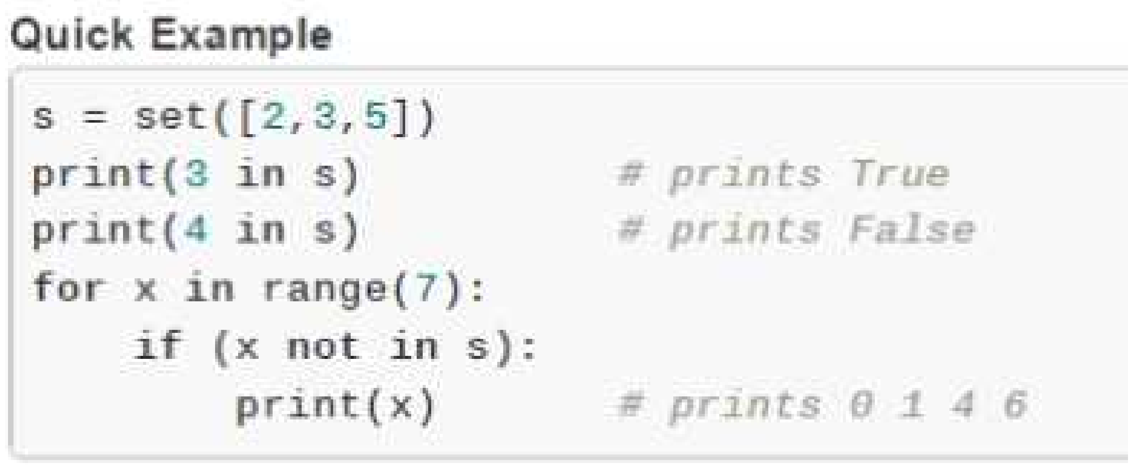
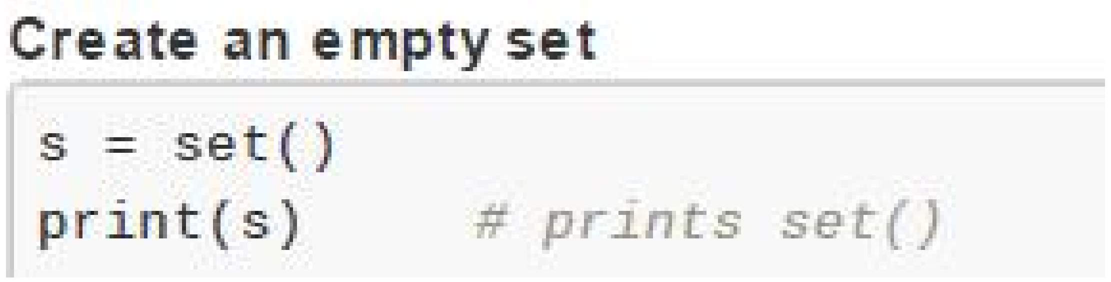
Set and Dictionary in Python

Set in Python



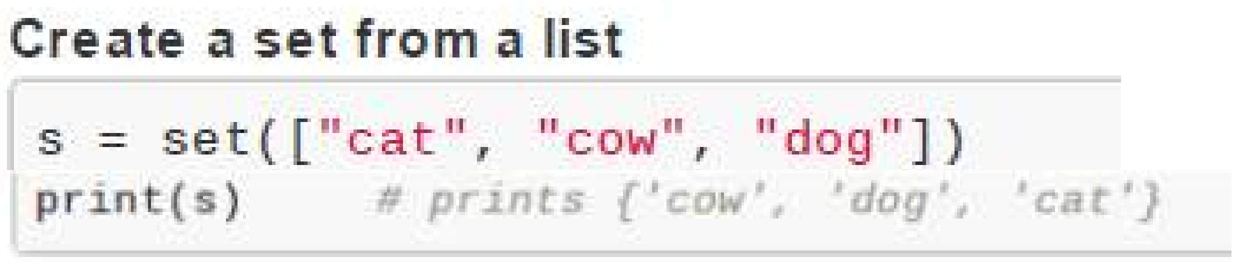
Quick Example

```
s = set([2,3,5])
print(3 in s)               # prints True
print(4 in s)               # prints False
for x in range(7):
    if (x not in s):
        print(x)            # prints 0 1 4 6
```



Create an empty set

```
s = set()
print(s)      # prints set()
```



Create a set from a list

```
s = set(["cat", "cow", "dog"])
print(s)     # prints {'cow', 'dog', 'cat'}
```

Create a set from any iterable object

```
s = set("wahoo")
print(s)      # surprised?
```

Create a statically-allocated set

```
s = { 2, 3, 5 }
print(s)     # prints { 2, 3, 5 }
```

Caution: { } is not an empty set!

```
s = { }
print(type(s) == set)  # False!
print(type(s))         # This is a dict (we'll learn about those soon)
```

Sets are Unordered

```
s = set([2,4,8])
print(s)            # prints {8, 2, 4} in standard Python
for element in s: # prints 8, 2, 4
    print(element)
```

Elements are Unique

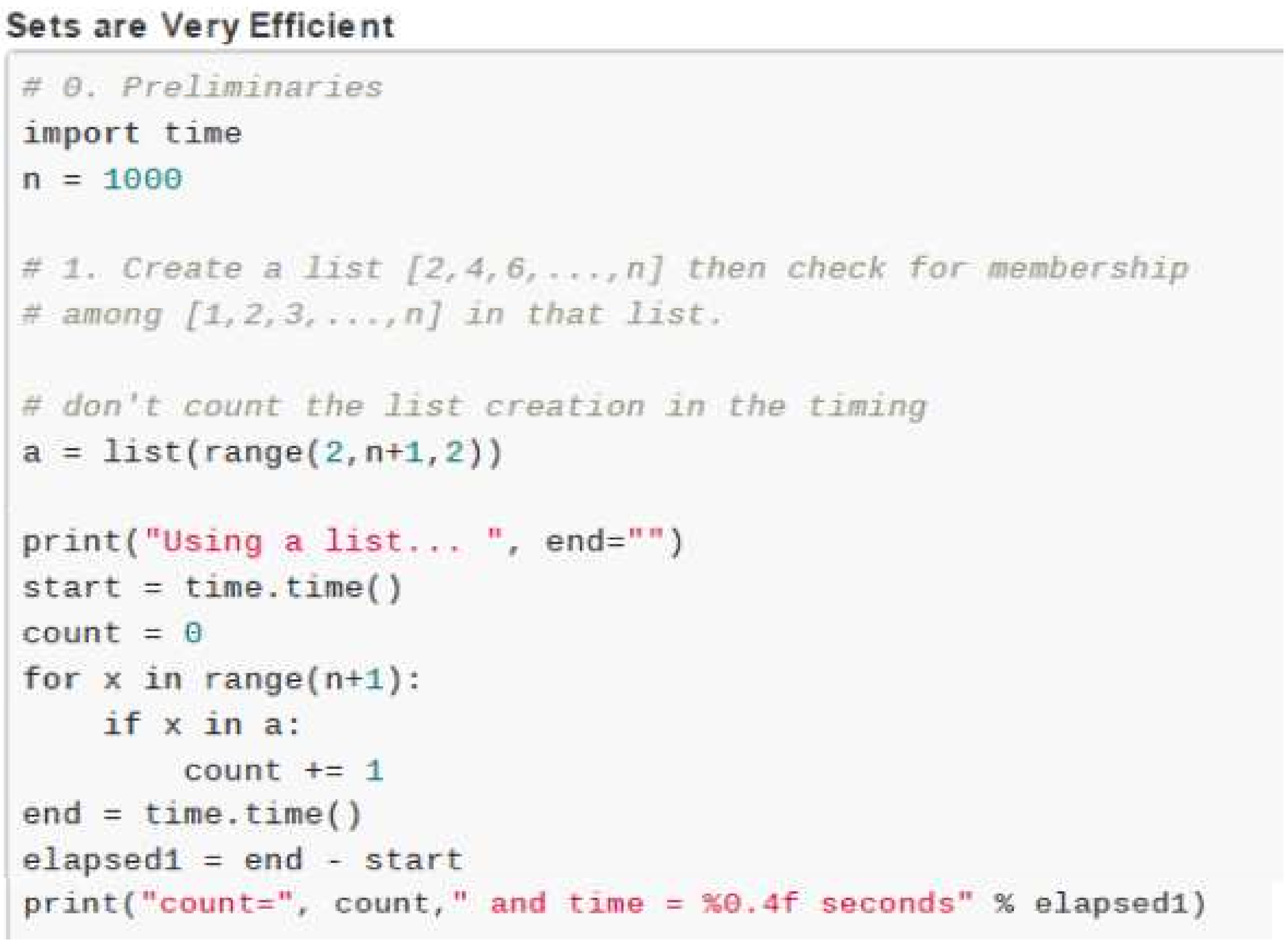
```
s = set([2,2,2])
print(s)              # prints {2}
print(len(s))         # prints 1
```

Elements Must Be Immutable

```
a = ["lists", "are", "mutable"]
s = set([a])        # TypeError: unhashable type: 'list'
print(s)
```

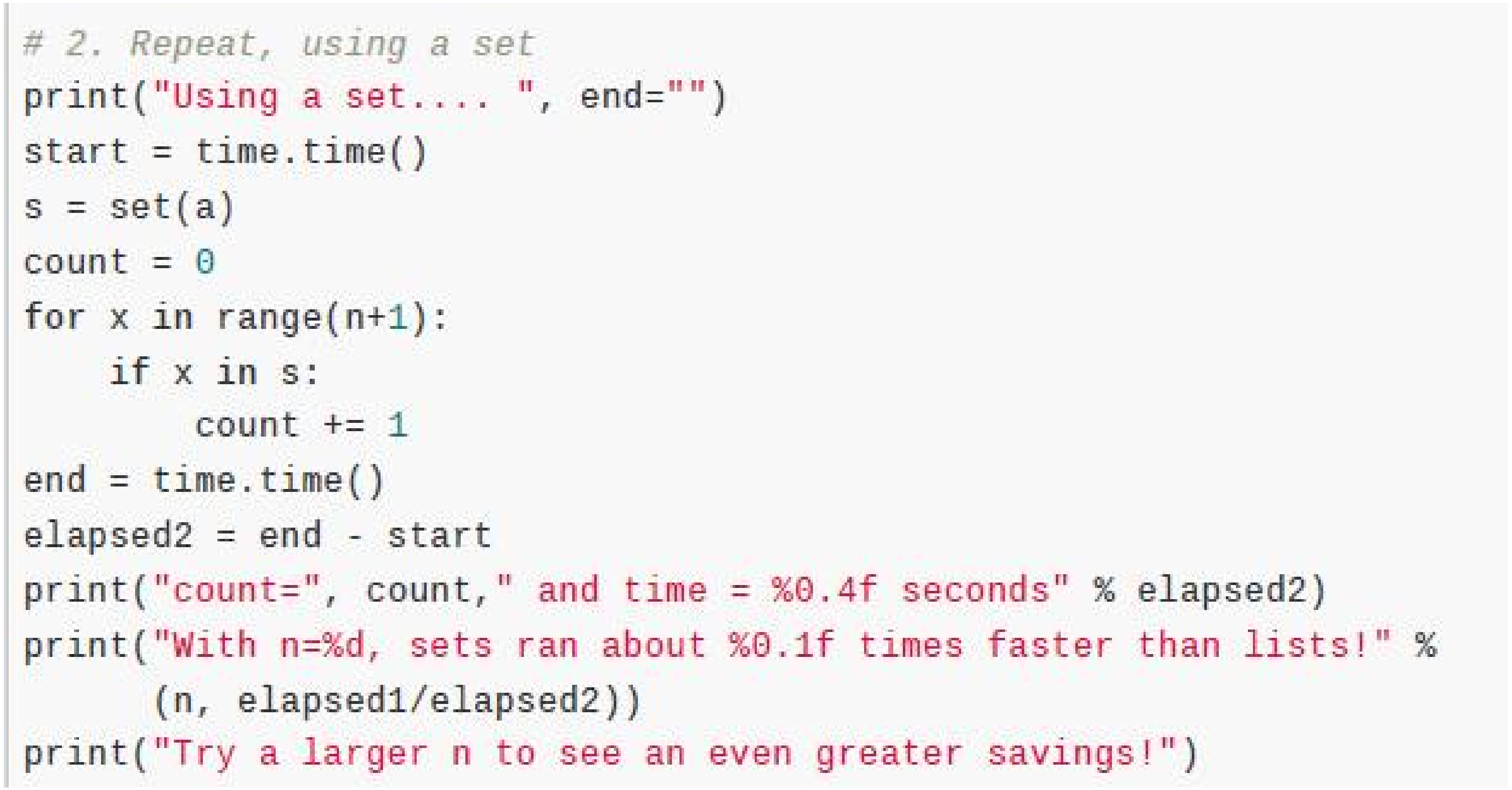
Another example:

```
s1 = set(["sets", "are", "mutable", "too"])
s2 = set([s1])       # TypeError: unhashable type: 'set'
print(s)
```



