

Data types

```
None
3, -4, 1.01, -2.0
True, False
'Hello, world!\n'
[1, 2.0, 'hi']
{'hi': 3, 'bye': 100}
```

Basic operators

```
True and False, True or False, not True
1 + 3, 1 - 3, 1 * 3
5 / 2 == 2.5, 5 // 2 == 2, 5 % 2 == 1
'hi' + 'bye' # 'hibye'
[1, 2, 3] + [4, 5, 6] # [1, 2, 3, 4, 5, 6]
```

List methods

```
lst = [1, 2, 3]
len(lst) # 3
lst[0] # 1
lst[0:2] # [1, 2]
lst[0] = 'howdy' # lst == ['howdy', 2, 3]
lst.append(29) # lst == ['howdy', 2, 3, 29]
lst.pop() # lst == ['howdy', 2, 3], returns 29
lst.pop(1) # lst == ['howdy', 3], returns 2
lst.insert(1, 100) # lst == ['howdy', 100, 3]
3 in lst # returns True
```

Dictionary methods

```
d = {'hi': 4, 'bye': 100}
d['hi'] # 'hi'
d[100] # raises KeyError!
'hi' in d # True
4 in d # False
d['howdy'] = 15 # adds new key-value pair
d['hi'] = -100 # changes a key-value pair
```

Control flow

```
if x == 5:
    y = 1
elif:
    z = 2
else:
    y = 100

for i in [0, 1, 2, 3]: # or, "for i in range(4):"
    print(i)

j = 0
while j < 10:
    print(j)
    j = j * 2
```

Class syntax

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def size(self):
        return (self.x ** 2 + self.y ** 2) ** 0.5

p = Point(3, 4) # constructor
p.x # attribute access: returns 3
p.size() # method call: returns 5.0

class MyWeirdClass(Point):
    pass
```

Exceptions

```
raise KeyError
try:
    lst[1000]
except IndexError:
    print('haha')
```

Stack and Queue ADTs

```
my_stack = Stack()
my_stack.is_empty()
my_stack.push(10)
my_stack.pop()

my_queue = Queue()
my_queue.is_empty()
my_queue.enqueue(10)
my_queue.dequeue()
```

Linked lists (iterative)

```
class Node:
    def __init__(self, item):
        self.item = item
        self.next = None

class LinkedList:
    def __init__(self, items):
        if len(items) == 0:
            self.first = None
        else:
            self.first = Node(items[0])
            curr = self.first
            for item in items[1:]:
                curr.next = Node(item)
                curr = curr.next
```