



- EXPLORER
- OPEN EDITORS
- percipio16_set_type.py Pe...
- PYTHON
- Automate-Boring-Stuff
 - my_code
 - Percipio_Python3-Course
 - 01_Start
 - 02_Data-Sequence Types
 - 03_Collections-Mapping-Lo...
 - percipio15_range_type_a...
 - percipio16_set_type.py
 - percipio17_dict_type.py
 - percipio18_while_loop.py
 - percipio19_forloop.py
 - percipio20_if_statement....
 - percipio21_exercise_nam...
 - 04_Modules-Functions
 - 05_Classes
 - 06_Working-with-Files
 - 07_Comprehensions
 - 08_Iterables-and-Generators
 - 09_Exceptions
 - Python Projects_2014
 - CMD_Python_Set-Path.txt
 - Python_Basics.txt
 - Python_Clear-Window-Comm..
 - python_exercises_00.py
 - python_exercises_01.py
 - Python_Tutorial_Running-Scr...
 - Python_Tutorials.md

```
percipio16_set_type.py x
1 # percipio16_set_type.py
2 # Percipio video: Collections, Mapping, & Looping; The Set type in Python
3 # Set class provides mapping of unique immutable elements
4 nl = '\n'
5 # (('',' ',' ',' '))
6 empty_set = set() # use the set function to create an empty set
7 print('empty_set ->', empty_set)
8 alpha = set(('a','b','c','d')) # use a set-constructor-function on a tuple-sequence creating a set with
9 curly-braces around an unordered list
10 print('alpha ->', alpha, '(use a set-constructor-function on a tuple-sequence creating a set with
11 curly-braces around an unordered list)')
12 print(nl)
13 # a useful feature of sets is to eliminate duplicates
14 dup_list = ['c','d','c','d', 'e','f'] # a list with duplicates
15 print('dup_list ->', dup_list, '(a list with duplicates)')
16 beta = set(dup_list) # This set-beta will have no duplicates in it
17 print('beta ->', beta, '(The dup_list-set with duplicates removed)')
18 uniq_list = list(beta) # Now convert that list without duplicates back into a list with unique set of elements
19 print('uniq_list ->', uniq_list, '(The dup_list set with duplicates removed recreated as a list)') # all
20 duplicates removed from dup_list
21 print(nl)
22 print('Set Operation Methods:' + nl + 'Union combines one set with another set also removing any duplicates'
23 + nl + 'Intersection finds where 2 sets have overlapping elements' + nl + 'Difference compares 2 sets
24 removing duplicate elements from the set in parenthesis' + nl + 'Symmetric_difference is the Union method
25 minus the intersection method')
26 print(nl)
27 # Set Operations
28 print(nl, 'Set Operations below use these 2 sets, alpha & beta')
29 print('alpha ->', alpha)
30 print('beta ->', beta)
31 print(nl)
32 # perform a combination of 2 sets through a union
33 print(nl, 'alpha.union(beta)')
34 gamma = alpha.union(beta) # union combines one set with another set also removing any duplicates (??WHY DOES
35 IT REMOVE DUPS??)
36 print('union or | ->', gamma, '(union combines 2 set-methods removing any duplicates)')
37 gamma = alpha | beta # another way to do a union
38 print('| or union ->', gamma, '("|" is another way to write union)')
39
40 print(nl, 'alpha.intersection(beta)')
41 delta = alpha.intersection(beta) # find where 2 sets overlap or intersect
42 print('intersection or & ->', delta, '(intersection finds where 2 sets have overlapping elements)')
43 delta = alpha & beta # another way to do a delta
44 print('& or intersection ->', delta, '("&" is another way to write intersection)')
45
46 print(nl, 'alpha.difference(beta)')
47 epsilon = alpha.difference(beta) # Do a set-difference using the difference method removing duplicate
```





EXPLORER

OPEN EDITORS

percipio16_set_type.py Pe...

PYTHON

Automate-Boring-Stuff

my_code

Percipio_Python3-Course

01_Start

02_Data-Sequence Types

03_Collections-Mapping-Lo...

percipio15_range_type_a...

percipio16_set_type.py

percipio17_dict_type.py

percipio18_while_loop.py

percipio19_forloop.py

percipio20_if_statement....

percipio21_exercise_nam...

04_Modules-Functions

05_Classes

06_Working-with-Files

07_Comprehensions

08_Iterables-and-Generators

09_Exceptions

Python Projects_2014

CMD_Python_Set-Path.txt

Python_Basics.txt

Python_Clear-Window-Comm..

python_exercises_00.py

python_exercises_01.py

Python_Tutorial_Running-Scr...

