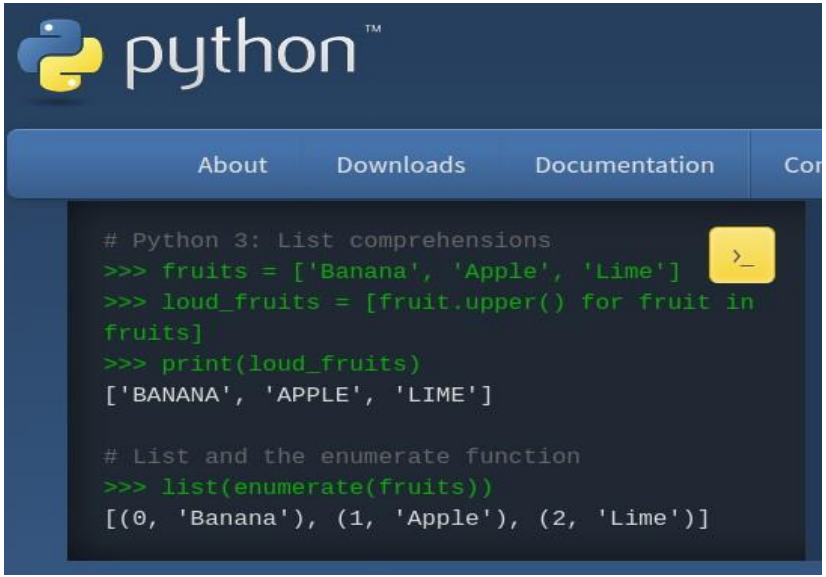
Discrete Structures!

CMPSC 102

ALLEGHENY COLLEGE

Get Python3



- Get Python3 from the Python Software Foundation: https://www.python.org/downloads/
- Or just stick with Jupyter https://jupyter.cs.allegheny.edu/

Install Your Own Python3

- Download and install the version of Python3 for your OS being sure to add the PATH to the environmental variables (check the path option!)
- Check with the installation material to learn how to launch Python3

Running the Python3 Shell

- Type python3 in the terminal to start the Python shell.

- Type statements or expressions at prompt:
    - `print("Hello, world")`
    - `x = 12**2`
    - `print(x)`
    - `print(x/2)`
    - `# bla bla bla...`
        - `(This is a comment: everything after the # is ignored)`

- To exit the Python shell, type: exit() or Ctrl + Z to exit.

Data Types

- Integers, counting numbers
  - `num_int = 1`
- Floats, decimals
  - `num_float = 3.1415`
- Strings:
  - `s_str = "Hello World"`

Data Types

```
height_int = 5
print(f" The height is: {height_int}")
print(" The height is:", height_int) # print another way

num_float = 3.14
print(f" The float variable is : {num_float}")

s_str =("Hello World’’)

print(" The integer is equal to: ", s_str)
```

Key Components

All programs built out of

- **Function calls**: Granting temporary kernel-time and/or using issuing parameters to a sub-sequence of instruction in a program.
- **Assignment statements**: The issuing of a value to a variable or place in memory to contain the value.
- **Iteration constructs**: Structures used in computer programming to repeat the same computer code multiple times (*loops*).
- **Conditional logic**: the use of logical rules in code to govern steps taken.
- **Variable creation**: The introduction of an object in
- memory to contain some value.
- **Variable computations**: The use of values contained in variables to create new value using an operator.
- **Variable output**: The revealing of some value in a variable by printing or another means.

Application - Using Python to Find a Name in a File

```
file = open("names")
for line in file:
    if line.startswith("John")
        print(line)
```

- Can you explain the behavior of this program segment?
- What are the **constructs** inside of this program segment?
- How is this different than a full-fledged Python program?
- What is the purpose of the *open* function?
- What is the purpose of the *line.startswith* function?

Application: Using Python to Find an Email in a File

```
file = open("emails")
for line in file:
 name, email = line.split(",")
 if name == "John Davis":
  print(email)
```

- Can you explain the behavior of this program segment?
- What are the **constructs** inside of this program segment?
- How is this different than a full-fledged Python program?
- What is the purpose of the *open* function?
- What is the purpose of the *line.split* function?

Runnable Application: Using Python to Find an Email in a File

```
#!/usr/bin/env python3
""" Demo program"""

myFile_list =["Bob Bye,bob@big.com", "Julie Roth, Jroth@thinktank.com",
"John Davis, JDavis@KingOfTheWorld.com"]

print("\n Opening myFile :{myFile_str}") #file = open("emails")
for line in myFile_list:
    print(f"\t + line : {line}, {type(line)}") name, email = line.split(",")
    if name == "John Davis":
        print(f"\tName found: {email}")
```

- Can you explain the behavior of this program segment?
- What are the **constructs** inside of this program segment?