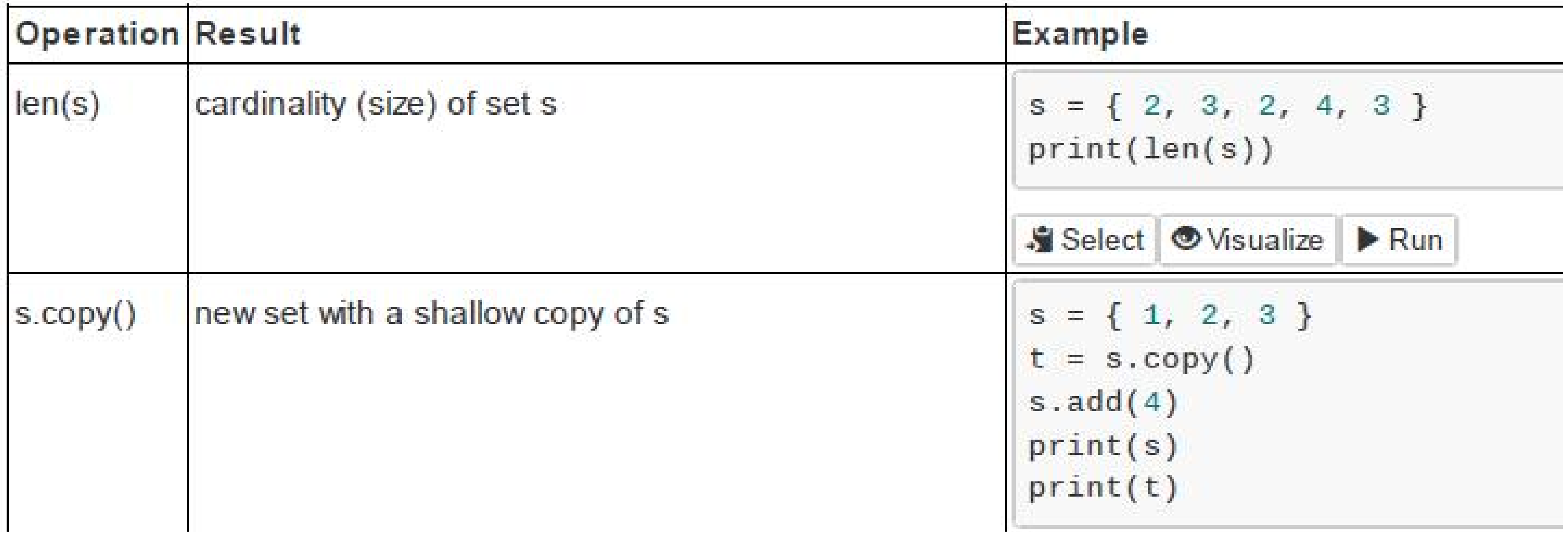
Sets are Very Efficient

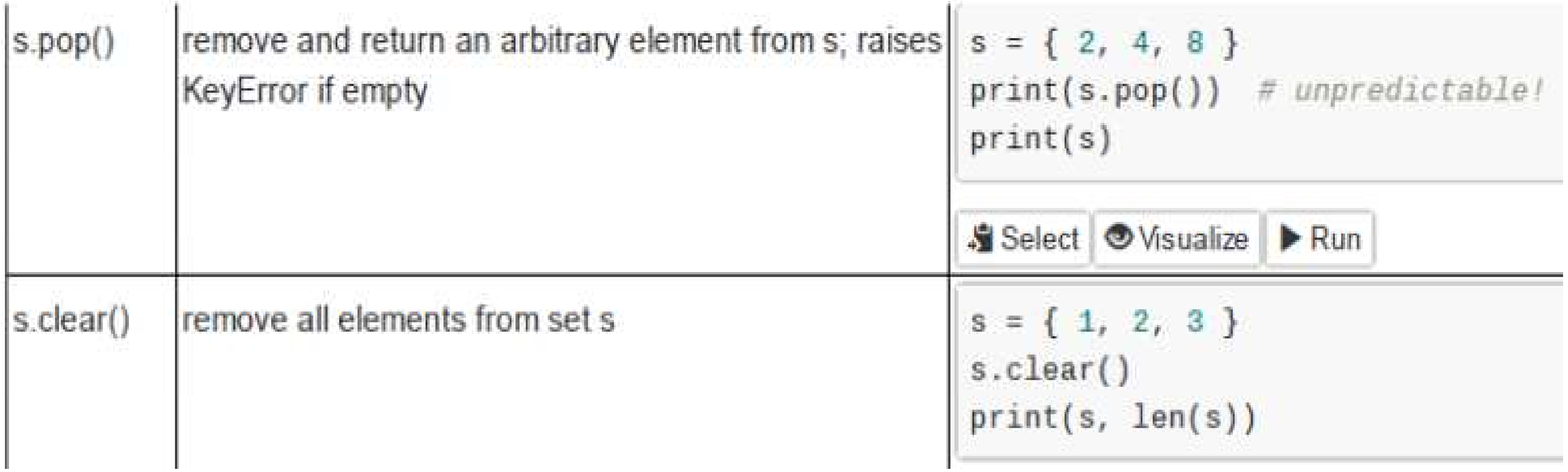
```
# 0. Preliminaries
import time
n = 1000

# 1. Create a list [2,4,6,...,n] then check for membership
# among [1,2,3,...,n] in that list.

# don't count the list creation in the timing
a = list(range(2,n+1,2))

print("Using a list... ", end="")
start = time.time()
count = 0
for x in range(n+1):
    if x in a:
        count += 1
end = time.time()
elapsed1 = end - start
print("count=", count," and time = %0.4f seconds" % elapsed1)
```



```python
# 2. Repeat, using a set
print("Using a set.... ", end="")
start = time.time()
s = set(a)
count = 0
for x in range(n+1):
    if x in s:
        count += 1
end = time.time()
elapsed2 = end - start
print("count=", count," and time = %0.4f seconds" % elapsed2)
print("With n=%d, sets ran about %0.1f times faster than lists!" %
      (n, elapsed1/elapsed2))
print("Try a larger n to see an even greater savings!")
```

Operations on a set


Select
Visualize
Run


Select
Visualize
Run

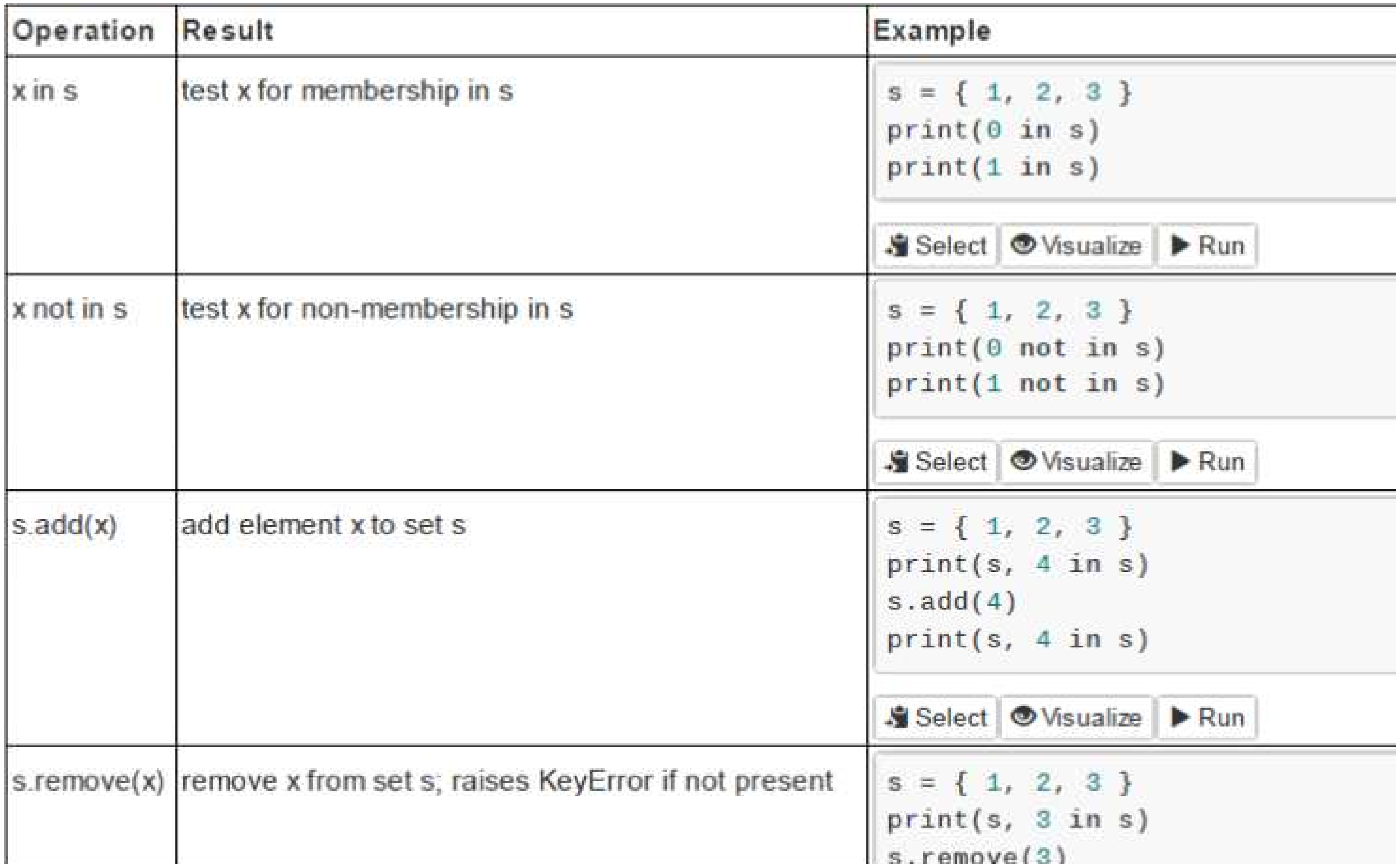
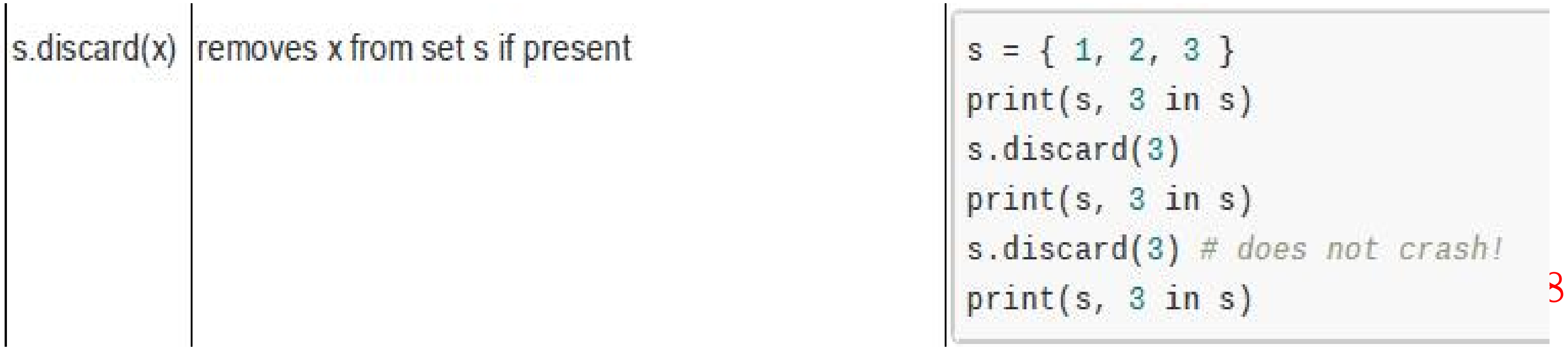
Operation	Result	Example

