KOLEKCJE

```
class list
class list(iterable)
class tuple
class tuple(iterable)
class dict(**kwarg)
class dict(mapping, **kwarg)
class dict(iterable, **kwarg)
vars()
vars(object)
class set
class set(iterable)
class frozenset(iterable=set())
class range(stop)
class range(start, stop, step=1)
class slice(stop)
class slice(start, stop, step=None)
```

k-kolekcja
l-lista
s-zbiór
fs – zbiór niemodyfikowalny (frozen set)

```
len(k)

filter(function, iterable)
map(function, iterable, *iterables)
any(iterable)
all(iterable)

reversed(seq)
sorted(iterable, /, *, key=None,
reverse=False)
zip(*iterables, strict=False)

sum(iterable, /, start=0)
max(iterable, *, key=None)
max(iterable, *, default, key=None)
max(arg1, arg2, *args, key=None)
min(iterable, *, key=None)
min(iterable, *, default, key=None)
min(arg1, arg2, *args, key=None)
```

```
l[i] = x
l[i:j] = t
del l[i:j]
l[i:j:k] = t
del l[i:j:k]

l.append(x)
l.extend(t) or s += t
l *= n
l.insert(i, x)
l.pop() or s.pop(i)
l.remove(x)

l.reverse()

ls.clear()
k.copy()

x in k
x not in k

k <= other
k < other
k >= other
k > other
```

SET, FROZEN SET

```
isdisjoint(other)
issubset(other)
issuperset(other)

union(*others)
set | other | ...
intersection(*others)
set & other & ...
difference(*others)
set - other - ...
symmetric_difference(other)
set ^ other
```

SET

```
add(elem)
remove(elem)
discard(elem)
pop()

update(*others)
set |= other | ...
intersection_update(*others)
set &= other & ...
difference_update(*others)
set -= other | ...
symmetric_difference_update(other)
set ^= other
```

ITERATORY

```
aiter(async_iterable)
awaitable anext(async_iterator)
awaitable anext(async_iterator, default)

enumerate(iterable, start=0)
iter(object)
iter(object, sentinel)
next(iterator)
next(iterator, default)
```

<table>
<tr>
<td>DICTIONARY - SŁOWNIKI

d = {'a': 1}
d | other
d |= other
update([other])

len(d)
list(d)
items()
keys()
values()

key in d
key not in d
d[key]
d[key] = value
get(key[, default])
pop(key[, default])
popitem()
del d[key]

setdefault(key[, default])</td>
<td>

iter(d)
reversed(d)

clear()
copy()

classmethod fromkeys(iterable[, value])</td>
<td>DICTIONARY VIEW OBJECT - OKULARY SŁOWNIKOWE

len(dictview)

x in dictview

iter(dictview)
reversed(dictview)

dictview.mapping</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>STRING - NAPISY

class str(object='')
class str(object=b'', encoding='utf-8', errors='strict')

str.encode(encoding='utf-8', errors='strict')</td>
<td>str.find(sub[, start[, end]])
str.index(sub[, start[, end]])
str.rfind(sub[, start[, end]])
str.rindex(sub[, start[, end]])
str.count(sub[, start[, end]])

str.endswith(suffix[, start[, end]])
str.startswith(prefix[, start[, end]])

str.join(iterable)
str.split(sep=None, maxsplit=- 1)
str.rsplit(sep=None, maxsplit=- 1)
str.partition(sep)
str.rpartition(sep)</td>
<td>str.replace(old, new[, count])

static str.maketrans(x[, y[, z]])
str.translate(table)</td>
</tr>
</table>

```
str.capitalize()
str.casefold()
str.lower()
str.upper()
str.swapcase()
str.title()

str.strip([chars])
str.lstrip([chars])
str.rstrip([chars])

str.removeprefix(prefix, /)
str.removesuffix(suffix, /)

str.ljust(width[, fillchar])
str.rjust(width[, fillchar])
str.center(width[, fillchar])
str.zfill(width)
str.expandtabs(tabsize=8)
```

```
str.format(*args, **kwargs)
str.format_map(mapping)
```

```
str.isascii()
str.isprintable()

str.isalnum()
str.isalpha()
str.islower()
str.isupper()
str.isidentifier()
str.istitle()
str.isspace()


str.isdecimal()
str.isdigit()
str.isnumeric()

ascii(object)
bin(x)
oct(x)
hex(x)
chr(i)
ord(c)
```

```
LICZBY

(całkowite, rzeczywiste, zespolone)

class int(x=0)
class int(x, base=10)
class float(x=0.0)
class complex(real=0, imag=0)
class complex(string)
```

```
divmod(x, y)
pow(base, exp, mod=None)

round(number, ndigits=None)
math.trunc(x)
math.floor(x)
math.ceil(x)

abs(x)

.is_integer()
.as_integer_ratio()

int(x)
float(x)
complex(x)
```

```
complex(re,im)
c.conjugate()
```

x+y x-y x*y x/y x//y x%y x**y +x -x	< > <= >= == !=	class bool(x=False) is is not

repr(object) format(value, format_spec='') print(*objects, sep=' ', end='\n', file=None, flush=False) input() input(prompt) open(file, mode='r', buffering=- 1, encoding=None, errors=None, newline=None, closefd=True, opener=None)	eval(expression, globals=None, locals=None) compile(source, filename, mode, flags=0, dont_inherit=False, optimize=- 1) exec(object, globals=None, locals=None, /, *, closure=None) help() help(request) breakpoint(*args, **kws) class memoryview(object)	
KLASY i OBIEKTY class object class property(fget=None, fset=None, fdel=None, doc=None) class super class super(type, object_or_type=None) @classmethod decorator @staticmethod decorator	setattr(object, name, value) delattr(object, name) getattr(object, name) getattr(object, name, default) hasattr(object, name) locals() globals() dir() dir(object) callable(object) hash(object)	id(object) isinstance(object, classinfo) issubclass(class, classinfo)