-2.cs.rutgers.edu
- h412-2.cs.rutgers.edu
- h414-1.cs.rutgers.edu
- h414-2.cs.rutgers.edu

- java.cs.rutgers.edu
- kill.cs.rutgers.edu
- kilaatu.rutgers.edu
- kleene.cs.rutgers.edu
- kolmogorov.cs.rutgers.edu
- less.cs.rutgers.edu
- lisp.cs.rutgers.edu
- ls.cs.rutgers.edu
- man.cs.rutgers.edu
- mv.cs.rutgers.edu
- pascal.cs.rutgers.edu
- perl.cs.rutgers.edu
- plastic.cs.rutgers.edu
- popsicle.cs.rutgers.edu
- post.cs.rutgers.edu
- prolog.cs.rutgers.edu
- pwd.cs.rutgers.edu
- python.cs.rutgers.edu
- rlab1.cs.rutgers.edu
- rlab2.cs.rutgers.edu
- rlab3.cs.rutgers.edu