Runnable Application: Using Python to Find an Email in a File

```
#!/usr/bin/env python3 """ Demo program"""

mylist =[
  "Bob Bye,bob@big.com",
  "Julie   Roth,Jroth@thinktank.com",
  "John  Davis,JDavis@KingOfTheWorld.com",
  "Tylor Swift,tSwift@Swifter.com",
  "The   Hulk,greenThumb@gardeningHelp.com",
  "Sherlock    Holmes,sHolmes@consultingDetective.com"
    ]

print("\n Opening mylist :{mylist}")

for line in mylist:
  print(f"\t + line : {line}, {type(line)}")
  name, email = line.split(",") if name == "John Davis":
    print(f"\t Name found: {email}")
  if "Sherlock" in name:
    print(f"\t Detective's Name found: {email}")
```

File: openEmail Demo_ii.py

Runnable Application: Using Python to Find an Email in a File

```
#!/usr/bin/env python3 """ Demo program"""

mylist =[
   "Bob Bye,bob@big.com",
   "Julie   Roth,Jroth@thinktank.com",
   "John  Davis,JDavis@KingOfTheWorld.com",
   "Tylor Swift,tSwift@Swifter.com",
   "The   Hulk,greenThumb@gardeningHelp.com",
   "Sherlock    Holmes,sHolmes@consultingDetective.com"
     ]

print("\n Opening mylist :{mylist}")

for line in mylist:
   print(f"\t + line : {line}, {type(line)}")
   name, email = line.split(",") if name == "John Davis":
      print(f"\t Name found: {email}")
   if "Sherlock" in name:
      print(f"\t Detective's Name found: {email}")
```

File: openEmail Demo_ii.py

Runnable Application: Using Python to Find an Email in a File

```
#!/usr/bin/env python3
""" Demo program"""

myFile = [1,2,3,4,5,6,7,8,9,10]
sum = 0
count = 0
for line in myFile:
  n = int(line)
  sum += n
  count += 1
print(sum/count)
```

File: getAverage demo.py

- Can you explain the behavior of this program segment?
- What are the **constructs** inside of this program segment?

Runnable Application: Using Python to Find an Email in a File

```
#!/usr/bin/env python3
""" Demo program"""

sum = 0
count = 0
myFile = open("data.txt")
for line in myFile:
  n = int(line)
  sum += n
  count += 1
print(sum/count)
```

File: getAverage file.py

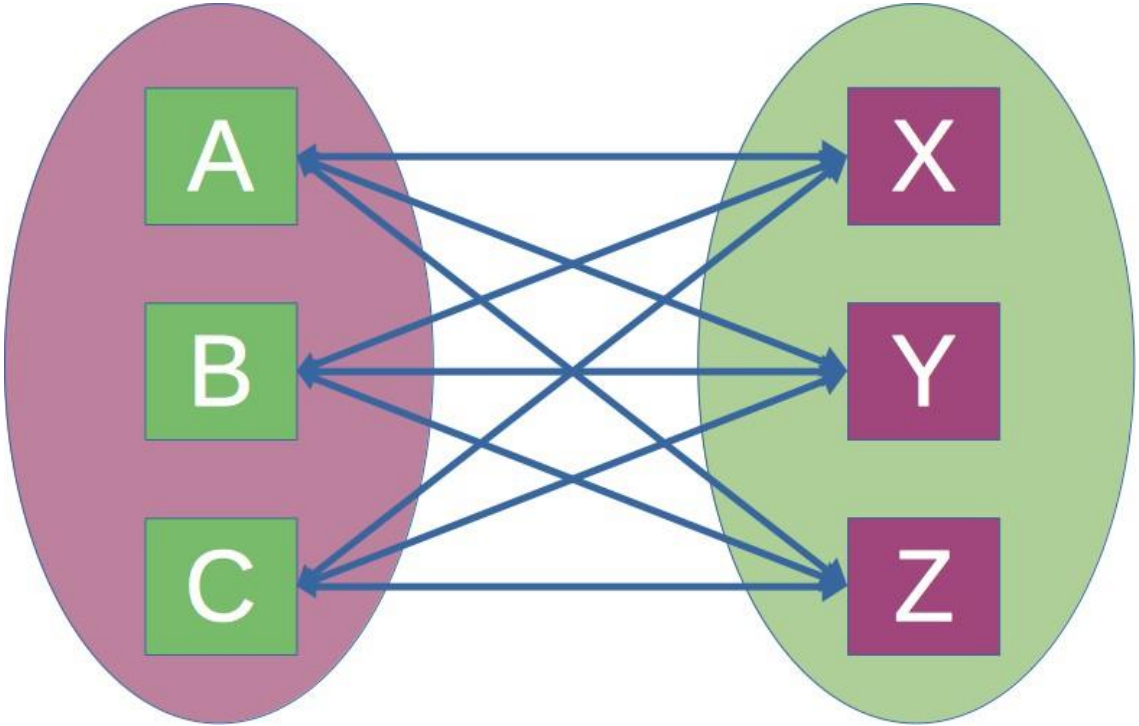
- What are the contents of the data.txt file?
- What is the purpose of the for line in file statement?

