Tuples (immutable)

```
    tuple = ()
```

Lists (mutable)

```
    list = []
    list[i:j]  # returns list subset
    list[-1]   # returns last element
    list[:-1]  # returns all but the last element
    *list      # expands all elements in place

    list[i] = val
    list[i:j] = otherlist  # replace ith to jth-1 elements with otherlist
    del list[i:j]

    list.append(item)
    list.extend(another_list)
    list.insert(index, item)
    list.pop()        # returns and removes last element from the list
    list.pop(i)       # returns and removes i-th element from the list
    list.remove(i)    # removes the first item from the list whose value is
i
    list1 + list2     # combine two list
    set(list)         # remove duplicate elements from a list

    list.reverse()    # reverses the elements of the list in-place
    list.count(item)
    sum(list)

    zip(list1, list2)  # returns list of tuples with n-th element of both
list1 and list2
```

```
    list.sort()        # sorts in-place, returns None
    sorted(list)       # returns sorted copy of list
    ",".join(list)     # returns a string with list elements seperated by
comma
```

Dict

```
    dict = {}
    dict.keys()
    dict.values()
    "key" in dict    # let's say this returns False, then...
    dict["key"]      # ...this raises KeyError
    dict.get("key")  # ...this returns None
    dict.setdefault("key", 1)
    **dict           # expands all k/v pairs in place
```

Iteration

```
    for item in ["a", "b", "c"]:
    for i in range(4):         # 0 to 3
    for i in range(4, 8):      # 4 to 7
    for i in range(1, 9, 2):  # 1, 3, 5, 7
    for key, val in dict.items():
    for index, item in enumerate(list):
```

String

```
    str[0:4]
    len(str)

    string.replace("-", " ")
    ",".join(list)
    "hi {0}".format('j')
    f"hi {name}" # same as "hi {}".format('name')
    str.find(",")
    str.index(",")   # same, but raises IndexError
    str.count(",")
    str.split(",")

    str.lower()
    str.upper()
    str.title()

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

```
str.islower()

/* escape characters */
>>> 'doesn\'t'  # use \' to escape the single quote...
    "doesn't"
>>> "doesn't"  # ...or use double quotes instead
    "doesn't"
>>> '"Yes," they said.'
    '"Yes," they said.'
>>> "\"Yes,\" they said."
    '"Yes," they said.'
>>> '"Isn\'t," they said.'
    '"Isn\'t," they said.'
```

Casting

```
int(str)
float(str)
str(int)
str(float)
'string'.encode()
```

Comprehensions

```
[fn(i) for i in list]           # .map
map(fn, list)                   # .map, returns iterator

filter(fn, list)                # .filter, returns iterator
[fn(i) for i in list if i > 0]  # .filter.map
```

Regex

```
import re

re.match(r'^[aeiou]', str)
re.sub(r'^[aeiou]', '?', str)
re.sub(r'(xyz)', r'\1', str)

expr = re.compile(r'^...$')
expr.match(...)
expr.sub(...)
```

File manipulation

Reading

```
file = open("hello.txt", "r") # open in read mode 'r'
file.close()
```

```
print(file.read())  # read the entire file and set the cursor at the end of
file
print file.readline() # Reading one line
file.seek(0, 0) # place the cursor at the beginning of the file
```

Writing (overwrite)

```
file = open("hello.txt", "w") # open in write mode 'w'
file.write("Hello World")

text_lines = ["First line", "Second line", "Last line"]
file.writelines(text_lines)

file.close()
```

Writing (append)

```
file = open("Hello.txt", "a") # open in append mode
file.write("Hello World again")
file.close()
```

Context manager

```
with open("welcome.txt", "r") as file:
    # 'file' refers directly to "welcome.txt"
   data = file.read()

# It closes the file automatically at the end of scope, no need for
`file.close()`.
```

Download Pdf File