len(s)	cardinality (size) of set s	`s = { 2, 3, 2, 4, 3 }`
		`print(len(s))`

s.copy()	new set with a shallow copy of s	`s = { 1, 2, 3 }`
		`t = s.copy()`
		`s.add(4)`
		`print(s)`
		`print(t)`

s.pop()	remove and return an arbitrary element from s; raises KeyError if empty	`s = { 2, 4, 8 }`
		`print(s.pop())  # unpredictable!`
		`print(s)`

s.clear()	remove all elements from set s	`s = { 1, 2, 3 }`
		`s.clear()`
		`print(s, len(s))`

Operation	Result	Example

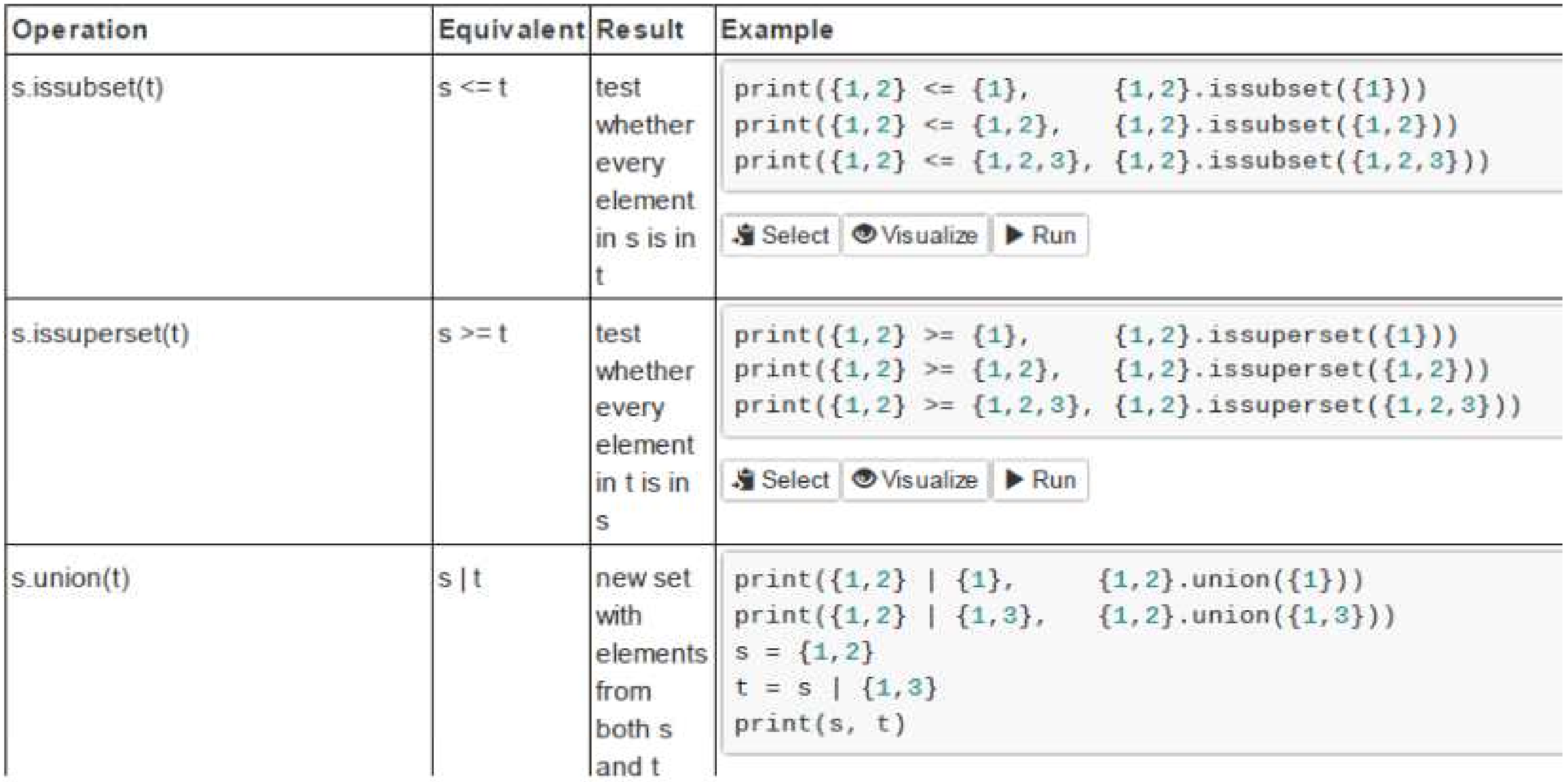
x in s	test x for membership in s	`s = { 1, 2, 3 }`
		`print(0 in s)`
		`print(1 in s)`
		Select Visualize ▶ Run

x not in s	test x for non-membership in s	`s = { 1, 2, 3 }`
		`print(0 not in s)`
		`print(1 not in s)`
		Select Visualize ▶ Run

s.add(x)	add element x to set s	`s = { 1, 2, 3 }`
		`print(s, 4 in s)`
		`s.add(4)`
		`print(s, 4 in s)`
		Select Visualize ▶ Run

s.remove(x)	remove x from set s; raises KeyError if not present	`s = { 1, 2, 3 }`
		`print(s, 3 in s)`
		`s.remove(3)`

s.discard(x)	removes x from set s if present	`s = { 1, 2, 3 }`
		`print(s, 3 in s)`
		`s.discard(3)`
		`print(s, 3 in s)`
		`s.discard(3) # does not crash!`
		`print(s, 3 in s)`

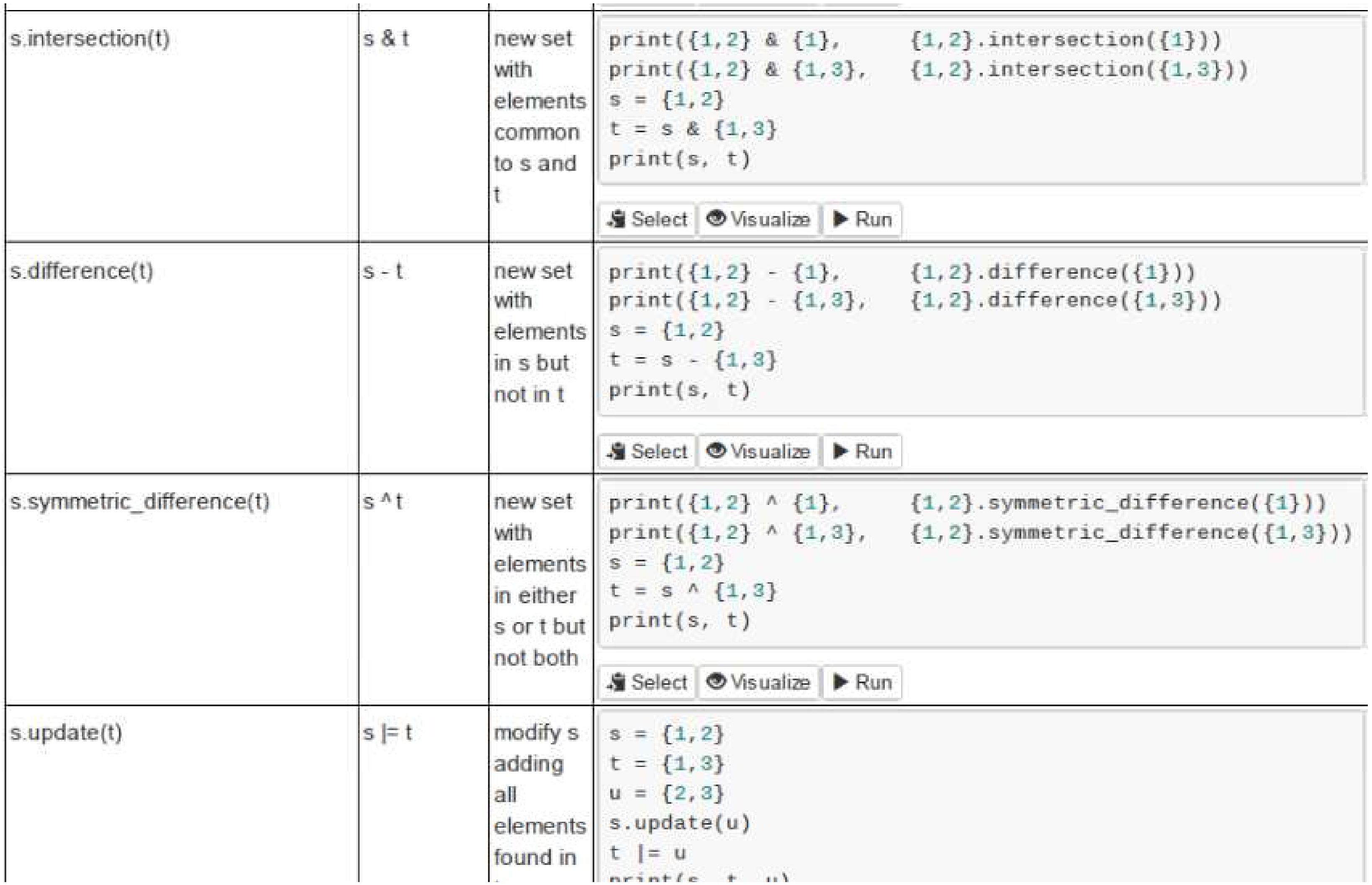


Operation	Equivalent	Result	Example

s.issubset(t)	s <= t	test whether every element in s is in t	`print({1,2} <= {1},     {1,2}.issubset({1}))`
			`print({1,2} <= {1,2},   {1,2}.issubset({1,2}))`
			`print({1,2} <= {1,2,3}, {1,2}.issubset({1,2,3}))`
			Select Visualize ▶ Run

s.issuperset(t)	s >= t	test whether every element in t is in s	`print({1,2} >= {1},     {1,2}.issuperset({1}))`
			`print({1,2} >= {1,2},   {1,2}.issuperset({1,2}))`
			`print({1,2} >= {1,2,3}, {1,2}.issuperset({1,2,3}))`
			Select Visualize ▶ Run

s.union(t)	s | t	new set with elements from both s and t	`print({1,2} | {1},     {1,2}.union({1}))`
			`print({1,2} | {1,3},   {1,2}.union({1,3}))`
			`s = {1,2}`
			`t = s | {1,3}`
			`print(s, t)`

