1-PythonReview

December 9, 2014

In [5]:
```
# enter goes to new line in same cell
a = 10
b = 20
c = a + b
print(a, b, c)
# shift-enter executes cell
```

```
10 20 30
```

In [11]:
```
def f(x):
    ' my simple single variable function '
    y = -5 + 2*x + 0.6*x**2
    return y
```

In [7]:
```
f(7)
```

Out[7]:
```
38.4
```

In [9]:
```
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib

WARNING: pylab import has clobbered these variables: ['f']
`%matplotlib` prevents importing * from pylab and numpy
```

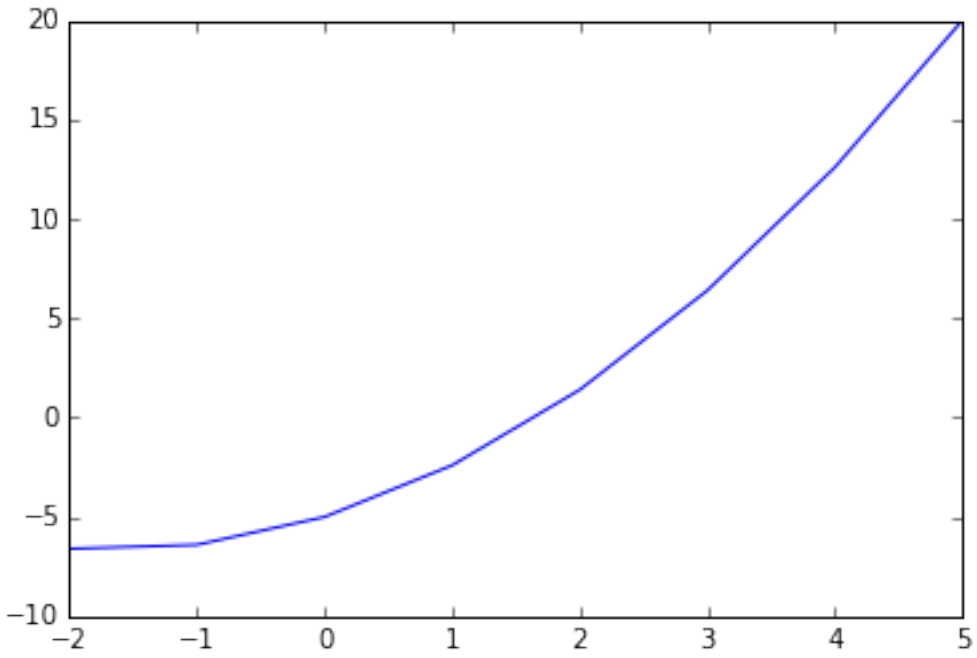
In [12]:
```
x = [-2, -1, 0, 1, 2, 3, 4, 5]
y = [f(i) for i in x]
```

In [13]:
```
import matplotlib as mpl
```

In [14]:
```
mpl.pylab.plot(x, y)
```

```
ERROR! Session/line number was not unique in database. History logging moved to new session 1079
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x108ed6c88>]
```




```
In [15]: pwd
```

```
Out[15]: '/Users/ijstokes/Dropbox/Public/python-mastery-isr19/python-mastery-isr19-code'
```

```
In [16]: ls
```

```
0-PythonMastery.ipynb  demo-python-math*      ianmath.py
1-PythonReview.ipynb   demopythonmath.py*     ianmath.pyc
README.md              demopythonmath.pyc     nbs/
```

```
In [18]: name = 'Ian Stokes-Rees'
```

```
In [20]: print('%40s' % name)
```

```
Ian Stokes-Rees
```

```
In [26]: val = 3.8365
         print('%07.2g' % val)
```

```
00003.8
```

```
In [29]: print('float: %07.2g name= %20s int# %3i' % (val, name, 34))
```

```
float: 00003.8 name=      Ian Stokes-Rees int#  34
```

```
In [30]: pwd
```

```
Out[30]: '/Users/ijstokes/Dropbox/Public/python-mastery-isr19/python-mastery-isr19-code'
```

```
In [31]: cd ../student/Data/ # NOT PYTHON!
```

```
/Users/ijstokes/Dropbox/Public/python-mastery-isr19/student/Data
```

```
In [32]: pwd # ALSO NOT PYTHON!

Out[32]: '/Users/ijstokes/Dropbox/Public/python-mastery-isr19/student/Data'

In [33]: ls # AGAIN, NOT PYTHON!

ctabus.csv      portfolio.dat*  portfolio3.dat* words.txt
dowstocks.dat*  portfolio1.dat* prices.csv
portfolio.csv*  portfolio2.dat* stocksim.py*

In [34]: !wc words.txt

6      29     142 words.txt

In [35]: !cat words.txt

look into my eyes
look into my eyes
the eyes the eyes the eyes
not around the eyes
don't look around the eyes
look into my eyes you're under

In [36]: # back to Python!
         fh = open('words.txt')

In [37]: !ls ..

Data                    PythonMasteryBinder.pdf Solutions
Exercises               README.html             pythonmaster.zip
Optional                RunIDLE.pyw

In [38]: !pwd

/Users/ijstokes/Dropbox/Public/python-mastery-isr19/student/Data

In [39]: !ls

ctabus.csv     portfolio.csv  portfolio1.dat portfolio3.dat stocksim.py
dowstocks.dat  portfolio.dat  portfolio2.dat prices.csv     words.txt
```

