

### Transforming Number Bases

Transform a number string of base a to a decimanl integer	<code>int(string, a)</code>
Transform an integer n to binary, octidecimal or hexadecimal	<code>bin(n), oct(n), hex(n)</code>

### Generating Series of numbers

<code>[0,a)</code>	<code>range(a)</code>
<code>[1,a)</code>	<code>range(1,a)</code>
<code>[a,b) skipping c</code>	<code>range(a,b,c)</code>

### Strings

Input returns a string	<code>input('prompt')</code>
Tab character	<code>\t</code>
New line character	<code>\n</code>
Quotation mark character	<code>\"</code>
Backslash character	<code>\\</code>
String index [a,b) with c steps	<code>string[a:b:c]</code>
Length including tabs, newlines,...	<code>len(string)</code>
Converts a char into an order	<code>ord("char")</code>
Converts an order into a char	<code>chr(number)</code>

### String Methods

#### Case Modification

`str.upper(), str.lower(), str.capitalize(), str.title()`

#### String Searching and Counting

Count occurrences	<code>str.count("s")</code>
If not found, returns -1	<code>str.find("s")</code>
If not found, returns ValueError	<code>str.index("s")</code>
finds last occurence	<code>str.rfind("s")</code>

#### Stripping Whitespace

`str.strip(), str.lstrip(), str.rstrip()`

#### Replacement and Modification

Expand tabs	<code>str.expandtabs()</code>
Replace a with b	<code>str.replace(a,b)</code>

#### Splitting and joining

`str.split(), " ".join(list)`

#### Alignment and formatting

`str.center(20, "-"), str.ljust(20, "-"), str.rjust(20, "-"), str.zfill(20)`

#### Prefix and suffix removal

### String Methods (cont)

`str.removeprefix("p"), str.removesuffix("s")`

#### Testing strings: (returns true or false)

<code>str.isalpha(), str.istitle()</code>	
Includes non-ASCII digits	<code>str.isnumeric()</code>
Only ASCII digits	<code>str.isdigit()</code>
Contains num or alpha or both	<code>str.isalnum()</code>

### Basic Programming Constructs

#### Conditional Execution

#### Iterative Execution

#### Flags

while active:

#### Count-controlled loop

for

#### More stamenents

Leaves the loop `break`

returns to the beginning

null operation

### F Strings

`f"{expression} this is a string"`

Break the statement into multiple lines

`\`

Braces can be included using double braces

`{{ }}`

Expressions should not use comments using `#`, nor you can use backslashes to escape in the expression

### List

#### Adding or changing elements

`list.append(x), list.extend(list2), list.insert(i, x), list[i]=x`

#### Removing elements

Removes the first occurrence of x `list.remove(x)`

Removes and returns [i] element `list.pop(i)`

Removes all elements from the list

Deletes [i] element `del list[i]`

#### Searching and counting



### List (cont)

Returns the index of first x occurrence between [a,b) indices and returns "ValueError" if x is not found

`list.index(x,a,b)`

Counts the occurrences of "x"

#### Sorting

Modifies original list and returns none, key = lambda, len, function\_name

`list.sort(key=None, reverse=False)`

Returns a new sorted list

#### List membership

"x in list", "x not in list"

#### Iterating through a list

for x in list:

for i,x in enumerate(list):

[expression for x in list if condition]

[expression if condition else expression2 for x in list]

[expression for x in [expression2 for y in list]]

for x,y in zip(list2,list2):

#### Transposing the matrix

`t_matrix = zip(*matrix)`

#### Other operators

`list.reverse()`, `len(list)`, `list.copy()`, `min(list)`, `max(list)`, `y = []+x`, `y= x[:]`, `y=list(x)`