Python_Tutorials.md

percipio16_set_type.py

```
38 print('& or intersection ->', delta, '("&" is another way to write intersection)')
39
40 print(nl, 'alpha.difference(beta)')
41 epsilon = alpha.difference(beta) # Do a set-difference using the difference method removing duplicate
    elements from the set in parenthesis
42 print('difference or - ->', epsilon, '(difference compares 2 sets removing duplicate elements from the set in
    parenthesis)')
43 epsilon = alpha - beta # another way to do a epsilon
44 print('- or difference ->', epsilon, '("-" is another way to write difference)')
45
46 print(nl, 'alpha.symmetric_difference(beta)')
47 eta = alpha.symmetric_difference(beta) # Union method minus the intersection method checking for overlapping
    elements
48 print('symmetric_difference or ^ ->', eta, '(symmetric_difference is the Union method minus the intersection
    method)')
49 eta = alpha ^ beta # another way to do a eta
50 print('^ or symmetric_difference ->', eta, '("^" is another way to write symmetric_difference)')
51
52
53 print(nl, 'Set Comparisons')
54 # Set Comparisons
55 # isdisjoint - tests two sets for any shared elements
56 # issubset - tests if all elements of set(left) are within all elements of set(right),
57 # issuperset - tests if all elements of set(right) are within all elements of set(left),
58 print('- isdisjoint - tests two sets for any shared elements; boolean True with no elements in common' + nl +
    '- issubset - tests if all elements of set(left) are within all elements of set(right), boolean True when
    subset exists' + nl + '- issuperset - tests if all elements of set(right) are within all elements of set
    (left), boolean True when subset exists')
59 print(nl, '')
60 print('epsilon.isdisjoint(delta) ->', epsilon.isdisjoint(delta), '()') #
61 print('epsilon.isdisjoint(eta) ->', epsilon.isdisjoint(eta), '()') # boolean True with no elements in common
62 print(nl, '')
63 print('epsilon.issubset(eta) ->', epsilon.issubset(eta), '()') #
64 print('epsilon.issubset(beta) ->', epsilon.issubset(beta), '()') # boolean True when subset exists
65 print(nl, '')
66 print('eta.issuperset(epsilon) ->', eta.issuperset(epsilon), '()') #
67 print('beta.issuperset(epsilon) ->', beta.issuperset(epsilon), '()') # boolean True when superset exists
68
69 print(nl, '')
70 feta = frozenset(eta) # frozensets are immutable (unchanging), set operations are allowed such as comparisons,
    but no modifying nor updating.
71 print('feta ->', feta, '()') # frozenset({'a', 'b', 'e', 'f'}) ()
72
73 # Below applies to sets (mutable), not frozensets (immutable)
74 # With a set, you're able to add elements, but each addition must be unique.
75 print(nl, 'zeta')
76 zeta = set() #
77 print('zeta = set() ->' zeta, '()') #
```




EXPLORER

OPEN EDITORS

percipio16_set_type.py Pe...

PYTHON

Automate-Boring-Stuff

my_code

Percipio_Python3-Course

01_Start

02_Data-Sequence Types

03_Collections-Mapping-Lo...

percipio15_range_type_a...

percipio16_set_type.py

percipio17_dict_type.py

percipio18_while_loop.py

percipio19_forloop.py

percipio20_if_statement....

percipio21_exercise_nam...

04_Modules-Functions

05_Classes

06_Working-with-Files

07_Comprehensions

08_Iterables-and-Generators

09_Exceptions

Python Projects_2014

CMD_Python_Set-Path.txt

Python_Basics.txt

Python_Clear-Window-Comm..

python_exercises_00.py

python_exercises_01.py

Python_Tutorial_Running-Scr...