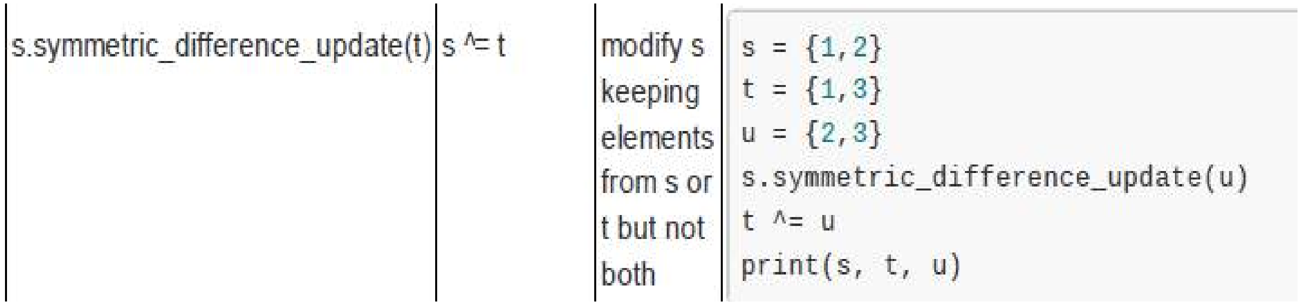


s.intersection(t)	s & t	new set with elements common to s and t	`print({1,2} & {1},     {1,2}.intersection({1}))`
			`print({1,2} & {1,3},   {1,2}.intersection({1,3}))`
			`s = {1,2}`
			`t = s & {1,3}`
			`print(s, t)`

s.difference(t)	s - t	new set with elements in s but not in t	`print({1,2} - {1},     {1,2}.difference({1}))`
			`print({1,2} - {1,3},   {1,2}.difference({1,3}))`
			`s = {1,2}`
			`t = s - {1,3}`
			`print(s, t)`

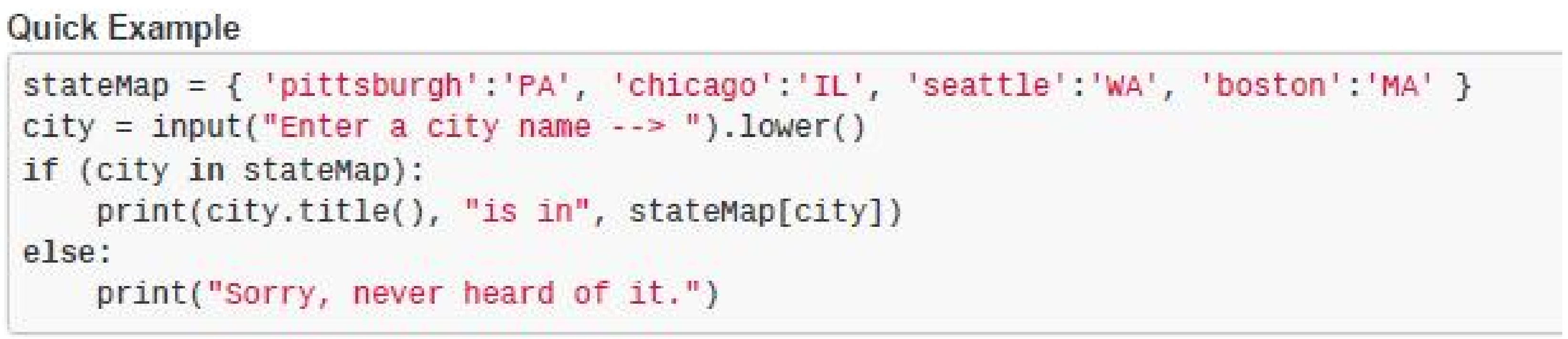
s.symmetric_difference(t)	s ^ t	new set with elements in either s or t but not both	`print({1,2} ^ {1},     {1,2}.symmetric_difference({1}))`
			`print({1,2} ^ {1,3},   {1,2}.symmetric_difference({1,3}))`
			`s = {1,2}`
			`t = s ^ {1,3}`
			`print(s, t)`

s.update(t)	s |= t	modify s adding all elements found in	`s = {1,2}`
			`t = {1,3}`
			`u = {2,3}`
			`s.update(u)`
			`t |= u`
			`print(s, t, u)`



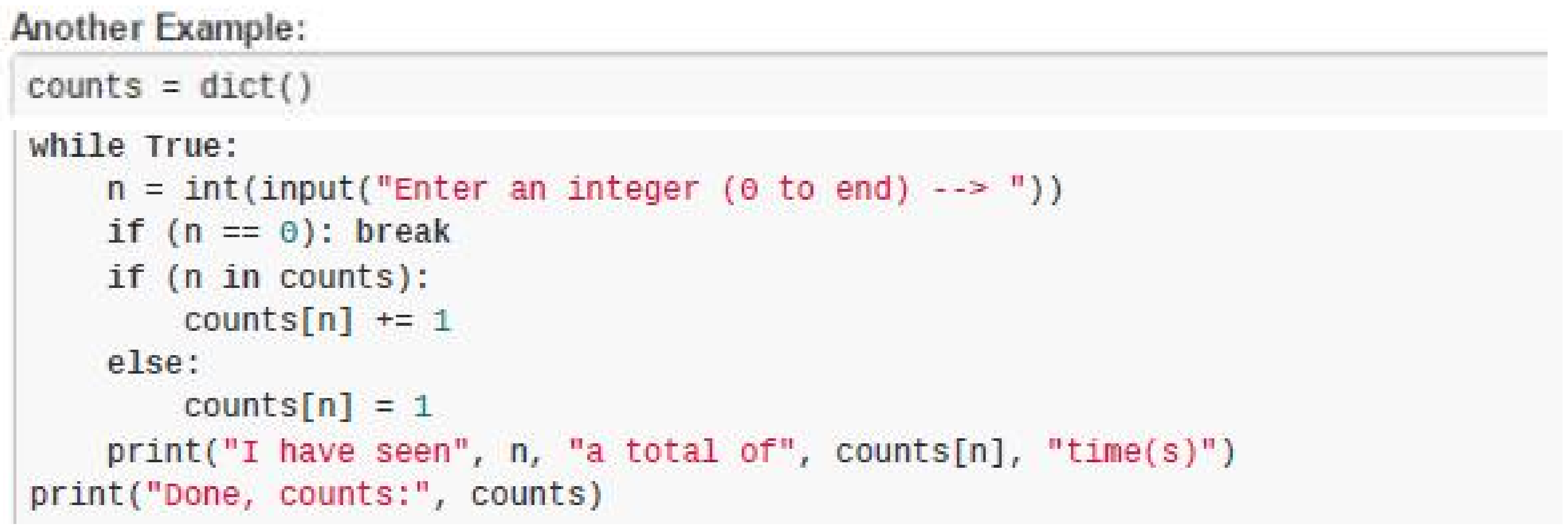
s.symmetric_difference_update(t)	s ^= t	modify s keeping elements from s or t but not both	`s = {1,2}`
			`t = {1,3}`
			`u = {2,3}`
			`s.symmetric_difference_update(u)`
			`t ^= u`
			`print(s, t, u)`

Dictionary in Python



Quick Example

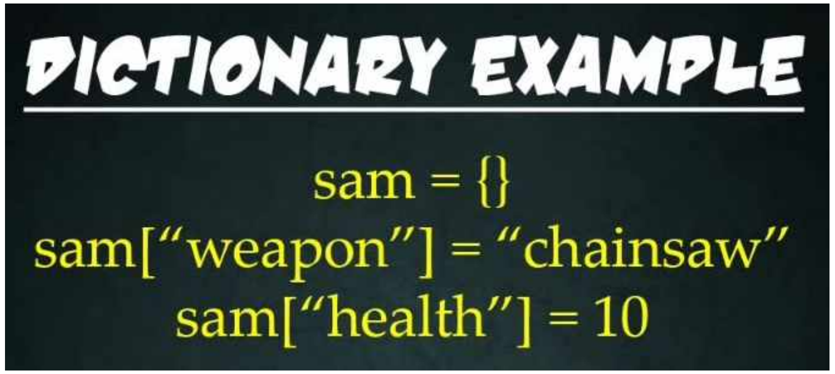
```python
stateMap = { 'pittsburgh':'PA', 'chicago':'IL', 'seattle':'WA', 'boston':'MA' }
city = input("Enter a city name --> ").lower()
if (city in stateMap):
    print(city.title(), "is in", stateMap[city])
else:
    print("Sorry, never heard of it.")
```



Another Example:

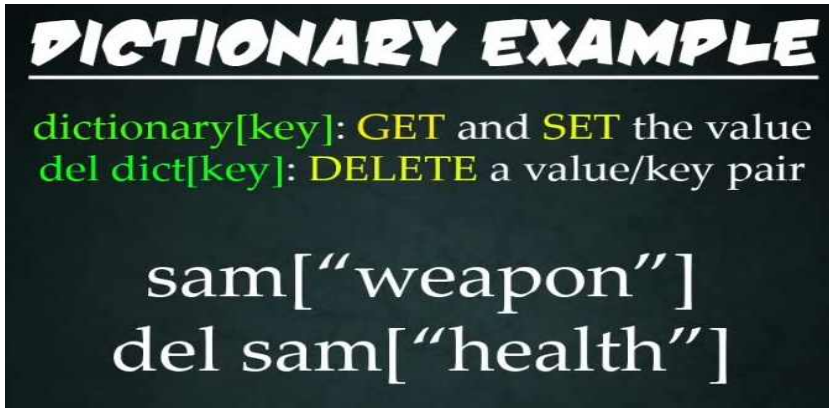
```python
counts = dict()
```

```python
while True:
    n = int(input("Enter an integer (0 to end) --> "))
    if (n == 0): break
    if (n in counts):
        counts[n] += 1
    else:
        counts[n] = 1
    print("I have seen", n, "a total of", counts[n], "time(s)")
print("Done, counts:", counts)
```