0.1 Exceptions

Python 3.4 builtin Exceptions: https://docs.python.org/3.4/library/exceptions.html

BAFP vs LBYL: Python is a "BAFP" language

BAFP:

- Better to Ask Forgiveness than Permission
- Just try it, and if it doesn't work out, an exception will be raised
- DON'T go implementing all sorts of input validation on internal/API functions: leave it up to the caller to call your functions/classes correctly. If they don't, they'll get an exception.

Exception Handling:

Both these conditions must hold to catch an exception:

1. You reasonably expect the exception
2. You are sure you know how to deal with it and should be responsible for it.

```
In [1]: words = 'foo bar zip zap ping pong'.split()

In [2]: words

Out[2]: ['foo', 'bar', 'zip', 'zap', 'ping', 'pong']

In [3]: words[3]

Out[3]: 'zap'

In [4]: words[-1]

Out[4]: 'pong'

In [5]: words[5]

Out[5]: 'pong'

In [6]: words[10] # what is going to happen with this index?


        ---------------------------------------------------------------------------
    IndexError                                Traceback (most recent call last)

        <ipython-input-6-9ffc748e089c> in <module>()
    ----> 1 words[10] # what is going to happen with this index?


        IndexError: list index out of range


In [20]: # Python raises an exception!
         index = 3
         try:
             print("Getting word from index:", index)
             print("word:", words[index])
             print("Phew, loooks like it was OK")
             words.upper()
             print("And now words:", words)
         except (IndexError, AttributeError) as ex:
             print("Oh dear, an exception occurred!", ex)
         except TypeError as ex:
             print("Looks like the index wasn't an int:", ex)
         finally:
             print("This ALWAYS gets invoked")

Getting word from index: 3
word: zap
Phew, loooks like it was OK
Oh dear, an exception occurred! 'list' object has no attribute 'upper'
This ALWAYS gets invoked

In [28]: import string
         for attr in dir(__builtins__):
             if attr[0] in string.ascii_uppercase:
                 print(attr)
```

```
ArithmeticError
AssertionError
AttributeError
BaseException
BlockingIOError
BrokenPipeError
BufferError
BytesWarning
ChildProcessError
ConnectionAbortedError
ConnectionError
ConnectionRefusedError
ConnectionResetError
DeprecationWarning
EOFError
Ellipsis
EnvironmentError
Exception
False
FileExistsError
FileNotFoundError
FloatingPointError
FutureWarning
GeneratorExit
IOError
ImportError
ImportWarning
IndentationError
IndexError
InterruptedError
IsADirectoryError
KeyError
KeyboardInterrupt
LookupError
MemoryError
NameError
None
NotADirectoryError
NotImplemented
NotImplementedError
OSError
OverflowError
PendingDeprecationWarning
PermissionError
ProcessLookupError
ReferenceError
ResourceWarning
RuntimeError
RuntimeWarning
StopIteration
SyntaxError
SyntaxWarning
SystemError
SystemExit
```

```
TabError
TimeoutError
True
TypeError
UnboundLocalError
UnicodeDecodeError
UnicodeEncodeError
UnicodeError
UnicodeTranslateError
UnicodeWarning
UserWarning
ValueError
Warning
ZeroDivisionError
```

```
In [29]: # list comprehension version
         [attr for attr in dir(__builtins__) if attr[0] in string.ascii_uppercase]
```

```
Out[29]: ['ArithmeticError',
          'AssertionError',
          'AttributeError',
          'BaseException',
          'BlockingIOError',
          'BrokenPipeError',
          'BufferError',
          'BytesWarning',
          'ChildProcessError',
          'ConnectionAbortedError',
          'ConnectionError',
          'ConnectionRefusedError',
          'ConnectionResetError',
          'DeprecationWarning',
          'EOFError',
          'Ellipsis',
          'EnvironmentError',
          'Exception',
          'False',
          'FileExistsError',
          'FileNotFoundError',
          'FloatingPointError',
          'FutureWarning',
          'GeneratorExit',
          'IOError',
          'ImportError',
          'ImportWarning',
          'IndentationError',
          'IndexError',
          'InterruptedError',
          'IsADirectoryError',
          'KeyError',
          'KeyboardInterrupt',
          'LookupError',
          'MemoryError',
          'NameError',
          'None',
```

```
 'NotADirectoryError',
 'NotImplemented',
 'NotImplementedError',
 'OSError',
 'OverflowError',
 'PendingDeprecationWarning',
 'PermissionError',
 'ProcessLookupError',
 'ReferenceError',
 'ResourceWarning',
 'RuntimeError',
 'RuntimeWarning',
 'StopIteration',
 'SyntaxError',
 'SyntaxWarning',
 'SystemError',
 'SystemExit',
 'TabError',
 'TimeoutError',
 'True',
 'TypeError',
 'UnboundLocalError',
 'UnicodeDecodeError',
 'UnicodeEncodeError',
 'UnicodeError',
 'UnicodeTranslateError',
 'UnicodeWarning',
 'UserWarning',
 'ValueError',
 'Warning',
 'ZeroDivisionError']
```

```
In [25]: name = 'Ian Stokes-Rees'
```

```
In [26]: name[5]
```

```
Out[26]: 't'
```

```
In [27]: name[0]
```

```
Out[27]: 'I'
```

```
In [46]: def check_value(val):
             if val < 0:
                 raise ValueError('Negative Value: ' + str(val))
             elif val > 9:
                 raise NameError('Too big! Only 0-9 allowed: ' + str(val))
             else:
                 print("Ah, just right: " + str(val))

         n = 15
         try:
             check_value(n)
         except ValueError as ex:
             print("Something is wrong with n:", n, "ex:", ex)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

    <ipython-input-46-c26a4286de36> in <module>()
      9 n = 15
     10 try:
---> 11     check_value(n)
     12 except ValueError as ex:
     13     print("Something is wrong with n:", n, "ex:", ex)


    <ipython-input-46-c26a4286de36> in check_value(val)
      3         raise ValueError('Negative Value: ' + str(val))
      4     elif val > 9:
----> 5         raise NameError('Too big! Only 0-9 allowed: ' + str(val))
      6     else:
      7         print("Ah, just right: " + str(val))


    NameError: Too big! Only 0-9 allowed: 15
```

