Python Built-in Objects

Objective: use Python built-in objects to store and manipulate information similar to an application

- learn to manipulate objects and their data just like you would on a regular spreadsheet.
- Why? This is how modern IT work gets done.
- If you know lists, tuples, dictionary, strings, you can grab data from anywhere and work with it

Concepts apply to all systems analysis tools concepts to perform current and future IT system

Mechanics

▪ mylist = [	‘a’,	‘b’,	‘c’,	10,	20,	30	]
a. iterator/index [i]	0	1	2	3	4	5	
b. len(mylist)	|->		~ n=six~			<-|	
a. print(mylist[i])	‘a’	‘b’	‘c’	10	20	30	

Description

▪ create the data for list, tuple, etc

a. iteration is the count; index is the numeric position number

b. len() inherits count of total items from mylist

c. for i in mylist: print(mylist[i]) # prints each position

Lists

- group similar\dissimilar information
- mutable (can change data)
- sequential with an ID# per position
- organize similar\dissimilar information
- modify: mylist.append(),.insert(),.pop()

```
mylist = []
mylist = ['bambam', "a+b=c", 2_0j, [1,2,3]]
for i in mylist: print(i)
```

```
bambam
a+b=c
20j
[1, 2, 3]
```

comprehension places formula before iterator to generate data

```
mylist =[i*2 for i in range(0,4) ]; mylist
```

```
[0, 2, 4, 6]
```

```
mytuple = (0,1,3,4)
mylist = [i*3 for i in mytuple]; mylist
```

```
[0, 3, 9, 12]
```

```
me1 = ['adam','carly','jackson','danny']
dict(enumerate(me1,start=100))
```

```
{100: 'adam', 101: 'carly', 102: 'jackson', 103: 'danny'}
```

Tuples

- sequential data object
- sequential with an ID# per position
- practical reference table to other data
- need a trailing comma!=>(1,2,)

```
mytuple = (1,2,3,)
mytuple = ('snhu', 2+0j, [1,2,3] )
```

```
('snhu', (2+0j), [1, 2, 3])
```

```
mytuple_dict_keys = (1,2,3,)
```

Dictionary

- essential for pairing related data
- go-to-tool for real-world modeling
- dict would reference your unique ID and an associated list would have the characteristic data in
- data unordered & random

```
mydict = {key : <-      values     -> }
mydict =
{"id.1":['first','last','age','height']}
```

```
{'id.1': ['first', 'last', 'age', 'height']}
```

```
minions = ['kevin','warzog','nano', 'oscar']
leader = ['commander lambda']
mydict = {"leader":leader,"team1":minions_1}
mydict
```

```
{'leader': ['commander lambda'], 'team1': {'nano', 'oscar'}}
```

```
minions = ['warzog','oscar']
leader = ['Sgt Lambda','Sgt Pi']
mydict = dict(zip(['Sgt Lambdi','Sgt
Pi'],['warzog','oscar']))
mydict
```

```
{'Sgt Lambdi': 'warzog', 'Sgt Pi': 'oscar'}
```

Functions

```
.append() #add item
```

```
.pop([i]) #remove specific item. -1=last
```

```
dict() #creates a dict. object
```

Iterators

Iterators
- **for**
- **range**
- **while**

- the act of looping instructions repeatably
- instructions continuously repeating until reaching a termination
- performing tasks continuntil end of range, data
- most efficient means to cycle information in lists, tuples, ranges, and sets

- Iterators are sequential like 0->1->2->3
- Python sequential objects= list, range, tuple

Mechanics

```
   ▪         mylist  = [        ‘a’,   ‘b’,   ‘c’,  10,    20,    30   ]
b. iterator/index [i]     0     1     2      3       4      5
c.        len(mylist)    |->           ~ n=six~           <-|
d. print( mylist[i]*3)      aaa        bbb   ccc   30    120    150
e. negative index [i]              -6     -5     -4    -3    -2     -1
   for i in mylist: print(mylist[i]*3)
```

Mechanics Description

▪ create the data for list, tuple, etc

d. iteration is the count; index is the position

e. len() inherits count of total items from mylist

f. for i in mylist:
```
      print(mylist[i]*3) #multiply each list iterate *3
```

g. negative index is neg. number values for an sequence position

for i in <object>:

- starts from 0 for all items in the object
- inherits length from object
- i shorthand for iterator
- regularly combined with conditional statements to make decisions if-elif-else

while i <= <value/object>:

- use to iterate in a forward or reverse direction

range(start,stop,step)

- use set a numeric range to iterator or calculate with
- default start is zero and default setp is one
- may inherit values form use objects, attributes

Misc

- row for row in open (‘filepath.txt’)
- generator <fix this>

sum((i*3 for i in range(2))