Mathematical Terminology

- Mathematical terminology is a vocabulary for discussing Python programs
- What are mathematical terms that aid programming?
- **Set**: an unordered collection of different entities
- **Sequence**: an ordered collection of entities
- **Relation**: a set that relates pairs of things with each other
- **Mapping**: a set of ordered pairs in every element is unique (sometimes called a "function" in mathematics)
- Can you find these mathematical concepts in the Python programs? For instance: *What is a file*?

Mathematical Terminology

Is this a function?

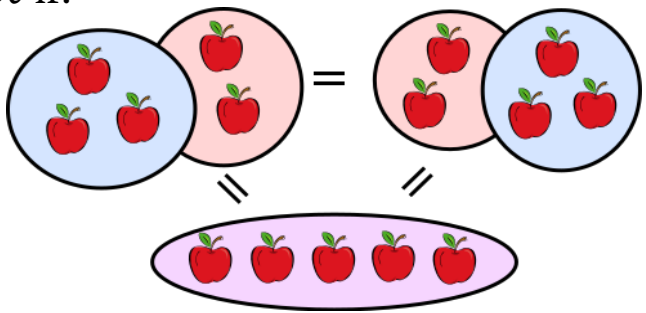


General Properties of Real Numbers

Property	Addition	Multiplication
Commutative	$a + b = b + a$	$a \cdot b = b \cdot a$
Associative	$a + (b + c) = (a + b) + c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c$
Distributive	$a \cdot (b + c) = a \cdot b + a \cdot c$	$a \cdot (b + c) = a \cdot b + a \cdot$
Identity	$a + 0 = a$	$a \cdot 1 = a$
Inverse	$a + (-a) = 0$	$a \cdot \frac{1}{a} = 1$

Properties - Commutative

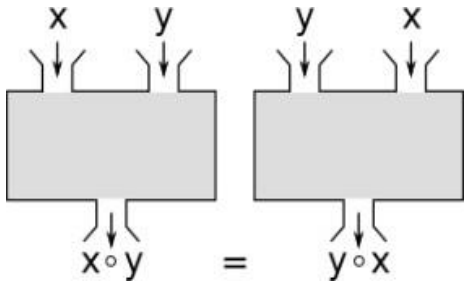
- The term "commutative" is used in several related senses. A binary operation $*$ on a set S is called *commutative* if:
- $x * y = y * x$ for all $x, y \in S$
    - An operation that does not satisfy the above property is called *non-commutative*.
- One says that x *commutes* with y under $*$ if: $x * y = y * x$
- A binary function $f : A \times A \to B$ is called *commutative* if:
- $f(x, y) = f(y, x)$ for all $x, y \in A$



Properties - Examples

Commutative

- The operator each side of equation do not create inequality
- Think operators like: Addition, multiplication, division



Not Commutative

- The operator each side of equation creates inequality
- Think operators like: subtraction
- $x - y \neq y - x$; $5 - 3 \neq 3 - 5$

Properties - Non-Commutative operations

- Washing and drying clothes resembles a noncommutative operation; washing and then drying produces a markedly different result to drying and then washing.
- Putting on left and then right socks on feet is commutative
- Putting on shirt and then sweater is not-commutative

Strings:

```
a = "face"
b = "book"
a + b == b + a # run the test!
"facebook" != "bookface"
```

Practical Variable Limitations in Python

Programming has computational limits

Python Output:

```
>>> 2**2**8 # a really long number
115792089237316195...584007913129639936

>>> 2**2**10 # a very, very long number!!
17976931...6329624224137216

>>> 2**2**100
^CTraceback (most recent call last): File
  "stdin", line 1, in module
KeyboardInterrupt
```

Mathematical thinking is infinite unlike computational wisdom

Practical Variable Limitations in Python

More computational limits

Python Output:

```
>>> 1.0 == 1.1
False
>>> 1.0 == 1
True
>>> 'h' + 'i' + '!'
'hi!'
>>> .33333 + .33333 + .33333 == 1
False
>>> .33333333333 + .33333333333 + .3333333333 == 1
False
>>> 1/3 0.3333333333333333
>>> 1/3 + 1/3 + 1/3 == 1
True
```

File: explore-python-variables.ipynb

Test Your Understanding

- Understanding the **connections** between
- **mathematics** and **programming**:
  - **Q1**: What is a **mapping** in the mathematics?
  - **Q2**: What is a **function** in mathematics and Python?
  - **Q3**: What are the **limits** for variables in the Python language?
  - **Q4**: What kinds of computational limits exist in Python? Or for any programming?