In []:

```
In [47]: if val < 0:
             raise NegativeValueError('Negative value not allowed: ' + str(val))
         elif val > 9:
             raise BiggerThan9Error('Too big! Only 0-9 allowed: ' + str(val))
         else:
             print("Ah, just right: " + str(val))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

    <ipython-input-47-94e7b7cfc140> in <module>()
      2     raise NegativeValueError('Negative value not allowed: ' + str(val))
      3 elif val > 9:
----> 4     raise BiggerThan10Error('Too big! Only 0-9 allowed: ' + str(val))
      5 else:
      6     print("Ah, just right: " + str(val))


    NameError: name 'BiggerThan10Error' is not defined
```

```
In [48]: class BiggerThan9Error(ValueError):
             ' Raised when a value is (surprise!) bigger than 9 '
             pass

         class NegativeValueError(Exception):
             ' Raised when a value is negative (and not expected to be so) '
             pass
```

```
In [51]: def check_val(val):
             if val < 0:
                 raise NegativeValueError('Negative value not allowed: ' + str(val))
             elif val > 9:
                 raise BiggerThan9Error('Too big! Only 0-9 allowed: ' + str(val))
             else:
                 print("Ah, just right: " + str(val))

         n = 12
         check_val(n)
```

```
---------------------------------------------------------------------------
BiggerThan9Error                          Traceback (most recent call last)

<ipython-input-51-9bd787756f40> in <module>()
  8
  9 n = 12
---> 10 check_val(n)


<ipython-input-51-9bd787756f40> in check_val(val)
  3         raise NegativeValueError('Negative value not allowed: ' + str(val))
  4     elif val > 9:
----> 5         raise BiggerThan9Error('Too big! Only 0-9 allowed: ' + str(val))
  6     else:
  7         print("Ah, just right: " + str(val))


BiggerThan9Error: Too big! Only 0-9 allowed: 12
```

```
In [54]: # BiggerThan9Error is a subclass of ValueError
         # so catching ValueError will also catch BiggerThan9Error

         n = -8

         try:
             check_val(n)
         except ValueError as ex:
             print("Value problem with n:", n, ex)
```

```
---------------------------------------------------------------------------
NegativeValueError                        Traceback (most recent call last)

<ipython-input-54-153a0bc4c267> in <module>()
  5
  6 try:
----> 7     check_val(n)
  8 except ValueError as ex:
  9     print("Value problem with n:", n, ex)


<ipython-input-51-9bd787756f40> in check_val(val)
```

```
      1 def check_val(val):
      2     if val < 0:
----> 3         raise NegativeValueError('Negative value not allowed: ' + str(val))
      4     elif val > 9:
      5         raise BiggerThan9Error('Too big! Only 0-9 allowed: ' + str(val))


NegativeValueError: Negative value not allowed: -8
```