### Sets

#### Set creation

`set = {1,2,3}`, `set()`

#### Adding/removing elements

`set.add(a)`, `set.update(set2)`, `set.clear()`

Raises a `KeyError` if a is not found

`set.remove(a)`

Does not give key error

`set.discard(a)`

Removes and returns a, returns `KeyError` if a not found

`set.pop(a)`

#### Set operations

`Set1USet2`

`set1.union(set2)`

`set1|set2`

Set1 and Set2

### Sets (cont)

`set1 & set2`

All in Set1 but not in Set2

`set1.difference(set2)`

In Set1 and Set2 but not in both

#### Set comparison

True if set1 is subset of set2

True if set1 is superset of set2

True if set1 has no elements in common with set2

#### Set membership

x in set, x not in set

#### Looping through sets

for x in set

{statement for x in set}

#### Set updates

removes elements from set1 that are also in set2

`set1.difference_update(set2)`

keeps only the intersection

`set1.intersection_update(set2)`

keeps only the symmetric difference

`set1.symmetric_difference_update(set2)`

#### More operations

`set.copy()`, `len(set)`, `min(set)`, `max(set)`

### Tuples

#### Tuple Creation

`tuple = (1,2,3)`, `single_tuple = (1,)`, `tuple()`

#### Accessing elements

[a,b]

`tuple[a:b]`

#### Tuple unpacking

unpacks elements into separate variables a,b,c

`a,b,c = tuple`

#### Checking membership

x in tuple, x not in tuple

#### Miscellaneous



### Tuples (cont)

`len(tuple), tuple(sorted(my_tuple))`

#### Named tuples

`from collections import namedtuple`

`Person = namedtuple('Person', ['name', 'age'])`

`alice = Person('Alice',30)`

`alice.name, alice.age`

#### Tuple methods

`tuple.count(a), tuple.index(a)`

#### Concatenation

`tuple1 + tuple2`

#### Looping in a tuple

`for x in tuple:`

`for i, x in enumerate(tuple):`

`new_tuple = tuple(expression for x in tuple)`

`for x in zip(list1,list2):`

Tuples are immutable. Once created, their elements cannot be changed or added.

### Dictionary

#### Dictionary Creation

`dic = {"key": "value"}, dict(), dict(zip(list1,list2))`

#### Accessing Values

Returns "KeyError" if key is not found `dic[key]`

Returns a default\_value if key is not found `dic.get(key, default_value)`

Returns value, if key not present, adds the key with default\_value and returns default\_value

#### Updating Values

`dic[key]=new_value`

#### Methods

returns a view of all keys

returns a view of all values

returns a view of all key-value pairs

#### Adding/removing elements

### Dictionary (cont)

removes and return the value deleted, KeyError if key not found

removes and returns last key-value pair as a tuple, KeyError if dic is empty

deletes the key-value pair

#### Dictionary size

`len(dic)`

#### More methods

Checking membership

Clearing dictionary

Copying

Adds new keys and updates existing keys

#### Looping

`for key in dic`

`for value in dic.values()`

`for key,value in dic.items():`

`{key,value for key,value in dic.items}`

### Importing Modules

#### Standard import

`import module_name`

`module_name.function_name()`

#### Alias import

`import module_name as a`

`a.func()`

#### Dynamic import

`a = __import__ ("module_name")`

`a.func()`

#### Importing and renaming within a module

`from module_name import function_name as func`

`func()`



### Functions

#### Defining a function

```
def function_name(parameter1, parameter2):
```

```
    statement
```

```
    return result
```

#### Calling a function

```
print(function_name(a,b))
```

#### Packing arguments

```
def function_name(*arg):
```

```
    function_name(a,b,c,d)
```

#### Unpacking arguments

```
def func(a,b,c,d):
```

```
    func(*tuple)
```

#### Packing a dictionary

```
def func(**kwargs):
```

```
    func(key1 = value1, key2 = value2)
```

#### Global and local variable

```
x = "global variable"
```

```
def func():
```

```
    ____ global x # Change x outside the func to be the x inside the func
```

```
    ____ x = "local variable"
```

#### Function documentation AKA Docstring

```
def func():
```

```
    ____ " " "This is a docstring" " "
```

#### Help

```
help(func)
```

### Lambda Functions

#### Basic

```
square = lambda x: x^2
```

```
square(2)
```

#### Default argument

```
lambda x, y = a: x*y
```



By [leenmajz](https://www.cheatography.com/leenmajz/)

[cheatography.com/leenmajz/](https://www.cheatography.com/leenmajz/)

Published 14th January, 2024.

Last updated 17th January, 2024.

Page 4 of 4.

Sponsored by [CrosswordCheats.com](https://www.crosswordcheats.com)

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>