DICTIONARY EXAMPLE

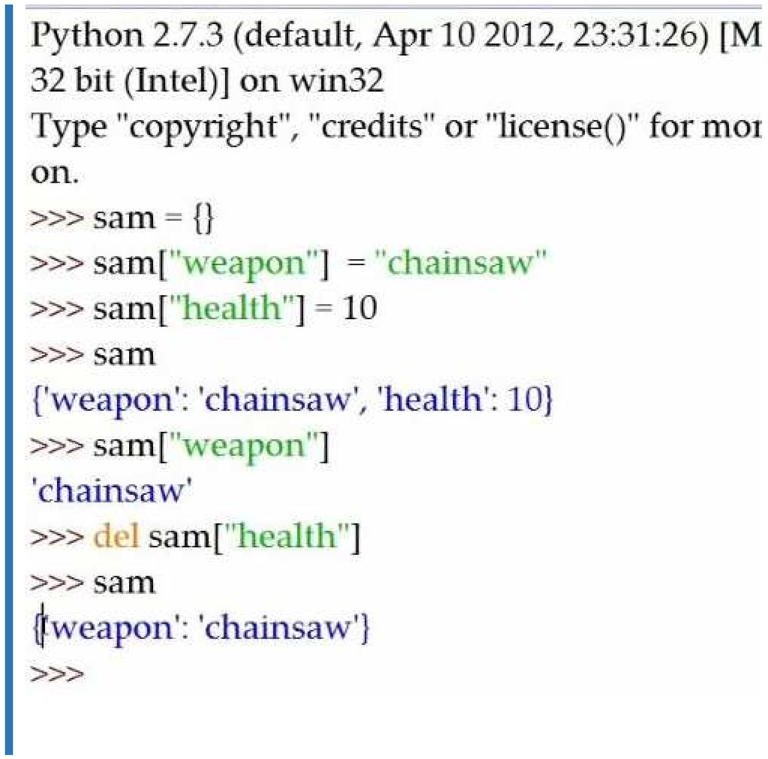
```
sam = {}
sam["weapon"] = "chainsaw"
sam["health"] = 10
```



DICTIONARY EXAMPLE

dictionary[key]: GET and SET the value
del dict[key]: DELETE a value/key pair

```
sam["weapon"]
del sam["health"]
```



```
Python 2.7.3 (default, Apr 10 2012, 23:31:26) [M
32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for mo
on.
>>> sam = {}
>>> sam["weapon"]  = "chainsaw"
>>> sam["health"] = 10
>>> sam
{'weapon': 'chainsaw', 'health': 10}
>>> sam["weapon"]
'chainsaw'
>>> del sam["health"]
>>> sam
{'weapon': 'chainsaw'}
>>>
```

```
myDict ➔ { key1:value1, key2: value2, key5:data5,….}
myDict[key8] = value13      # add a "key8:value13" pair
myDict[key2]                # retrieve the value part of
key2
del myDict[key5]            # delete the "key5:data5" pair
```

Create an empty dictionary

```
d = dict()
print(d)     # prints {}
```

Create an empty dictionary using braces syntax

```
d = { }
print(d)     # prints {}
```

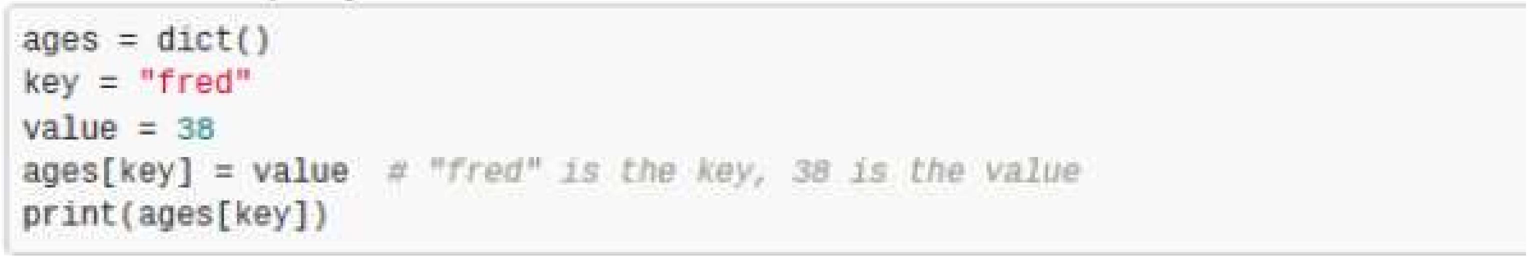
Create a dictionary from a list of (key, value) pairs

```
pairs = [("cow", 5), ("dog", 98), ("cat", 1)]
d = dict(pairs)
print(d)     # unpredictable order!
```

Statically-allocate a dictionary

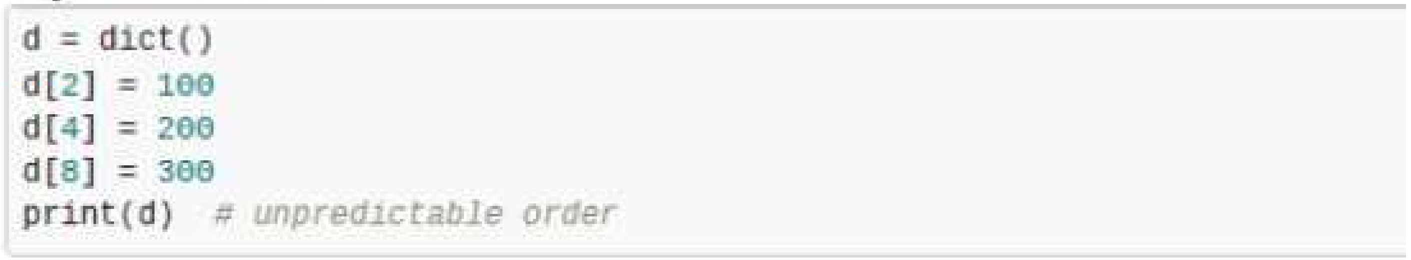
```
d = { "cow":5, "dog":98, "cat":1 }
print(d)     # ditto!
```

Dictionaries Map Keys to Values



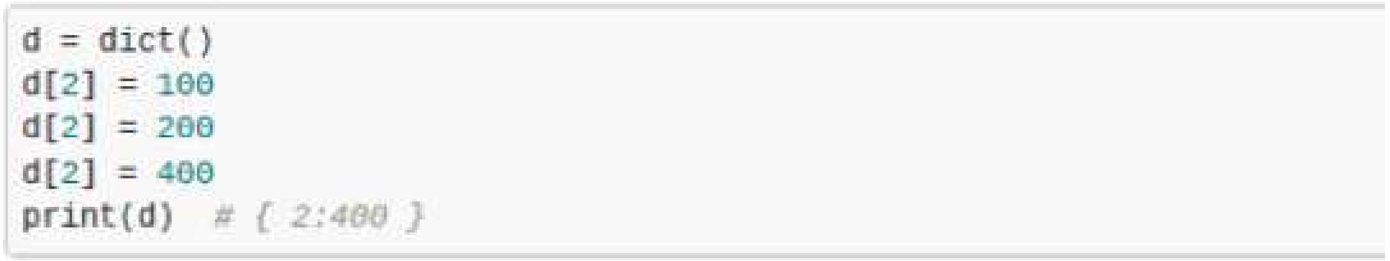
```
ages = dict()
key = "fred"
value = 38
ages[key] = value  # "fred" is the key, 38 is the value
print(ages[key])
```

Keys are unordered



```
d = dict()
d[2] = 100
d[4] = 200
d[8] = 300
print(d)  # unpredictable order
```

Keys are unique



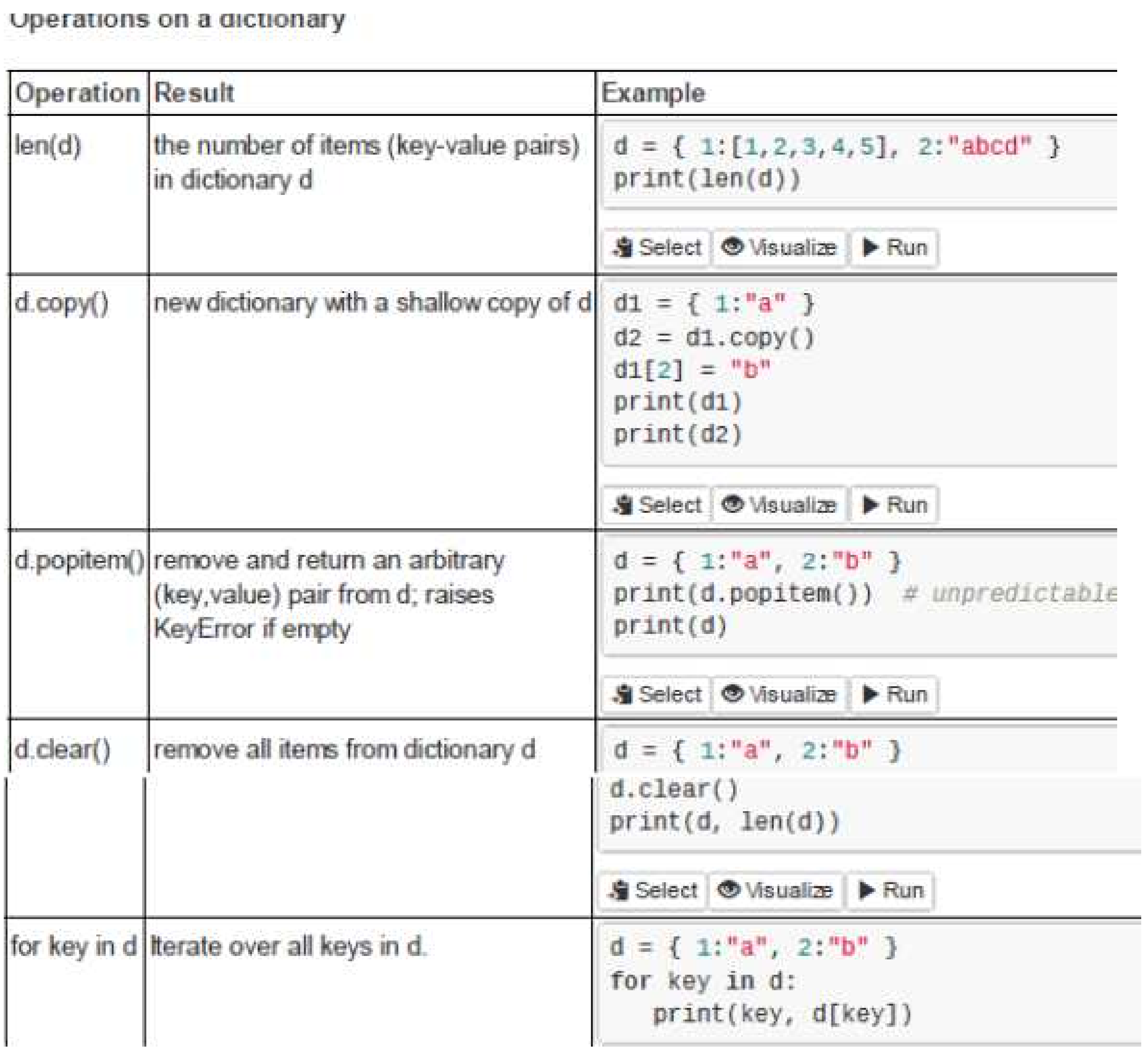
```
d = dict()
d[2] = 100
d[2] = 200
d[2] = 400
print(d)  # { 2:400 }
```

Keys must be immutable

```
d = dict()
a = [1] # lists are mutable, so...
d[a] = 42 # Error: unhashable type: 'list'
```

Values are Unrestricted

```
# values may be mutable
d = dict()
a = [1,2]
d["fred"] = a
print(d["fred"])
a += [3]
print(d["fred"]) # sees change in a!

# but keys may not be mutable
d[a] = 42       # TypeError: unhashable type: 'list'
```