0.2 Classes

```
In [55]: class Point:
             ' A class to represents 2-d points in space '
             pass

In [56]: a = Point() # create an instance of Point

In [57]: a

Out[57]: <__main__.Point at 0x106c09e10>

In [58]: hex(id(a))

Out[58]: '0x106c09e10'

In [59]: b = Point()

In [60]: b

Out[60]: <__main__.Point at 0x106c09668>

In [61]: isinstance(a, Point)

Out[61]: True

In [62]: isinstance(b, Point)

Out[62]: True

In [63]: a.x = 3 # add an attribute to a

In [65]: a.x # read an attribute from a

Out[65]: 3

In [66]: a.color = 'red'

In [67]: a.color

Out[67]: 'red'

In [68]: a.y = 4

In [69]: b.x = 10
         b.y = 9

In [71]: from math import sqrt
         def dist(p1, p2):
             return sqrt((p2.x-p1.x)**2 + (p2.y-p1.y)**2)
```

```
In [72]: dist(a, b)

Out[72]: 8.602325267042627

In [73]: import math

In [74]: math.fastsin = dist

In [75]: math.fastsin(a, b)

Out[75]: 8.602325267042627

In [76]: math.sin = dist

In [77]: math.sin(a, b)

Out[77]: 8.602325267042627

In [78]: a.__dict__

Out[78]: {'x': 3, 'y': 4, 'color': 'red'}

In [79]: a.name = 'Sunnyvale'

In [80]: a.__dict__

Out[80]: {'x': 3, 'color': 'red', 'name': 'Sunnyvale', 'y': 4}

In [81]: a.__dict__['size'] = 'medium' # add 'size' key to dict

In [82]: a.__dict__

Out[82]: {'x': 3, 'color': 'red', 'name': 'Sunnyvale', 'y': 4, 'size': 'medium'}

In [83]: a.size

Out[83]: 'medium'

In [84]: a.size = 'large'

In [85]: a.__dict__

Out[85]: {'x': 3, 'color': 'red', 'name': 'Sunnyvale', 'y': 4, 'size': 'large'}

In [86]: del a.__dict__['name']

In [87]: a.__dict__

Out[87]: {'x': 3, 'color': 'red', 'y': 4, 'size': 'large'}

In [88]: del a.size

In [89]: a.__dict__

Out[89]: {'x': 3, 'color': 'red', 'y': 4}

In [90]: b.__dict__

Out[90]: {'x': 10, 'y': 9}

In [91]: a.__class__
```

```
Out[91]: __main__.Point

In [92]: b.__class__

Out[92]: __main__.Point

In [93]: a.__class__ is Point

Out[93]: True

In [94]: id(a.__class__)

Out[94]: 4324236984

In [95]: id(Point)

Out[95]: 4324236984

In [96]: Point.__name__

Out[96]: 'Point'

In [97]: import sys

In [98]: sys.getsizeof(a)

Out[98]: 56

In [99]: sys.getsizeof(a.__dict__)

Out[99]: 480

In [106]: for k in a.__dict__.keys():
              print(k, '\t', sys.getsizeof(k), '\t', sys.getsizeof(a.__dict__[k]))

x           50          28
color           54          52
y           50          28

In [107]: def setup_point(p, x, y):
              ' add an x and y attribute to the object p '
              p.x = x
              p.y = y

In [108]: c = Point()

In [109]: setup_point(c, 6, 7)

In [110]: c.x

Out[110]: 6

In [111]: c.y

Out[111]: 7

In [112]: dist(a, c)

Out[112]: 4.242640687119285
```

```
In [113]: dist(b, c)

Out[113]: 4.47213595499958

In [114]: class Point:
              ' 2d point with x and y attributes '
              from math import sqrt
              def dist(p1, p2):
                  return sqrt((p2.x-p1.x)**2 + (p2.y-p1.y)**2)
              def setup_point(p, x, y):
                  ' add an x and y attribute to the object p '
                  p.x = x
                  p.y = y


In [115]: a = Point()
          b = Point()

In [116]: Point.setup_point(a, 3, 4)
          Point.setup_point(b, 10, 12)

In [117]: a.x, a.y # tuple-packing: create a 2-tuple from referenced objects
                   # NOTE: no round brackets needed

Out[117]: (3, 4)

In [118]: b.x, b.y

Out[118]: (10, 12)

In [119]: Point.dist(a, b)

Out[119]: 10.63014581273465

In [121]: class Point:
              ' 2d point with x and y attributes '
              from math import sqrt

              def setup_point(p, x, y):
                  ' add an x and y attribute to the object p '
                  p.x = x
                  p.y = y