Python_Tutorials.md

percipio16_set_type.py

```
76 zeta = set() #
77 print('zeta = set() ->', zeta, '()') #
78 zeta.add(3) #
79 print('zeta.add(3) ->', zeta, '()') #
80 zeta.add(3) #
81 print('zeta.add(3) ->', zeta, '()') # adding an element which already exists in a set will be ignored, no
error
82 zeta.add(4) #
83 print('zeta.add(4) ->', zeta, '()') # {3, 4} ()
84
85 print(nl, 'gamma')
86 print('gamma ->', gamma, '()') # {'e', 'f', 'a', 'c', 'b', 'd'} ()
87 gamma.discard('a') #
88 print('gamma.discard("a") ->', gamma, '()') # will remove the element if present in the set
89 gamma.discard('z') # removes an element within a set, if element is not in set, results will be ignored
without an error
90 print('gamma.discard("z") ->', gamma, '()') #
91 gamma.remove('b') # removes an element within a set, if element is not in set, results in an ERROR
92 print('gamma.remove("b") ->', gamma, '()') #
93 random_element = gamma.pop() # pop method removes a random element
94 print('random_element returned ->', random_element, '(pop method removes a random element)')
95 print('gamma ->', gamma, '()') # {'f', 'c', 'd'} ()
96
97 print(nl, 'zeta')
98 zeta_ref = zeta # creates a variable that references an existing set. If set if changed, reference is
changed.
99 zeta_copy = zeta.copy() # copies a set creating a duplicate
100 zeta.clear() # CAUTION: removes ALL elements from a set
101 print('zeta ->', zeta, '()') # shows set is now clear
102 print('zeta_ref ->', zeta_ref, '()') # shows the variable-reference to that set is also clear
103 print('zeta_copy ->', zeta_copy, '()') # shows the copy of the set made before the clear is still populated
with elements
104
105 print(nl, 'alpha_diff')
106 print('alpha ->', alpha, '()') #
107 alpha_diff = alpha.copy() # copies a set creating a duplicate
108 alpha_diff.difference_update(beta) # update method using difference
109 print('alpha_diff ->', alpha_diff, '()') # Difference compares 2 sets removing duplicate elements from the
set in parenthesis
110
111 print(nl, 'alpha_intersect')
112 alpha_intersect = alpha.copy() # copies a set creating a duplicate
113 alpha_intersect.intersection_update(beta) # update method using intersection
114 print('alpha_intersect ->', alpha_intersect, '()') # finds where 2 sets overlap or intersect
115
116 print(nl, 'alpha_sym_diff')
117 alpha_sym_diff = alpha.copy() # copies a set creating a duplicate
118 alpha_sym_diff.symmetric_difference_update(beta) # update method using symmetric difference
```




EXPLORER

OPEN EDITORS

percipio16_set_type.py Pe...

PYTHON

- Automate-Boring-Stuff
- my_code
- Percipio_Python3-Course
 - 01_Start
 - 02_Data-Sequence Types
 - 03_Collections-Mapping-Lo...
 - percipio15_range_type_a...
 - percipio16_set_type.py
 - percipio17_dict_type.py
 - percipio18_while_loop.py
 - percipio19_forloop.py
 - percipio20_if_statement....
 - percipio21_exercise_nam...
 - 04_Modules-Functions
 - 05_Classes
 - 06_Working-with-Files
 - 07_Comprehensions
 - 08_Iterables-and-Generators
 - 09_Exceptions
- Python Projects_2014
- CMD_Python_Set-Path.txt
- Python_Basics.txt
- Python_Clear-Window-Comm..
- python_exercises_00.py
- python_exercises_01.py
- Python_Tutorial_Running-Scr...
- Python_Tutorials.md

percipio16_set_type.py x

```
117 alpha_sym_diff = alpha.copy() # copies a set creating a duplicate
118 alpha_sym_diff.symmetric_difference_update(beta) # update method using symmetric_difference
119 print('alpha_sym_diff ->', alpha_sym_diff, '()') # Union method minus the intersection method ((where 2 sets
    have overlapping elements))
120
121 print(nl, 'alpha_union')
122 alpha_union = alpha.copy() # copies a set creating a duplicate
123 alpha_union.update(beta) # update method using union
124 print('alpha_union ->', alpha_union, '()') # union combines alpha-set with beta-set also removing any
    duplicates
125
126 print(nl)
127 print(nl)
128 print('Other examples I made')
129 left = set((1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
130 right = set((2, 4, 6, 8, 10))
131 print('left ->', left)
132 print('right ->', right)
133 print(nl, '')
134 print('isdisjoint - tests two sets for any shared elements; boolean True with no elements in common' + nl +
    'issubset - tests if all elements of set(left) are within all elements of set(right), boolean True when
    subset exists' + nl + 'issuperset - tests if all elements of set(right) are within all elements of set(left),
    boolean True when subset exists')
135 print(nl, '')
136 print('left.isdisjoint(right) ->', left.isdisjoint(right), '(are any elements shared between two sets?)')
137 print('right.isdisjoint(left) ->', right.isdisjoint(left))
138 print(nl, '')
139 print('left.issubset(right) ->', left.issubset(right), '(are left-set-elements all within right-set-elements?)
    ')
140 print('right.issubset(left) ->', right.issubset(left))
141 print(nl, '')
142 print('left.issuperset(right) ->', left.issuperset(right), '(are right-set-elements all within
    left-set-elements?)')
143 print('right.issuperset(left) ->', right.issuperset(left))
144
```