Operations on a dictionary

Operation	Result	Example

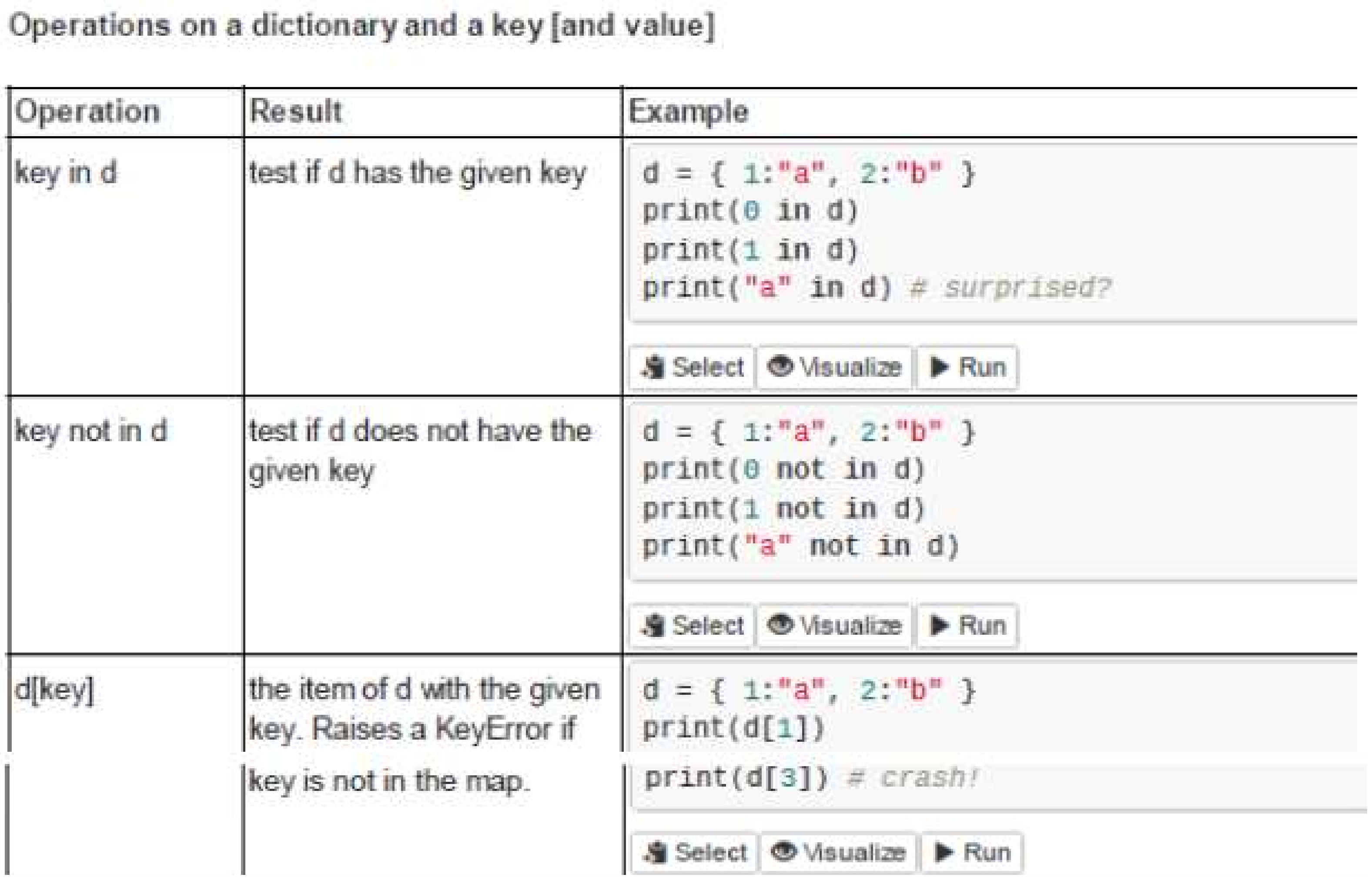
len(d)	the number of items (key-value pairs) in dictionary d	`d = { 1:[1,2,3,4,5], 2:"abcd" }`
		`print(len(d))`
		Select Visualize Run

d.copy()	new dictionary with a shallow copy of d	`d1 = { 1:"a" }`
		`d2 = d1.copy()`
		`d1[2] = "b"`
		`print(d1)`
		`print(d2)`
		Select Visualize Run

d.popitem()	remove and return an arbitrary (key,value) pair from d; raises KeyError if empty	`d = { 1:"a", 2:"b" }`
		`print(d.popitem())  # unpredictable`
		`print(d)`
		Select Visualize Run

d.clear()	remove all items from dictionary d	`d = { 1:"a", 2:"b" }`
		`d.clear()`
		`print(d, len(d))`
		Select Visualize Run

for key in d	Iterate over all keys in d.	`d = { 1:"a", 2:"b" }`
		`for key in d:`
		`    print(key, d[key])`


Select
Visualize
Run

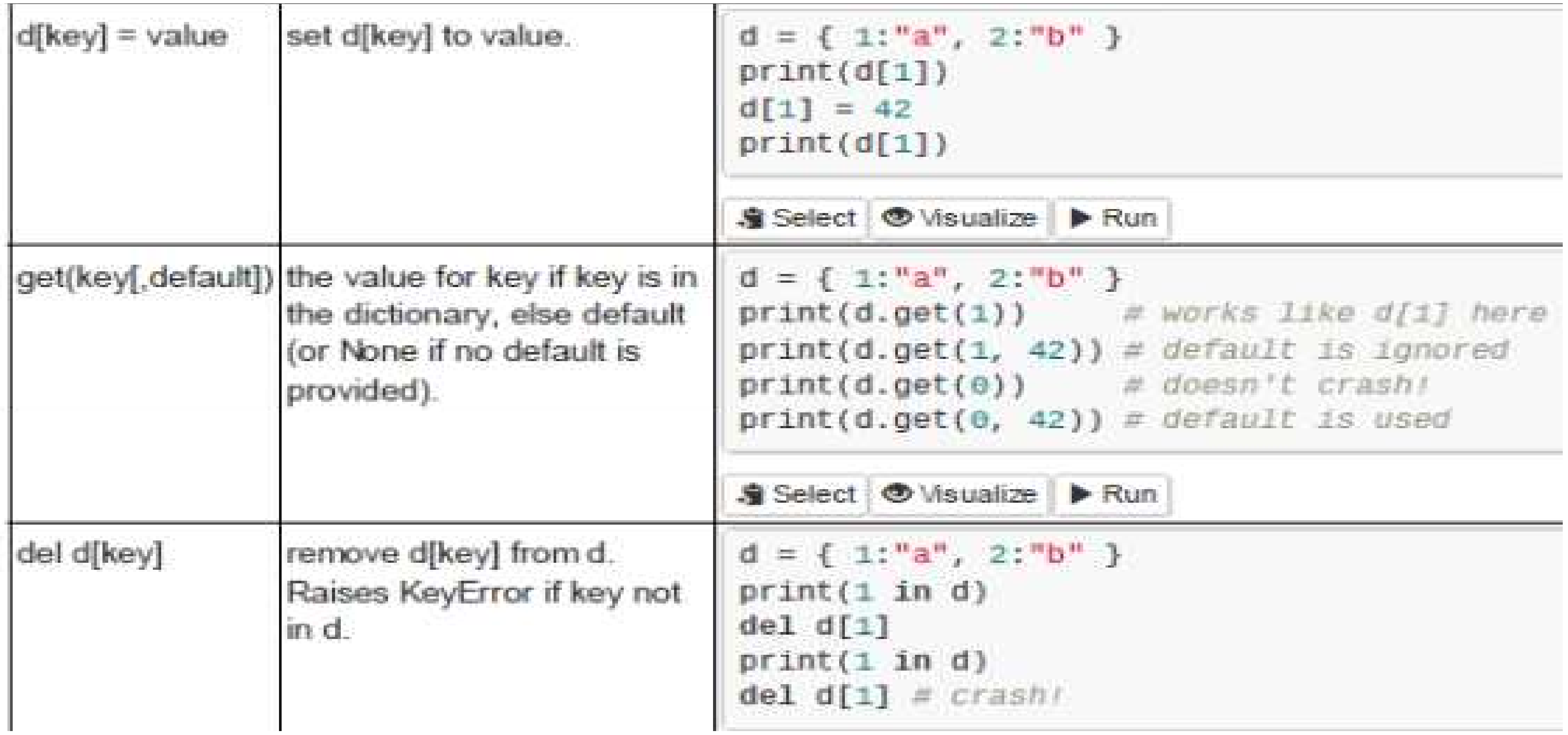
Operations on a dictionary and a key [and value]

Operation	Result	Example

key in d	test if d has the given key	`d = { 1:"a", 2:"b" }`
		`print(0 in d)`
		`print(1 in d)`
		`print("a" in d) # surprised?`

key not in d	test if d does not have the given key	`d = { 1:"a", 2:"b" }`
		`print(0 not in d)`
		`print(1 not in d)`
		`print("a" not in d)`

d[key]	the item of d with the given key. Raises a KeyError if key is not in the map.	`d = { 1:"a", 2:"b" }`
		`print(d[1])`
		`print(d[3]) # crash!`



d[key] = value	set d[key] to value.	`d = { 1:"a", 2:"b" }`
		`print(d[1])`
		`d[1] = 42`
		`print(d[1])`
		Select Visualize ▶ Run

get(key[,default])	the value for key if key is in the dictionary, else default (or None if no default is provided).	`d = { 1:"a", 2:"b" }`
		`print(d.get(1))      # works like d[1] here`
		`print(d.get(1, 42)) # default is ignored`
		`print(d.get(0))      # doesn't crash!`
		`print(d.get(0, 42)) # default is used`
		Select Visualize ▶ Run

del d[key]	remove d[key] from d. Raises KeyError if key not in d.	`d = { 1:"a", 2:"b" }`
		`print(1 in d)`
		`del d[1]`
		`print(1 in d)`
		`del d[1] # crash!`

Little bit Advanced Features

```
# suppose we have testFile.py as follows

def testFile(dest):
    print(dest)

if __name__ == '__main__':      # Is this the main file?
    createFile('ham')
    print('done!!')
```