              def distorigin(p):
                  return sqrt(p.x**2 + p.y**2)

              def dist(p1, p2):
                  return sqrt((p2.x-p1.x)**2 + (p2.y-p1.y)**2)


In [122]: d = Point()

In [123]: Point.setup_point(d, 6, 7)

In [125]: Point.distorigin(d)

Out[125]: 9.219544457292887
```

```
In [126]: from functools import partial

In [127]: from random import choice, randint

In [130]: choice('yes no maybe'.split())

Out[130]: 'no'

In [133]: for i in range(10):
              print("Call me " + choice('yes no maybe'.split()))

Call me no
Call me no
Call me no
Call me no
Call me no
Call me yes
Call me no
Call me maybe
Call me maybe
Call me no

In [134]: def oracle():
              return choice('yes no maybe'.split())

In [135]: oracle()

Out[135]: 'no'

In [136]: oracle()

Out[136]: 'yes'

In [138]: oracle()

Out[138]: 'yes'

In [140]: poracle = partial(choice, 'yes no maybe'.split())

In [141]: poracle

Out[141]: functools.partial(<bound method Random.choice of <random.Random object at 0x1008a3218>>, ['yes

In [142]: poracle()

Out[142]: 'no'

In [143]: poracle()

Out[143]: 'yes'

In [144]: poracle()

Out[144]: 'yes'

In [145]: randint(1, 6)

Out[145]: 2
```

```
In [147]: for i in range(5):
              print(randint(1,6))

6
2
3
4
3

In [148]: roll = partial(randint, 1, 6)

In [149]: roll()

Out[149]: 6

In [150]: roll()

Out[150]: 4

In [151]: roll()

Out[151]: 4

In [152]: roll()

Out[152]: 5

In [153]: d = partial(randint, 1)

In [165]: for i in range(1, 5):
              print(d(20) + d(10))

11
8
11
5

In [166]: dist

Out[166]: <function __main__.dist>

In [167]: setup_point

Out[167]: <function __main__.setup_point>

In [168]: Point.dist

Out[168]: <function __main__.Point.dist>

In [169]: Point.setup_point

Out[169]: <function __main__.Point.setup_point>

In [175]: a = Point()
          Point.setup_point(a, 3, 4)

In [176]: isinstance(a, Point)

Out[176]: True
```

```
In [177]: dir(a)

Out[177]: ['__class__',
           '__delattr__',
           '__dict__',
           '__dir__',
           '__doc__',
           '__eq__',
           '__format__',
           '__ge__',
           '__getattribute__',
           '__gt__',
           '__hash__',
           '__init__',
           '__le__',
           '__lt__',
           '__module__',
           '__ne__',
           '__new__',
           '__reduce__',
           '__reduce_ex__',
           '__repr__',
           '__setattr__',
           '__sizeof__',
           '__str__',
           '__subclasshook__',
           '__weakref__',
           'dist',
           'distorigin',
           'setup_point',
           'sqrt',
           'x',
           'y']

In [178]: a.__dict__

Out[178]: {'x': 3, 'y': 4}

In [180]: Point.__dict__.keys()

Out[180]: dict_keys(['sqrt', '__dict__', '__weakref__', 'distorigin', '__module__', '__doc__', 'dist', 'setu

In [181]: a.dist

Out[181]: <bound method Point.dist of <__main__.Point object at 0x106c1e668>>

In [182]: Point.dist

Out[182]: <function __main__.Point.dist>

In [183]: b.setup_point

Out[183]: <bound method Point.setup_point of <__main__.Point object at 0x106c17080>>

In [184]: b = Point()
          Point.setup_point(b, 12, 10)
```

```
In [187]: a.dist()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

    <ipython-input-187-a0d02c07396e> in <module>()
----> 1 a.dist()


    TypeError: dist() missing 1 required positional argument: 'p2'
```

```
In [188]: a.dist(b)

Out[188]: 10.816653826391969

In [189]: Point.dist(a, b)

Out[189]: 10.816653826391969

In [190]: part = partial(Point.dist, a)

In [191]: part(b)

Out[191]: 10.816653826391969

In [192]: c = Point()
          c.setup_point(5, 6) # c is automatically bound to the first parameter of
