Chapter 11: Exceptions

Last Update: May 1, 2023

Program 1:

11 | 01-tuple.py

```
 1 def next():
 2     input("Next> ")
 3
 4 fruits = ("banana", "apple", "mango")
 5 print(f"Case 1: {fruits}, {type(fruits)}")
 6 next()
 7
 8 languages = 'Python', 'Java', 'C/C++'
 9 print(f"Case 2: {languages}, {type(languages)}")
10 next()
11
12 empty = ()
13 print(f"Case 3: {empty}, {type(empty)}")
14 next()
15
16 double = 'One', 'two'
17 print(f"Case 4: {double}, {type(double)}")
18 next()
19
20 double = ('One', 'two')
21 print(f"Case 5: {double}, {type(double)}")
22 next()
23
24 double = (('One', 'two'))
25 print(f"Case 6: {double}, {type(double)}")
26 next()
27
28 single = ('One')
29 print(f"Case 7: {single}, {type(single)}")
30 next()
31
32 single = tuple('One')
33 print(f"Case 8: {single}, {type(single)}")
34 next()
35
36 single = ('One',)
37 print(f"Case 9: {single}, {type(single)}")
```

Program 2:

11	05-tuples.py

```
 1 def next():
 2     input("Next> ")
 3
 4 print("Case 1: tuple of strings")
 5 fruits = ("banana", "apple", "mango")
 6 print(f"\tfruits = {fruits}, type = {type(fruits)}")
 7 next()
 8
 9 print("Case 2: casting a string into a tuple")
10 t = tuple('Python')
11 print(f"\tt = {t}, type = {type(t)}")
12 next()
13
14 print("Case 3: A tuple of one element")
15 t = 'Python',
16 print(f"\tt = {t}, type = {type(t)}")
17 next()
18
19 print("Case 4: A string, not a tuple")
20 t = 'Python'
21 print(f"\tt = {t}, type = {type(t)}")
22 next()
23
24 print("Case 5: A tuple of one number")
25 t = 5,
26 print(f"\tt = {t}, type = {type(t)}")
27 next()
28
29 print("Case 6: Use (but don't change) a tuple like a list")
30 for i in range(len(fruits)):
31     print(f"\t{fruits[i]}")
32 print()
33 for idx, item in enumerate(fruits, 1):
34     print(f"\t{idx}: {item}")
```

Program 3:

11 | 11-list and dict.py

```
 1 def next():
 2   input("\nNext> ")
 3
 4 mycar = {
 5   "brand": "Ford",
 6   "model": "Mustang",
 7   "year": 1964
 8 }
 9 mycar_list = ["Ford", "Mustang", 1964]
10
11 print('\n*** 1: List with index')
12 print(mycar_list[0]) # brand
13 print(mycar_list[1]) # model
14 print(mycar_list[2]) # year
15 next()
16
17 # This is better than the last one
18 print('\n*** 2: List with constants')
19 BRAND = 0
20 MODEL = 1
21 YEAR  = 2
22 print(mycar_list[BRAND])
23 print(mycar_list[MODEL])
24 print(mycar_list[YEAR])
25 next()
26
27 # This is a better way
28 print('*** 3: Dictionary')
29 print(mycar['brand'])
30 print(mycar['model'])
31 print(mycar['year'])
```

Program 4:

11	13-check key.py

```
def next():
  input("\nNext> ")

mycar = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print('Mycar:', mycar)
print(f'The length of the dictionary is {len(mycar)}')
next()

print(f"Adding a color:")
mycar['color'] = 'red'
print('Mycar:', mycar)
print(f'The length of the dictionary is {len(mycar)}')
next()

print(f"Adding/Replacing another color:")
mycar['color'] = 'yellow'
print('Mycar:', mycar)
print(f'The length of the dictionary is {len(mycar)}')
next()

mykey = input('Enter a key to search: ')
mycar.get(mykey, 'NOT here')
if mykey in mycar:
    print(f'{mykey} is {mycar[mykey]}')
else:
    print((f'{mykey} is not in {mycar.keys()}'))

# This may fail
print(f'{mykey} is {mycar[mykey]}')
```

Program 5:

11	dd-xxxx.py

```
 1 def next():
 2     input("Next> ")
 3 cs1336 = {'name': 'Python', 'courseNum': 1336, 'instructor': 'Huang'}
 4 cs1430 = {'name': 'C++', 'courseNum': 1430}  # no instructor
 5
 6 print(f"CS1336: {cs1336}")
 7 print(f"CS1430: {cs1430}")
 8 next()
 9
10 print('*** 1: Usig the key as an index to access the value:')
11 for key in cs1336:
12     print(cs1336[key])
13 next()
14
15 print('\n*** 2: Using get() function to access the value:')
16 for key in cs1336:
17     print(cs1336.get(key))
18 next()
19
20 print('\n*** 3: Using get(): Non-existing key value replaced with
21 None ')
22 for key in cs1336:
23     print(cs1430.get(key))
24 next()
25
26 print('\n*** 4: Using get(): Non-existing key value replaced with
27 "missing" ')
28 for key in cs1336:
29     print(cs1430.get(key, 'Missing')) # any message here
30 next()
31
32 print('\n*** 5: Non-existing key using index:')
33 for key in ['name', 'courseNum', 'instructor']:
34     print(cs1430[key])
```

Program 6:

11	30-iterator.py

```
 1 def next():
 2     input("Next> ")
 3
 4 mycar = {
 5   "brand": "Ford",
 6   "model": "Mustang",
 7   "year": 1964
 8 }
 9
10 print(f"\nIterate over keys:")
11 for key in mycar:
12     print(f'\t{key}')
13 next()
14
15 print(f"\nIterate over keys as index:")
16 for key in mycar:
17     print(f'\t{mycar[key]}')
18 next()
19
20 print(f"\nIterate over keys:")
21 for key in mycar.keys():
22     print(f"\t{key}")
23 next()
24
25 print("\nIterate over values:")
26 for value in mycar.values():
27     print(f'\t{value}')
28 next()
29
30 print("\nIterate over items:")
31 for key, value in mycar.items():
32     print(f'\t{key} -> {value}  ')
```

Program 7:

11	39-methods.py

```
 1 def next():
 2     input('Next> ')
 3
 4 stu = {
 5     101: "Huang, Stephen",
 6     102: "Johnson, Olin",
 7     190: "Smith, John",
 8     123: "Anderson, Robert"
 9 }
10 print("\t*** Case 1: copy()")
11 print(f"Original: {id(stu)}, {stu}")
12 stu2 = stu
13 print(f"Stu2:     {id(stu2)}, {stu2}")
14 newstu = stu.copy()
15 print(f"New:      {id(newstu)}, {newstu}")
16 next()
17
18 print("\t*** Case 2: clear()")
19 stu.clear()
20 print(f"Original: {id(stu)}, {stu}")
21 print(f"Original: {id(stu2)}, {stu2}")
22 stu = newstu.copy()
23 print(f"New:      {id(newstu)}, {newstu}")
24 next()
25
26 print("\t*** Case 3: keys, values, items")
27 print(stu.keys())
28 print(stu.values())
29 print(stu.items())
30 next()
31
32 print("\t*** Case 4: pop(101)")
33 print(stu)
34 stu.pop(101)
35 print(stu)
36 next()
37
38 print("\t*** Case 5: popitem(), last item")
39 print(stu)
40 stu.popitem()
   print(stu)
   next()

   print("\t*** Case 6: update({dict})")
   print(stu)
   stu.update({102: "Johnson, Robert"})
   print(stu)
   stu.update({102: 'Johnson, Olin', 120: 'Shah, Shishir'}) # like
   append()
   print(stu)
   next()

   print("\t*** Case 7: fromkeys({dict})")
```

```
keys = (101, 102, 103)
values = ('default value')  # multiple value?
dict4 = dict.fromkeys(keys, values)
print(dict4)
next()

print("\t*** Case 8: setdefault(key, value), set the value if no
key")
car = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}

print(car)
car.setdefault("year", 1977)
print(car)
car.setdefault("color", 'Red')
print(car)
```

Program 8:

```
11  44-histogram.py
 1  def histogram(str):
 2      d = dict()
 3      for ch in str:
 4          ch = ch.lower()
 5          d[ch] = d.get(ch, 0) + 1
 6      return d
 7
 8  def dictprint(d):
 9      print(f'\nKey Value')
10      print('--- -----')
11      for ch, count in sorted(d.items(),key=lambda s:s[1],
12  reverse=True):
13          print(f'\'{ch}\'  {count:3d} |{"X"*count}')
14      print('--- -----')
15
16  for str in ['brontosaurus', 'to be or not to be', 'mississippi']:
17      h = histogram(str)
18      dictprint(h)
19
20  h = histogram('Four score and seven years ago our fathers brought '+
21                'forth on this continent, a new nation, conceived in'+
22                ' Liberty, and dedicated to the proposition that all'+
23                ' men are created equal.')
24  dictprint(h)
```

Program 9:

```
 1 | 45-dict category.py
```

```
 1 def expense(log, cat, cost):
 2     log[cat] = log.get(cat,0)+cost
 3
 4 trip = {}
 5 expense(trip, 'uber', 39.00)
 6 expense(trip, 'food', 30.00)
 7 expense(trip, 'bar', 20.00)
 8 expense(trip, 'food', 15.50)
 9 expense(trip, 'hotel', 303.00)
10 expense(trip, 'uber', 34.00)
11 expense(trip, 'food', 31.00)
12 expense(trip, 'food', 32.00)
13 expense(trip, 'uber', 14.00)
14 expense(trip, 'food', 21.00)
15 expense(trip, 'food', 12.00)
16 print(trip)
```

Program 10:

```
11 | dd-xxxx.py
```

```
 1 price = {'Apple':2.50, 'Grapes':2.00, 'Bananas':1.75, 'pear':3.00}
 2 print(type(sorted(price)))
 3 print()
 4
 5 print(f"*** sorted(price):")
 6 for k in sorted(price):
 7     print(k)
 8 print()
 9
10 print(f"*** sorted(price.items()):")
11 for item in sorted(price.items()):
12     print(item)
13 print()
14
15 print(f"*** price.items() in reverse:")
16 for k, v in price.items():
17     print(v,k)
18 print()
19
20 print(f"*** price.items() in decreasing price:")
21 for k, v in sorted(price.items(), key=lambda x: x[1], reverse=True):
22     print(v, k)
23 print()
24
25 print(f"dict -> list")
26 list2 = [(v,k) for k, v in price.items()]
27 print(list2)
28 print(f"list -> dict")
29 dict2 = dict(list2)
30 print(dict2)
```

Program 11:

11	67-sort.py
1	`people = {3: "Alice", 2: "Jill", 4: "Bob", 1: "Ava"}`
2	`print(people)`
3	
4	`people1 = people.items()`
5	`people1 = dict(sorted(people1))`
6	`print(people1)`
7	
8	`people1 = dict(sorted(people.items(), key=lambda item: item[0]))`
9	`print(people1)`
10	`people2 = dict(sorted(people.items(), key=lambda item: item[1]))`
11	`print(people2)`