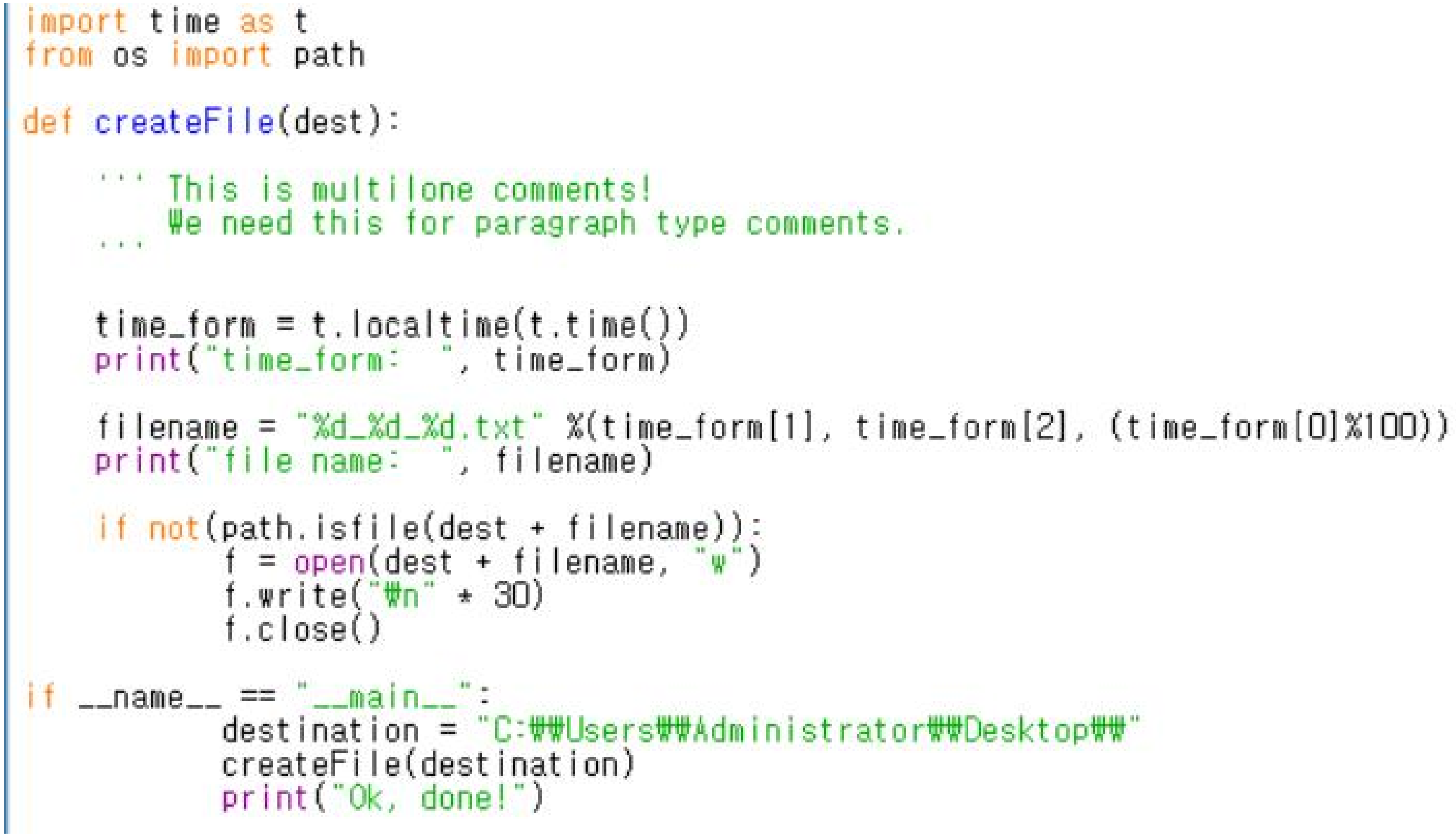
==

testFile.py를 Python interpreter에서 수행하면 (즉 python createTextFile.py 하면)
if __name__ = '__main__': 이 true가 되고 그 아래 문장들이 수행됨

반면에 import createTextFile 하면
if __name__ = '__main__': 이 false가 되고 그 아래 문장들이 수행이 안됨

** __name__ 은 python의 special variable로써 나를 부른 program의 이름을 가지고 있음

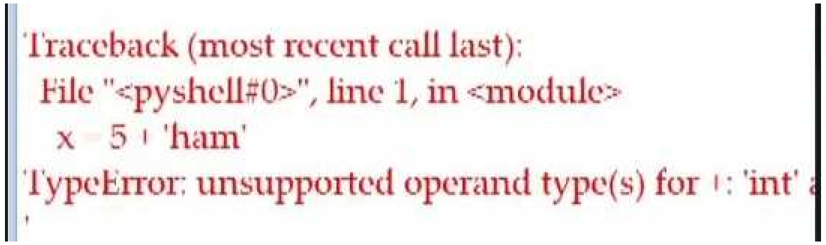
Create a "Month_Day_Year.txt" file
Within file, 30 blank lines



```
import time as t
from os import path

def createFile(dest):

    ''' This is multilone comments!
        We need this for paragraph type comments.
    '''

    time_form = t.localtime(t.time())
    print("time_form:  ", time_form)

    filename = "%d_%d_%d.txt" %(time_form[1], time_form[2], (time_form[0]%100))
    print("file name:  ", filename)

    if not(path.isfile(dest + filename)):
            f = open(dest + filename, "w")
            f.write("₩n" * 30)
            f.close()

if __name__ == "__main__":
            destination = "C:₩₩Users₩₩Administrator₩₩Desktop₩₩"
            createFile(destination)
            print("Ok, done!")
```

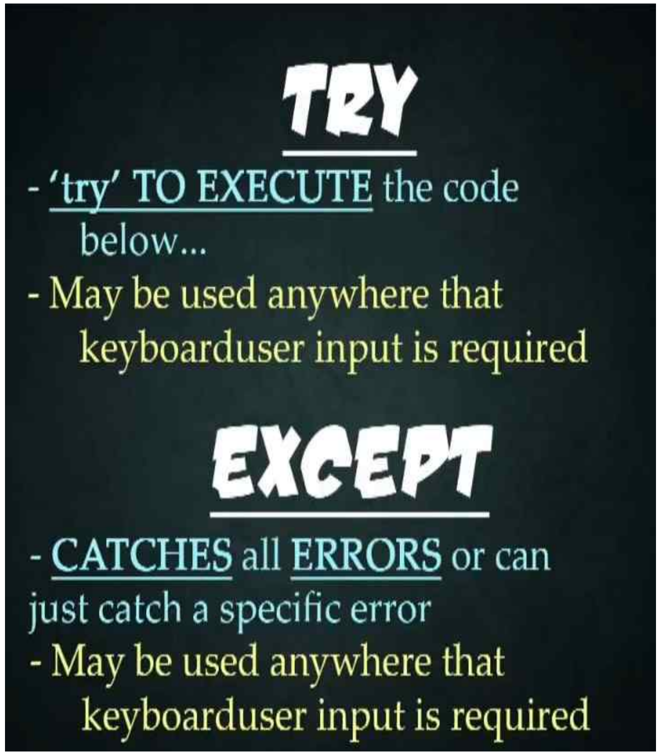
Exception Handling

x = 5 + “ham”



```
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    x = 5 + 'ham'
TypeError: unsupported operand type(s) for +: 'int' a
'
```

```
try:
    x = 5 +  "ham"
except:
    print("darn it")
```

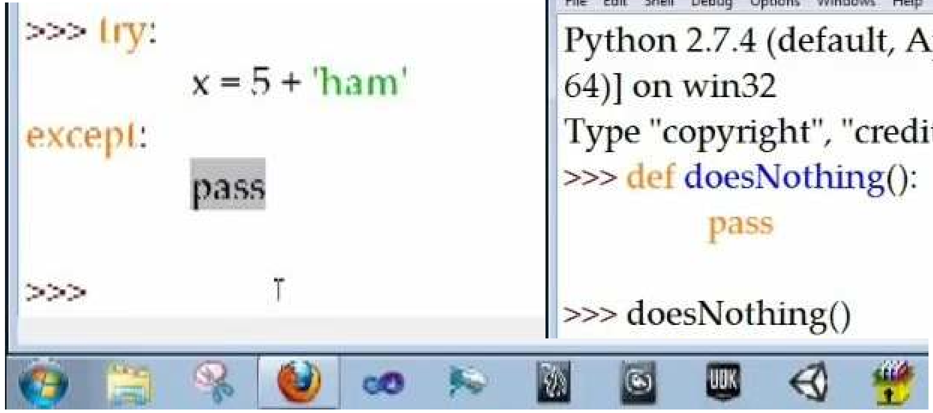


TRY

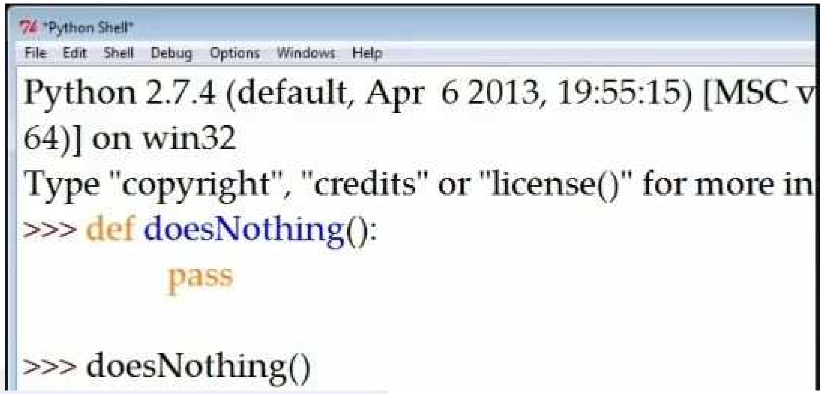
- ‘try’ TO EXECUTE the code below...
- May be used anywhere that keyboarduser input is required

EXCEPT

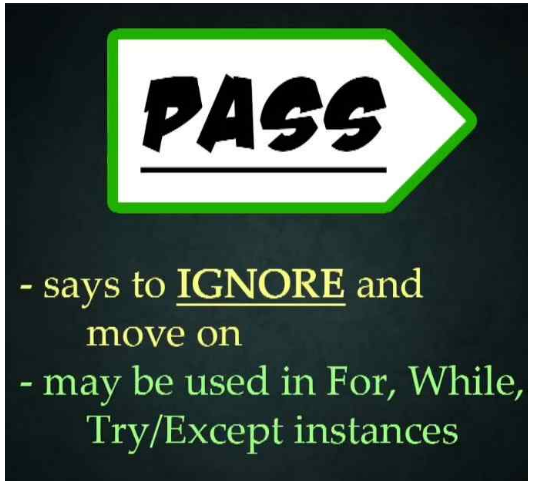
- CATCHES all ERRORS or can just catch a specific error
- May be used anywhere that keyboarduser input is required



```
>>> try:
        x = 5 + 'ham'
except:
        pass

>>>
```

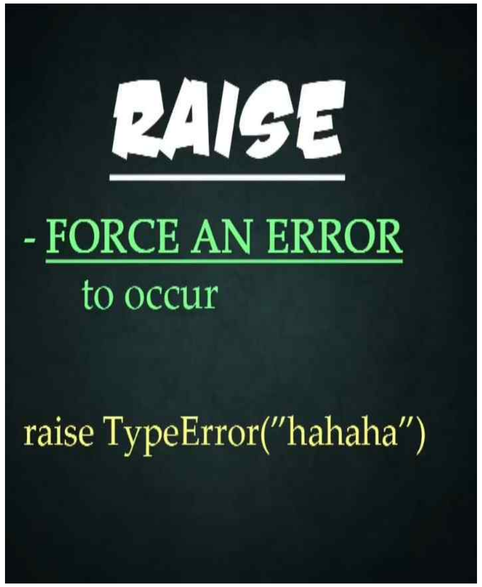


```
Python 2.7.4 (default, Apr  6 2013, 19:55:15) [MSC v
64)] on win32
Type "copyright", "credits" or "license()" for more in
>>> def doesNothing():
        pass

>>> doesNothing()
```



PASS

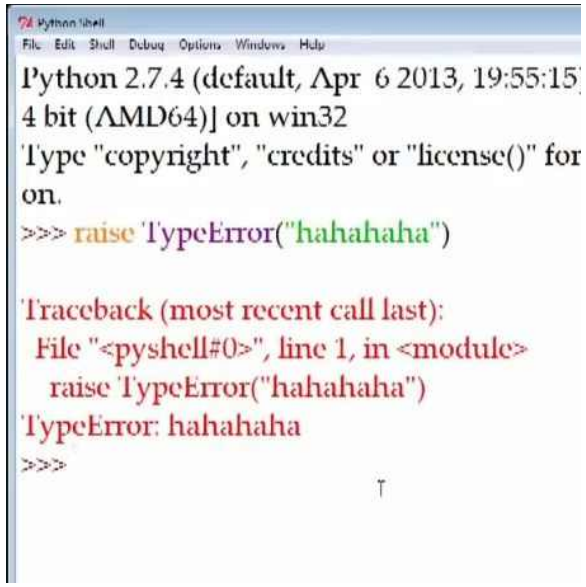
- says to IGNORE and move on
- may be used in For, While, Try/Except instances