                              # setup_point as a partial closure

In [193]: c.distorigin()

Out[193]: 7.810249675906654

In [194]: # We want to be able to just do:
          d = Point(12, 15)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

    <ipython-input-194-23ac97cbc411> in <module>()
      1 # We want to be able to just do:
----> 2 d = Point(12, 15)


    TypeError: object() takes no parameters
```

```
In [195]: # one change: setup_point -> __init__
          # NOTE: this is the initializer, NOT the constructor
          class Point:
              ' 2d point with x and y attributes '
              from math import sqrt

              def __init__(p, x, y): # was setup_point before
```

```
        ' add an x and y attribute to the object p '
        p.x = x
        p.y = y

    def distorigin(p):
        return sqrt(p.x**2 + p.y**2)

    def dist(p1, p2):
        return sqrt((p2.x-p1.x)**2 + (p2.y-p1.y)**2)
```

```
In [196]: d = Point(12, 15) # this will automatically call __init__
```

```
In [197]: d.__dict__
```

```
Out[197]: {'x': 12, 'y': 15}
```

```
In [198]: # one change: first method parameter -> "self"
          # this is just a convention, but it is a VERY STRONG convention
          # that you should ALWAYS use.
          class Point:
              ' 2d point with x and y attributes '
              from math import sqrt

              def __init__(self, x, y): # was setup_point before
                  ' add an x and y attribute to the object p '
                  self.x = x
                  self.y = y

              def distorigin(self):
                  return sqrt(self.x**2 + self.y**2)

              def dist(self, other):
                  return sqrt((other.x-self.x)**2 + (other.y-self.y)**2)
```

```
In [207]: # other dunder methods: __len__
          # one change: first method parameter -> "self"
          # this is just a convention, but it is a VERY STRONG convention
          # that you should ALWAYS use.
          class Point:
              ' 2d point with x and y attributes '
              from math import sqrt

              def __init__(self, x, y): # was setup_point before
                  ' add an x and y attribute to the object p '
                  self.x = x
                  self.y = y

              def distorigin(self):
                  return sqrt(self.x**2 + self.y**2)

              def dist(self, other):
                  return sqrt((other.x-self.x)**2 + (other.y-self.y)**2)
```

```
            def __getitem__(self, index): # called by [ ]
                if index == 0:
                    return self.x
                elif index == 1:
                    return self.y
                else:
                    raise ValueError("Only supports index 0 and 1")
```

```
In [208]: a = Point(3, 5)

In [209]: a.distorigin()

Out[209]: 5.830951894845301

In [210]: a.x

Out[210]: 3

In [211]: a.x = 10

In [212]: a.distorigin()

Out[212]: 11.180339887498949

In [213]: a[0]

Out[213]: 10

In [214]: a[1]

Out[214]: 5

In [215]: a.x

Out[215]: 10

In [216]: a.y

Out[216]: 5

In [217]: a[3]
```

```
    ---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

    <ipython-input-217-94e7916e7615> in <module>()
----> 1 a[3]


    <ipython-input-207-222512266527> in __getitem__(self, index)
     24             return self.y
     25         else:
---> 26             raise ValueError("Only supports index 0 and 1")
     27


    ValueError: Only supports index 0 and 1
```

```
In [218]: a[-1]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-218-9f1255355f85> in <module>()
----> 1 a[-1]


<ipython-input-207-222512266527> in __getitem__(self, index)
 24             return self.y
 25         else:
---> 26             raise ValueError("Only supports index 0 and 1")