Examples

```
 mylist = [1,4]
for i in mylist:
    print(i*3)
```

```
3
12
```

```
for i in range (0,2,1):
    print("loop#{a}, value={b}".
format(a=i,b=(my_formula(i))))
```

```
loop#0, value=0
loop#1, value=2
```

Generator function

```
from math import log10
def myfunction(x):
    return log10(x)
myL = [1,2,3]
data = (round(myfunction(i),3) for
i in myL)
print(list(data))
```

```
[0.0, 0.301, 0.477]
```

Examples

```
i = 0
while i <=1:
  print(i)
  i +=1
```

```
0
1
```

```
i=1
while i < 2:
    print("loop#
i={}".format(str(i)))
    i +=1
print("final loop i is ="+str(i))
```

Examples

```
for i in range(0,2): print(i)
```

```
0
1
```

```
for i in range(len(<object>)):
  print[i]
```

Examples

```
with open (‘path of file.txt’,
‘r’) as data_file:
  for line in data_file:
    print(line)
```

-Quickly create lists or dict with-

enumerate auto adds list index

```
#
me1 =
['adam','carly','jackson','danny']
me2 = list(enumerate(me1)); me2
```

```
[(0, 'adam'), (1, 'carly'), (2, 'jackson'), (3,
'danny')]
```

essetial Functions	Functions are the workhorses helping transform, transpose, combine and just about anything else you can think of

Mechanics	Description
• each function has unique parameters (values it accepts) and means of operating. To figure out read the docs and when necessary look for examples on stackoverflow, jupyterform, and google but try to be selective so your time is not wasted	it.304 – choose 2-3 and write an example

Built-in Functions

A
abs()
aiter()
all()
any()
anext()
ascii()

B
bin()
bool()
breakpoint()
bytearray()
bytes()

C
callable()
chr()
classmethod()
compile()
complex()

D
delattr()
dict()
dir()
divmod()

E
enumerate()
eval()
exec()

F
filter()
float()
format()
frozenset()

G
getattr()
globals()

H
hasattr()
hash()
help()
hex()

I
id()
input()
int()
isinstance()
issubclass()
iter()

L
len()
list()
locals()

M
map()
max()
memoryview()
min()

N
next()

O
object()
oct()
open()
ord()

P
pow()
print()
property()

R
range()
repr()
reversed()
round()

S
set()
setattr()
slice()
sorted()
staticmethod()
str()
sum()
super()

T
tuple()
type()

V
vars()

Z
zip()

_
__import__()

```python
def sum(a, b):
    return (a + b)

a = int(input('Enter 1st number: '))
b = int(input('Enter 2nd number: '))

print(f'Sum of {a} and {b} is {sum(a, b)}')
```

```
Enter 1st number :                    1
Enter 2nd number:                     2
Sum of 1 and 2 is                     3
```

Objects

Building your own Object 'class'

a. **Classes** are a framework for creating objects, functions specific to an object family, attributes, and child class via inheritance
b. **Objects** are entities that perform work.
c. **Methods** are instructions detailing "how" to perform work. Built parent or child level.
d. **Attributes** are alpha\numeric values associated with an object or class. Methods can use this values to perform work and make decisions
e. **self** <self.attribute> is the first argument in a class function self-identifying itself while processing instructions
f. **Function** – instructions to perform a task independent of any object. Methods are functions but associated with an object.

```
#create parent object
mydict = {"training_tests":[], "total_GEMMs ":0}
class myMouse:
    pass
    name = ""
    GEMM_class = ""  #genetically modified mice
    obstacle = ""
#create a function to inventory training performed
def add_tests(traintype):
    mydict["work_performed "].append(traintype)
    mydict["mice_involved"] =+1

#create 2 unique children
m1 = myMouse ()     # a is shorthand for animal
m2 = myMouse ()     # <object names user defined>

#add some characteristics
m1.name = "mighty_mouse"
m1.GEMM_class = "knock-in"
m1.obstacle = "maze_run"
add_tests(m1.obstacle)          #updates the reporting function
m2.name = "wiggle_wig"
m2.GEMM_class = "knock-out"
m2. obstacle = "running_wheel"
add_tests(m2.obstacle)          #updates the reporting function

#simple report <we will have more examples week 4>
mydict_rpt = {m1.name:m1. GEMM_class, m2.name:m2.GEMM_class,"metrics=>":mydict}
    mydict_rpt
```

