

copybytes.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # copybytes.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9:
10: MAX_BYTE_COUNT = 512 # Arbitrary
11:
12: def main():
13:
14:     if len(sys.argv) != 3:
15:         print('usage: python %s infile outfile' % sys.argv[0],
16:               file=sys.stderr)
17:         sys.exit(1)
18:
19:     in_name = sys.argv[1]
20:     out_name = sys.argv[2]
21:
22:     try:
23:         in_file = open(in_name, mode='rb')
24:         out_file = open(out_name, mode='wb')
25:
26:         byte_array = in_file.read(MAX_BYTE_COUNT)
27:         while byte_array != b'':
28:             out_file.write(byte_array)
29:             byte_array = in_file.read(MAX_BYTE_COUNT)
30:
31:         out_file.close()
32:         in_file.close()
33:
34:     except Exception as ex:
35:         print(ex, file=sys.stderr)
36:         sys.exit(1)
37:
38: if __name__ == '__main__':
39:     main()

```

copystrings.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # copystrings.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9:
10: def main():
11:
12:     if len(sys.argv) != 3:
13:         print('usage: python %s infile outfile' % sys.argv[0],
14:               file=sys.stderr)
15:         sys.exit(1)
16:
17:     in_name = sys.argv[1]
18:     out_name = sys.argv[2]
19:
20:     try:
21:         in_file = open(in_name, mode='r', encoding='utf-8')
22:         out_file = open(out_name, mode='w', encoding='utf-8')
23:
24:         for line in in_file:
25:             out_file.write(line)
26:
27:         out_file.close()
28:         in_file.close()
29:
30:     except Exception as ex:
31:         print(ex, file=sys.stderr)
32:         sys.exit(1)
33:
34: if __name__ == '__main__':
35:     main()

```

copystringsfinally.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # copystringsfinally.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9:
10: def main():
11:
12:     if len(sys.argv) != 3:
13:         print('usage: python %s infile outfile' % sys.argv[0],
14:               file=sys.stderr)
15:         sys.exit(1)
16:
17:     in_name = sys.argv[1]
18:     out_name = sys.argv[2]
19:
20:     try:
21:         in_file = open(in_name, mode='r', encoding='utf-8')
22:         try:
23:             out_file = open(out_name, mode='w', encoding='utf-8')
24:             try:
25:                 for line in in_file:
26:                     out_file.write(line)
27:             finally:
28:                 out_file.close()
29:         finally:
30:             in_file.close()
31:
32:     except Exception as ex:
33:         print(ex, file=sys.stderr)
34:         sys.exit(1)
35:
36: if __name__ == '__main__':
37:     main()

```

copystringswith.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # copystringswith.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9:
10: def main():
11:
12:     if len(sys.argv) != 3:
13:         print('usage: python %s infile outfile' % sys.argv[0],
14:               file=sys.stderr)
15:         sys.exit(1)
16:
17:     in_name = sys.argv[1]
18:     out_name = sys.argv[2]
19:
20:     try:
21:         with open(in_name, mode='r', encoding='utf-8') as in_file:
22:             with open(out_name, mode='w', encoding='utf-8') as out_file:
23:                 for line in in_file:
24:                     out_file.write(line)
25:
26:     except Exception as ex:
27:         print(ex, file=sys.stderr)
28:         sys.exit(1)
29:
30: if __name__ == '__main__':
31:     main()

```

linesort.py (Page 1 of 1)

```
1: #!/usr/bin/env python
2:
3: #-----
4: # linesort.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9:
10: #-----
11:
12: def main():
13:
14:     lines = []
15:     for line in sys.stdin:
16:         line = line.rstrip('\n')
17:         lines.append(line) # or: lines += [line]
18:     lines.sort()
19:     for line in lines:
20:         print(line)
21:
22: #-----
23:
24: if __name__ == '__main__':
25:     main()
```

blank (Page 1 of 1)

```
1: This page is intentionally blank.
```

linesorttim.py (Page 1 of 2)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # linesorttim.py
5: # Author:
6: # https://www.codespeedy.com/timsort-algorithm-implementation-in-python/
7: #-----
8:
9: import sys
10:
11: minrun = 32
12:
13: def InsSort(arr, start, end):
14:     for i in range(start+1, end+1):
15:         elem = arr[i]
16:         j = i-1
17:         while j>=start and elem<arr[j]:
18:             arr[j+1] = arr[j]
19:             j -= 1
20:         arr[j+1] = elem
21:     return arr
22:
23: def merge(arr, start, mid, end):
24:     if mid==end:
25:         return arr
26:     first = arr[start:mid+1]
27:     last = arr[mid+1:end+1]
28:     len1 = mid-start+1
29:     len2 = end-mid
30:     ind1 = 0
31:     ind2 = 0
32:     ind = start
33:
34:     while ind1<len1 and ind2<len2:
35:         if first[ind1]<last[ind2]:
36:             arr[ind] = first[ind1]
37:             ind1 += 1
38:         else:
39:             arr[ind] = last[ind2]
40:             ind2 += 1
41:         ind += 1
42:
43:     while ind1<len1:
44:         arr[ind] = first[ind1]
45:         ind1 += 1
46:         ind += 1
47:
48:     while ind2<len2:
49:         arr[ind] = last[ind2]
50:         ind2 += 1
51:         ind += 1
52:
53:     return arr
54:
55: def TimSort(arr):
56:     n = len(arr)
57:
58:     for start in range(0, n, minrun):
59:         end = min(start+minrun-1, n-1)
60:         arr = InsSort(arr, start, end)
61:
62:     curr_size = minrun
63:     while curr_size<n:
64:         for start in range(0, n, curr_size*2):
65:             mid = min(n-1, start+curr_size-1)

```

linesorttim.py (Page 2 of 2)

```

66:             end = min(n-1, mid+curr_size)
67:             arr = merge(arr, start, mid, end)
68:             curr_size *= 2
69:         return arr
70:
71: #-----
72:
73: def main():
74:     lines = []
75:     for line in sys.stdin:
76:         line = line.rstrip('\n')
77:         lines.append(line) # or: lines += [line]
78:     TimSort(lines)
79:     for line in lines:
80:         print(line)
81:
82: #-----
83:
84: if __name__ == '__main__':
85:     main()

```

concord.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # concord.py
5: # Author: Bob Dondero
6: #-----
7:
8: import sys
9: import re
10:
11: #-----
12:
13: def process_line(line, concordance):
14:
15:     line = line.lower()
16:
17:     re_letters = re.compile(r'[a-z]+')
18:     words = re_letters.findall(line)
19:
20:     for word in words:
21:         if word in concordance:
22:             concordance[word] += 1
23:         else:
24:             concordance[word] = 1
25:
26: #-----
27:
28: def main():
29:
30:     concordance = {}
31:
32:     for line in sys.stdin:
33:         process_line(line, concordance)
34:
35:     #for word in concordance:
36:     #    print('%s: %d' % (word, concordance[word]))
37:     for word, count in concordance.items():
38:         print('%s: %d' % (word, count))
39:
40: #-----
41:
42: if __name__ == '__main__':
43:     main()

```

variadic.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # variadic.py
5: # Author: Bob Dondero
6: #-----
7:
8: def my_func(i, j, *args, **kwargs):
9:     print(i)
10:    print(j)
11:    for arg in args:
12:        print(arg)
13:    for key, value in kwargs.items():
14:        print(key, value)
15:
16: def main():
17:     my_func('a', 'b', 'c', 'd', key1='e', key2='f')
18:
19: if __name__ == '__main__':
20:     main()

```

euclid.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # euclid.py
5: # Author: Bob Dondero
6: #-----
7:
8: def gcd(i, j):
9:
10:     if (i == 0) and (j == 0):
11:         raise ZeroDivisionError(
12:             'gcd(i,j) is undefined if i and j are 0')
13:     i = abs(i)
14:     j = abs(j)
15:     while j != 0: # Euclid's algorithm
16:         i, j = j, i%j
17:     return i
18:
19: #-----
20:
21: def lcm(i, j):
22:
23:     if (i == 0) or (j == 0):
24:         raise ZeroDivisionError(
25:             'lcm(i,j) is undefined if i or j is 0')
26:     i = abs(i)
27:     j = abs(j)
28:     return (i // gcd(i, j)) * j

```

euclidstrong.py (Page 1 of 1)

```

1: #!/usr/bin/env python
2:
3: #-----
4: # euclidstrong.py
5: # Author: Bob Dondero
6: #-----
7:
8: def gcd(i, j):
9:
10:     if not isinstance(i, int) or not isinstance(j, int):
11:         raise TypeError('gcd() arguments must be integers')
12:
13:     if (i == 0) and (j == 0):
14:         raise ZeroDivisionError(
15:             'gcd(i,j) is undefined if i and j are 0')
16:
17:     i = abs(i)
18:     j = abs(j)
19:     while j != 0: # Euclid's algorithm
20:         i, j = j, i%j
21:     return i
22:
23: #-----
24:
25: def lcm(i, j):
26:
27:     if not isinstance(i, int) or not isinstance(j, int):
28:         raise TypeError('lcm() arguments must be integers')
29:
30:     if (i == 0) or (j == 0):
31:         raise ZeroDivisionError(
32:             'lcm(i,j) is undefined if i or j is 0')
33:
34:     i = abs(i)
35:     j = abs(j)
36:     return (i // gcd(i, j)) * j

```