4478 INTRODUCTION TO INFORMATION TECHNOLOGY
8936 INTRODUCTION TO INFORMATION TECHNOLOGY G

Revision – Week 13

UNIVERSITY OF CANBERRA

About the Final Assignment

To Consider:

This Assignment opens May 13th at 1:00 PM (AEST).
It closes May 15th at 1:00 PM (AEST)

- Two questions.
- First question on software testing.
- Second question on exception handling.
- Submissions on Canvas as usual.

Submission:

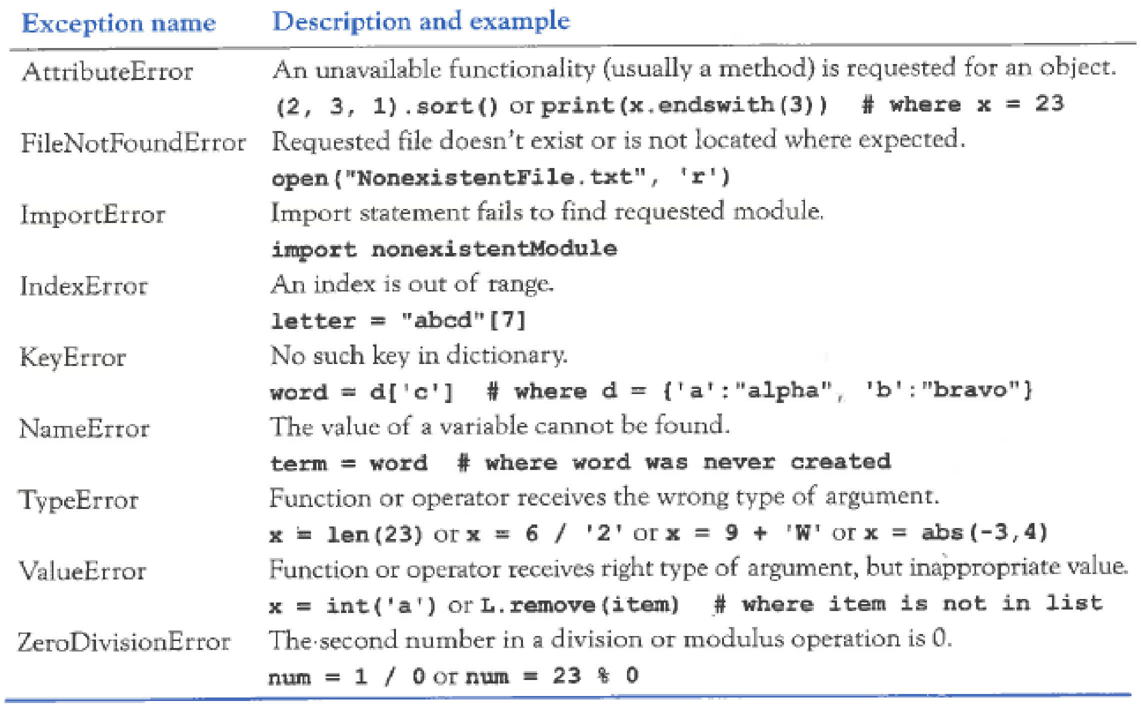
- Software Testing: HTML file.
- Exception Handling: .py file (please, zip it first)

Revision

This Week’s Revision Content

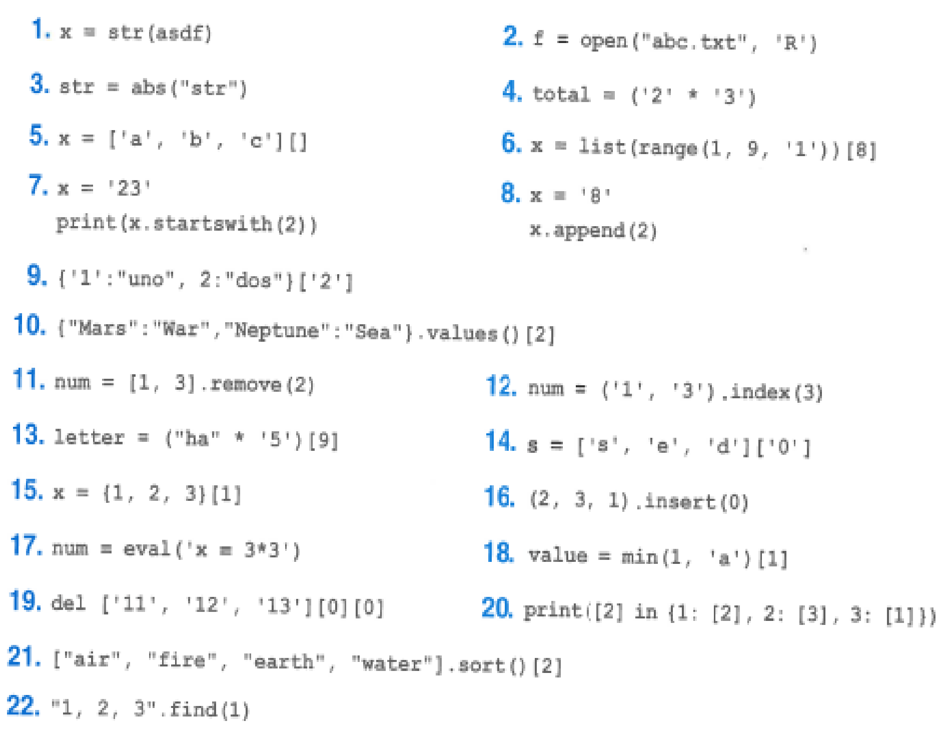
- Practice with software testing.
- Practice with exception handling.

Exception Handling

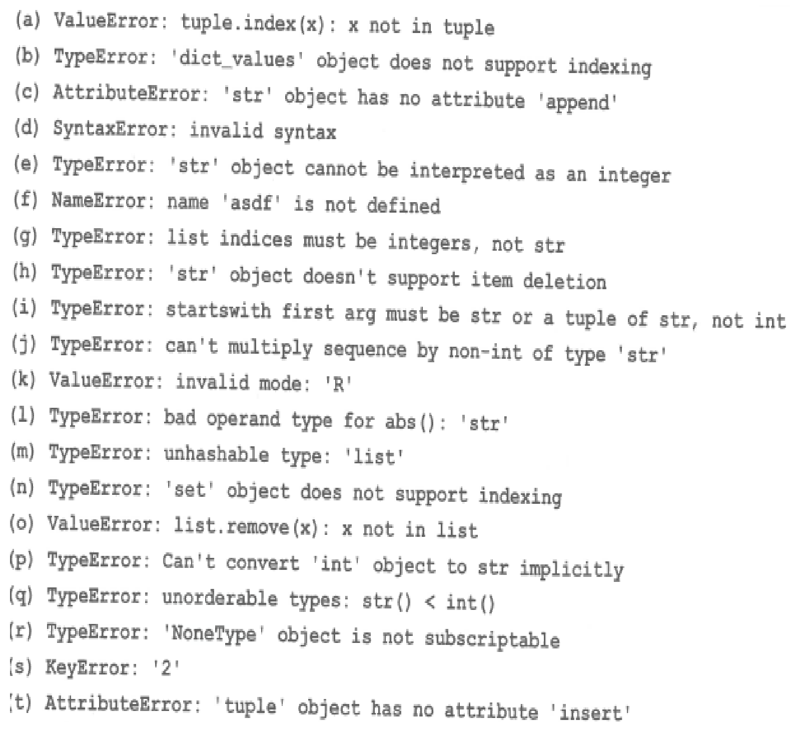


Exception name	Description and example
AttributeError	An unavailable functionality (usually a method) is requested for an object. `(2, 3, 1).sort()` or `print(x.endswith(3))  # where x = 23`
FileNotFoundError	Requested file doesn't exist or is not located where expected. `open("NonexistentFile.txt", 'r')`
ImportError	Import statement fails to find requested module. `import nonexistentModule`
IndexError	An index is out of range. `letter = "abcd"[7]`
KeyError	No such key in dictionary. `word = d['c']  # where d = {'a':"alpha", 'b':"bravo"}`
NameError	The value of a variable cannot be found. `term = word  # where word was never created`
TypeError	Function or operator receives the wrong type of argument. `x = len(23)` or `x = 6 / '2'` or `x = 9 + 'W'` or `x = abs(-3,4)`
ValueError	Function or operator receives right type of argument, but inappropriate value. `x = int('a')` or `L.remove(item)  # where item is not in list`
ZeroDivisionError	The second number in a division or modulus operation is 0. `num = 1 / 0` or `num = 23 % 0`

Exception Handling



```
1. x = str(asdf)                          2. f = open("abc.txt", 'R')

3. str = abs("str")                       4. total = ('2' * '3')

5. x = ['a', 'b', 'c'][]                  6. x = list(range(1, 9, '1'))[8]

7. x = '23'                               8. x = '8'
   print(x.startswith(2))                    x.append(2)

9. {'1':"uno", 2:"dos"}['2']

10. {"Mars":"War","Neptune":"Sea"}.values()[2]

11. num = [1, 3].remove(2)                12. num = ('1', '3').index(3)

13. letter = ("ha" * '5')[9]              14. s = ['s', 'e', 'd']['0']

15. x = {1, 2, 3}[1]                      16. (2, 3, 1).insert(0)

17. num = eval('x = 3*3')                 18. value = min(1, 'a')[1]

19. del ['11', '12', '13'][0][0]          20. print([2] in {1: [2], 2: [3], 3: [1]})

21. ["air", "fire", "earth", "water"].sort()[2]

22. "1, 2, 3".find(1)
```



```
(a) ValueError: tuple.index(x): x not in tuple
(b) TypeError: 'dict_values' object does not support indexing
(c) AttributeError: 'str' object has no attribute 'append'
(d) SyntaxError: invalid syntax
(e) TypeError: 'str' object cannot be interpreted as an integer
(f) NameError: name 'asdf' is not defined
(g) TypeError: list indices must be integers, not str
(h) TypeError: 'str' object doesn't support item deletion
(i) TypeError: startswith first arg must be str or a tuple of str, not int
(j) TypeError: can't multiply sequence by non-int of type 'str'
(k) ValueError: invalid mode: 'R'
(l) TypeError: bad operand type for abs(): 'str'
(m) TypeError: unhashable type: 'list'
(n) TypeError: 'set' object does not support indexing
(o) ValueError: list.remove(x): x not in list
(p) TypeError: Can't convert 'int' object to str implicitly
(q) TypeError: unorderable types: str() < int()
(r) TypeError: 'NoneType' object is not subscriptable
(s) KeyError: '2'
(t) AttributeError: 'tuple' object has no attribute 'insert'
```

The following program will perform properly if the user enters 0 in response to the request for input. However, the program will crash if the user responds with "eight". Rewrite the program using a **`try/except`** statement so that it will handle both types of responses. See Fig. 6.1.

```
while True:
    n = int(input("Enter a nonzero integer: "))
    if n != 0:
        reciprocal = 1 / n
        print("The reciprocal of {0} is {1:,.3f}".format(n, reciprocal))
        break
    else:
        print("You entered zero. Try again.")
```

```
Enter a nonzero integer: 0
You entered zero. Try again.
Enter a nonzero integer: eight
You did not enter a nonzero integer. Try again.
Enter a nonzero integer: 8
The reciprocal of 8 is 0.125
```

State Capitals Assume that the list *stateCapitals* contains the names of the 50 state capitals. Write a robust code segment that requests the name of a capital and removes it from the list. See Fig. 6.2.

```
Enter a state capital to delete: Chicago
Not a state capital.
Enter a state capital to delete: Springfield
Capital deleted.
```

Cabin Baggage

One piece of cabin baggage is allowed.

There is a limit of 100cm on the linear dimension (length + breadth + height).

The weight cannot be more than 10kg.

And if the linear dimension is more than 80cm, the weight cannot be more than 8kg.

The function *baggage OK()* determines whether the cabin baggage is acceptable or not. You are to test this function.

Notes:
1: There may be no errors to find.
2: Description and coverage are important.
3: Avoid numbers in your description
4: Do not write more than 12 tests. You will not need that many.

Use whole numbers for weight and linear dimension.
Use true | false for baggage OK().

There is an example line in the table. You will need to remove / edit this one.

CabinBaggage.CabinBaggageFixture			
weight	linear dimension	baggage OK()	Description
5	90	true	Light but large luggage

Competency

In order to be considered for a job, an applicant must perform satisfactorily in an aptitude test and a skills test. That is

either
  Competency 1: at least 80 on the aptitude test and at least 40 on the skills test
or
  Competency 2: at least 70 on the aptitude test and at least 60 on the skills test
or both.

The function *CompetencySatisfied()* determines whether someone will be employed or not. You are to test this function.

Notes:
1: There may be no errors to find.
2: Description and coverage are important.
3: Avoid numbers in your description
4: You will need no more than 12 tests.

Use integer numbers for aptitude and skills.
Use true | false for CompetencySatisfied.

There is an example line in the table. You will need to remove / edit this one.

Competency.CompetencyFixture			
aptitude	skills	CompetencySatisfied()	Description
70	60	true	min aptitude, min skills, competency 1

Parking

Parking calculates the charge for parking a car or motor bike.
For cars, the cost is $10 per half hour or part thereof, and for motor bikes the cost is $7 per hour or part thereof.

For example, if a car was parked for 35 minutes, the charge would be $20.

The functions *Chris(), Kim(), Pat()* provide three implementations of the above business rules and determine what the parking fee is (i.e. how much one has to pay).

Notes:
1: There may be no errors to find.
2: Description and coverage are important.
3: Avoid numbers in your description
4: Do not write more than 12 tests.

For this application you have to set values for *vehicle* and *time*, and expected values for *Chris, Kim, Pat*.
vehicle should be a "car" or "bike"
time should be a whole number >= 0.
Values in Chris(), Kim(), Pat() should be a whole number >= 0 in each column.

There is an example line in the table. You will need to remove / edit this one.

Parking.ParkingFixture					
vehicle	time	Chris()	Kim()	Pat()	description
car	35	20	20	20	car, more than 1/2 hour so charge for 1 hour