 27


ValueError: Only supports index 0 and 1
```

```
In [219]: a
```

```
Out[219]: <__main__.Point at 0x106c12c50>
```

```
In [220]: b = Point(3, 4)
```

```
In [221]: b
```

```
Out[221]: <__main__.Point at 0x106c34320>
```

```
In [242]: # other dunder methods: __len__
          # one change: first method parameter -> "self"
          # this is just a convention, but it is a VERY STRONG convention
          # that you should ALWAYS use.
          class Point:
              ' 2d point with x and y attributes '
              from math import sqrt

              def __init__(self, x, y): # was setup_point before
                  ' add an x and y attribute to the object p '
                  self.x = x
                  self.y = y

              def distorigin(self):
                  return sqrt(self.x**2 + self.y**2)

              def dist(self, other):
                  return sqrt((other.x-self.x)**2 + (other.y-self.y)**2)

              def __getitem__(self, index): # called by [ ]
                  if index == 0:
                      return self.x
                  elif index == 1:
                      return self.y
                  elif index%7 == 0:
                      return "BINGO! " + str(index)
```

```
                else:
                    raise ValueError("Only supports index 0 and 1")

            def __repr__(self):
                return 'Point({x}, {y})'.format(x=self.x, y=self.y)

            def __str__(self):
                return 'A Point object at ({x}, {y}), {d} from the origin'.format(
                        x=self.x, y=self.y, d=self.distorigin())
```

```
In [243]: a = Point(3, 5)
          b = Point(10, 20)

In [230]: a

Out[230]: Point(3, 5)

In [231]: b

Out[231]: Point(10, 20)

In [232]: points = [a, b]

In [233]: points

Out[233]: [Point(3, 5), Point(10, 20)]

In [234]: for p in points:
              print(p)

A Point object at (3, 5), 5.830951894845301 from the origin
A Point object at (10, 20), 22.360679774997898 from the origin

In [235]: for p in points:
              print(repr(p))

Point(3, 5)
Point(10, 20)

In [237]: a[0]

Out[237]: 3

In [238]: a[1]

Out[238]: 5

In [239]: a

Out[239]: Point(3, 5)

In [240]: a[3]


    ---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

    <ipython-input-240-94e7916e7615> in <module>()
----> 1 a[3]
```

```
    <ipython-input-228-816043b84e8c> in __getitem__(self, index)
     24             return self.y
     25         else:
---> 26             raise ValueError("Only supports index 0 and 1")
     27
     28     def __repr__(self):


    ValueError: Only supports index 0 and 1
```

```
In [244]: a[21]
```

```
Out[244]: 'BINGO! 21'
```

```
In [245]: a[49]
```

```
Out[245]: 'BINGO! 49'
```

```
In [246]: a[50]
```

```
    ---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

    <ipython-input-246-6b6d1322ced0> in <module>()
----> 1 a[50]


    <ipython-input-242-b2749a1e1d8b> in __getitem__(self, index)
     26             return "BINGO! " + str(index)
     27         else:
---> 28             raise ValueError("Only supports index 0 and 1")
     29
     30     def __repr__(self):


    ValueError: Only supports index 0 and 1
```

```
In [247]: a[48] # this is just a syntactic shortcut to a.__getitem__(48)
```

```
    ---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

    <ipython-input-247-79ef440423d2> in <module>()
----> 1 a[48]


    <ipython-input-242-b2749a1e1d8b> in __getitem__(self, index)
     26             return "BINGO! " + str(index)
     27         else:
---> 28             raise ValueError("Only supports index 0 and 1")
```

```
 29
 30     def __repr__(self):


ValueError: Only supports index 0 and 1
```

In []: