Exercises

Week 41

DM536 Introduction to Programming
DM562 Scientific Programming
DM857 Introduction to Programming
DS830 Introduction to Programming

1 Handling Errors

1. For each of the following types of errors, write a (buggy) python program that would result in them without using `raise`.

   (a) `ValueError` Raised when an operation or function receives an argument that has the right type but an inappropriate value, and the situation is not described by a more precise exception.

   (b) `TypeError` Raised when an operation or function is applied to an object of inappropriate type. The associated value is a string giving details about the type mismatch.

   (c) `SyntaxError` Raised when the parser encounters a syntax error.

   (d) `NameError` Raised when a local or global name is not found. This applies only to unqualified names.

   (e) `AttributeError` Raised when an attribute (e.g., of an object) reference or assignment fails.

   (f) `AssertionError` Raised when an assert statement fails.

2. For each of the following programs, find all the possible sources of errors. Discuss which should or can be addressed using preconditions, `try`/`except`, or other changes to the program.

   (a)
   ```
   a = input('Enter a value for a:')
   b = input('Enter a value for b:')
   c = a * int(b)
   ```

   (b)
   ```
   a = input('Enter a value for a:')
   b = input('Enter a value for b:')
   c = a / int(b)
   ```

   (c)
   ```
   x = 5
   str = 'x is '
   print(str + x)
   ```

   (d)
   ```
   def quota(jobs,workers):
       return jobs / workers
   ```

   (e)
   ```
   def fahrenheit_to_celsius(degrees):
       return degrees * conversion_factor
   ```

   (f)
   ```
   def get_int(message):
       s = input(message)
       i = int(s)
       return i
   ```

3. For each of the following programs, show its output and reconstruct how errors propagate.

   (a)
   ```
   try:
     x = int(input('x = '))
   except ValueError:
     x = 0
   finally:
     print(x)
   ```

   (b)
   ```
   try:
     x = 1 / int(input('x = '))
   except ValueError:
     x = 1
   ```

```
except ZeroDivisionError:
  x = 0
finally:
  print(x)
```

(c)
```
x = 0
try:
  x = 1 / int(input('x = '))
except ValueError:
  x = 1 / x
except ZeroDivisionError:
  x = 2
finally:
  print(x)
```

(d)
```
try:
  x = 0
  y = 1 / x
  print(y)
except ZeroDivisionError:
  x = 1
finally:
  print(x)
  print(y)
```

(e)
```
x = 1
try:
  try:
    y = int(input('y = '))
  except ValueError:
    x = 0
  finally:
    print(1 / x)
except ZeroDivisionError:
  x = 2
finally:
  print(x)
```

(f)
```
try:
  try:
    x = int(input('x = ')) / 0
  except ValueError:
    x = 0
  finally:
    print(1 / x)
except ZeroDivisionError:
  print(x)
```

2 Lists

1. For each of the following programs, compute its output.

(a)
```
xs = [0,1,2,3,4,5]
print(xs[0],xs[2],x[-1])
```

(b)
```
xs = [0,1,2,3,4,5]
print(xs[1:2])
print(xs[1:3])
print(xs[:3])
print(xs[3:])
print(xs[1:4:2])
print(xs[1:5:2])
print(xs[1:15:2])
print(xs[15:1:2])
```

(c)
```
xs = [[0,1],[2,3,4],[],[5]]
print(xs[1])
print(xs[1][1])
print(xs[:3][1])
print(xs[1][1:])
```

(d)
```
xs = [0,1,2,3,4,5]
print(xs[len(xs)])
```

2. For each of the following programs, compute its output.

(a)
```
xs = [0,1,2,3,4,5]
xs[0] = xs[2]
print(xs[0],xs[2])
```

(b)
```
xs = [0,1,2,3,4,5]
xs[0],xs[2] = xs[2],xs[0]
print(xs[0],xs[2])
```

(c)
```
xs = [0,1,2,3,4,5]
del xs[2]
print(xs)
```

(d)
```
xs = [0,1,2,3,4,5]
xs.append(7)
print(xs)
```

(e)
```
xs = [0,1,2,3,4,5]
print(xs.pop())
print(xs)
```

3. Write a function `get_or_default(xs,i,default)` that returns the `i`-th element of `xs` or `default` if there is no such element.

4. Write a function `get_cyclic(xs,i)` that returns the `i`-th element of `xs` where indexes are considered as if the list was circular (when you reach one end you start from the other).

5. Write a function `get_in_even(xs)` that given a list `xs` returns a list containing only the elements of `xs` in even positions.

6. Write a function `get_in_odd(xs)` that given a list `xs` returns a list containing only the elements of `xs` in odd positions.

7. Write a function `swap(xs,i,j)` exchanges the items in positions `i` and `j` of `xs`.

8. For each of the following programs, compute its output.

   (a)
   ```
   xs = [0,1,2,3,4,5]
   ys = [ x + 1 for x in xs]
   print(ys)
   ```
   (b)
   ```
   xs = [0,1,2,3,4,5]
   ys = [ x for x in xs if x % 2 == 0]
   print(ys)
   ```
   (c)
   ```
   xs = [[0,1],[2,3],[4,5]]
   ys = [ y for x in xs for y in x]
   print(ys)
   ```

9. For each of the following programs, compute its output.

   (a)
   ```
   xs = ['1','2','3']
   ys = [len(x) for x in xs]
   print(ys)
   ```
   (b)
   ```
   xs = ['1','2','3']
   ys = '-'.join(xs)
   print(ys)
   ```
   (c)
   ```
   xs = '123'
   ys = '-'.join(xs)
   print(ys)
   ```
   (d)
   ```
   xs = [1,2,3]
   ys = [ str(x) for x in xs]
   zs = '-'.join(zs)
   print(zs)
   ```

10. Write a function `only_even(xs)` that given a list `xs` of numbers returns a copy of the list containing only the even numbers of `xs`.

11. Write a function `replace(xs,a,b)` that given a list `xs` returns a new list with all elements of `xs` except for those equal to `a` which are instead replaced with `b`. For instance, `replace([0,1,0,1],0,'Z')` returns `['Z',1,'Z',1]`.

12. Write a function `join(xs)` that given a list of lists returns a list obtained concatenating all its elements. For instance, `join([[1],[2,3],[4]])` returns `[1,2,3,4]`.

13. Write a function `singletons(xs)` that given a list returns the list of singletons for its elements (a singleton is a 1-element list). For instance, `singletons([1,2,3,4])` returns `[[1],[2],[3],[4]])`.