RAISE

- FORCE AN ERROR
  to occur

raise TypeError("hahaha")



```
Python 2.7.4 (default, Apr  6 2013, 19:55:15)
4 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for
on.
>>> raise TypeError("hahahaha")

Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    raise TypeError("hahahaha")
TypeError: hahahaha
>>>
```

```
try:
    x = 5 +  "ham"

except ZeroDivisionError:
    print("darn it")

finally:
        print("Let's go further!")
```

- Exception Handling: A mechanism to handle exceptional problems.
  - It eliminates the need to check at each step of the algorithm

```
try:
      <body>
except:
      <exception handling>
```

- To explicitly filter out all error types

```
try:
      <body>
except <error_1> :
      <exception handling>
…
except <error_n> :
      <exception handling>
```

- To explicitly filter out error types and store the error as a variable

```
try:
        <body>
except <error_1> as <variable_1> :
        <exception handling>
…
except <error_n> as <variable_n> :
        <exception handling>
```

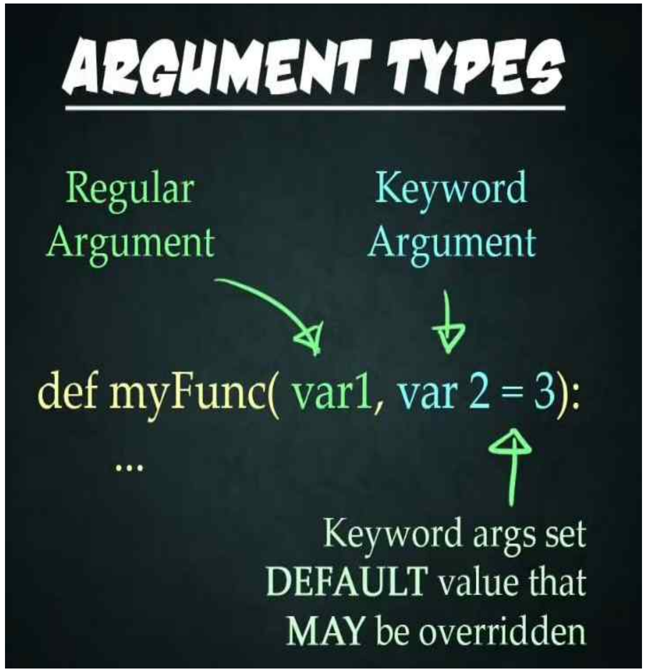
Exception Handling: Example

```
# quadratic5.py
#    A program that computes the real roots of a quadratic equation.
#    Illustrates exception handling to avoid crash on bad inputs

import math

def main():
    print("This program finds the real solutions to a quadratic\n")

    try:
        a, b, c = eval(input("Please enter the coefficients (a, b, c): "))
        discRoot = math.sqrt(b * b - 4 * a * c)
        root1 = (-b + discRoot) / (2 * a)
        root2 = (-b - discRoot) / (2 * a)
        print("\n The solutions are:", root1, root2)
    except ValueError:
        print("\n No real_number roots")
```

Exception Handling

- Full list of standard built-in exceptions (users may create their own) is listed here. https://docs.python.org/3/library/exceptions.html

- In the quadratic equation example, other types of exceptions may arise
  - not entering the right number of parameters (“unpack tuple of wrong size”),
  - entering an identifier instead of a number (NameError),
  - entering an invalid Python expression (TypeError).
  - Refer to sample code *quadratic6.py*



ARGUMENT TYPES

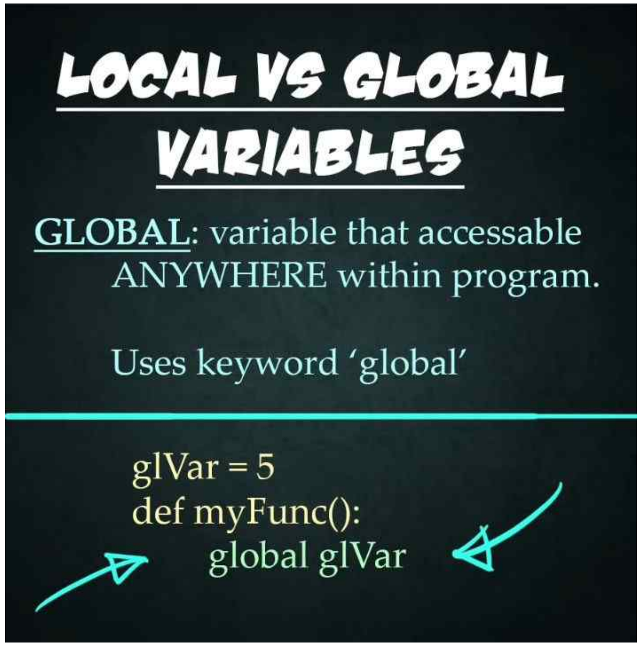
Regular Argument

Keyword Argument

```
def myFunc( var1, var 2 = 3):
    ...
```

Keyword args set DEFAULT value that MAY be overridden

```
def myFunc(var1, var2=3):
        return var1 +  var2

myFunc(10, 10)

myFunc(10)
```



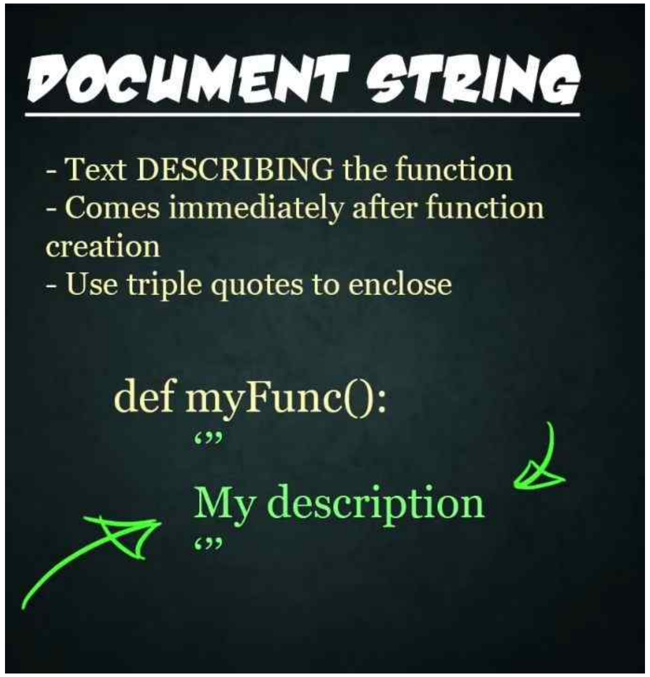
LOCAL VS GLOBAL VARIABLES

GLOBAL: variable that accessable ANYWHERE within program.

Uses keyword 'global'

```
glVar = 5
def myFunc():
    global glVar
```

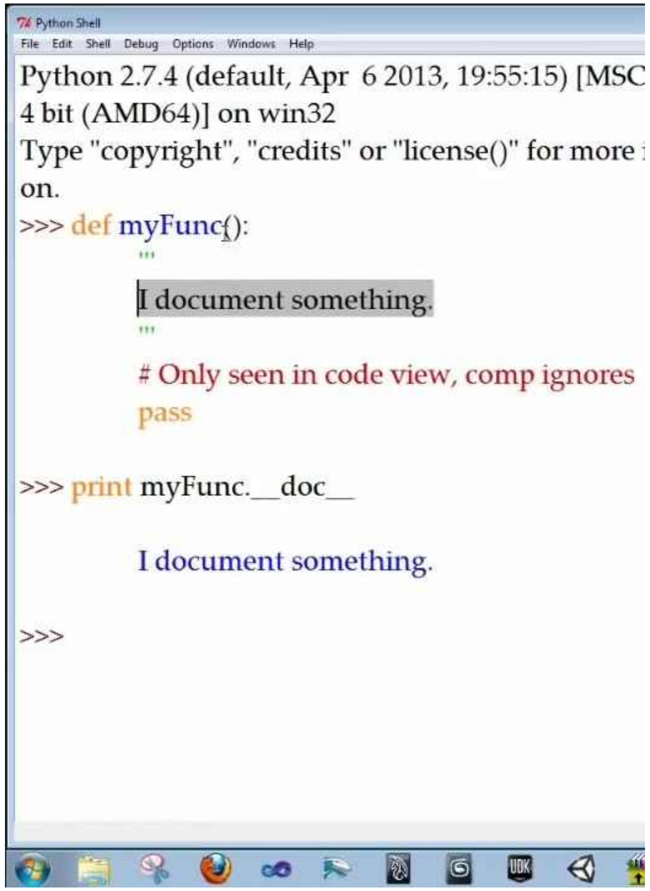
```
glVar = 5

def myFunc1():
      global glVar
      glVar = glVar – 10
      print("Current glVar: ", glVar)


def myFunc2():
      global glVar
       glVar = glVar +  10
      print("Current glVar: ", glVar)


myFunc1()
myFunc2()
```



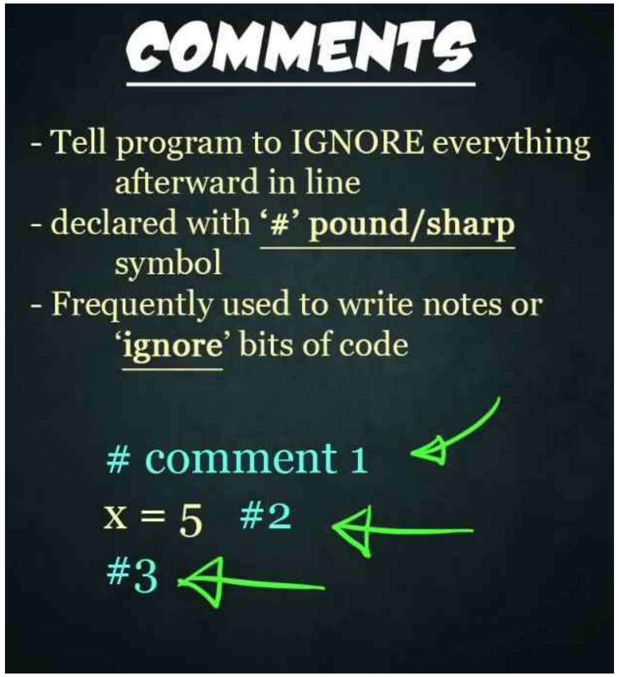
DOCUMENT STRING

- Text DESCRIBING the function
- Comes immediately after function creation
- Use triple quotes to enclose

```
def myFunc():
    “”
    My description
    “”
```



```
Python 2.7.4 (default, Apr  6 2013, 19:55:15) [MSC
4 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more
on.
>>> def myFunc():
            '''
            I document something.
            '''
            # Only seen in code view, comp ignores
            pass

>>> print myFunc.__doc__

            I document something.

>>>
```



COMMENTS

- Tell program to IGNORE everything afterward in line
- declared with ‘#’ **pound/sharp** symbol
- Frequently used to write notes or ‘**ignore**’ bits of code

```
# comment 1
x = 5   #2
#3
```