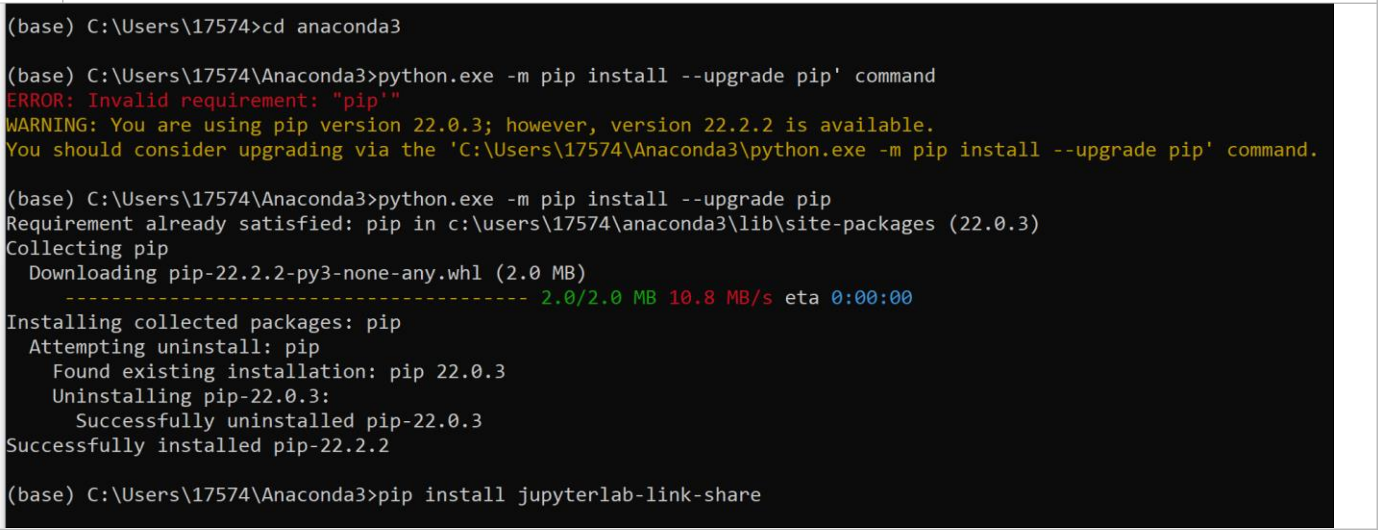
```
{mighty_mouse: 'knock-in',
 'wiggle_wig': knock-out,
 'metrics=>': {'training_tests ': ['maze_run, '  running_wheel '], 'total_GEMMs: 2}}
```

Installation	•	• Warning <for less experienced it.minions • Take your time and read prompts •	Critical source locations Python Package Index = source repository of Python
	Mechanics		Description

Upgrading your Jupyter labs to use share doc feature

- https://jupyterlab.readthedocs.io/en/stable/getting_started/installation.html
- Python Package Index = source repository of Python software (https://pypi.org/)

Task	Instructions
Using terminal\ command line 1) upgrade pip <installation engine) a. https://pypi.org/project/pip/ b. this installs pip-22.2.2	C:\users\17574\anaconda3\python.exe -m pip install --upgrade pip
2) upgrade jupyter notebooks a. done on command line either conda or pip	command line: conda install -c conda-forge jupyterlab
3) add the share notebook feature a. github source b. https://github.com/jupyterlab-contrib/jupyterlab-link-share	command line: pip install jupyterlab-link-share
Open jupyter notebook I GET THERE USING Anaconda Prompt #will then open and run in browswer	cL\Users\<your_computer_name>jupyter-lab



```
(base) C:\Users\17574>cd anaconda3

(base) C:\Users\17574\Anaconda3>python.exe -m pip install --upgrade pip' command
ERROR: Invalid requirement: "pip'"
WARNING: You are using pip version 22.0.3; however, version 22.2.2 is available.
You should consider upgrading via the 'C:\Users\17574\Anaconda3\python.exe -m pip install --upgrade pip' command.

(base) C:\Users\17574\Anaconda3>python.exe -m pip install --upgrade pip
Requirement already satisfied: pip in c:\users\17574\anaconda3\lib\site-packages (22.0.3)
Collecting pip
  Downloading pip-22.2.2-py3-none-any.whl (2.0 MB)
     ---------------------------------------- 2.0/2.0 MB 10.8 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.0.3
    Uninstalling pip-22.0.3:
      Successfully uninstalled pip-22.0.3
Successfully installed pip-22.2.2

(base) C:\Users\17574\Anaconda3>pip install jupyterlab-link-share
```

Fun with formatting

Lists	Tuples	Dictionary	Strings

• tbd
•
`me1 = ['adam','carly','jackson','danny']`
`for i, person in enumerate(me1):`
`    print("{}st position is {}".format(i+1,person))`
1st position is adam
2st position is carly
3st position is jackson
4st position is danny
	mytuple=		