Types

str, int, float

```
a = "hello!"     # string
count = 3        # integer
pi = 3.14        # float
```

list ordered collection

```
a = ["a", "b", 3]
a[0]          # "a"
a[1]          # "b"
a[-1]         # "3"
a[1:2]        # ["b", 3]
```

tuple same as list, but immutable

```
a = ("a", "b", 3)
```

dict collection of keys and values

```
a = {"test": 1, "b": "hello"}
a["test"]       # 1
a["b"]          # "hello"
del a["test"]   # delete "test"
a["c"] = 3  # add "c" to dict
```

sets "keys-only dict", with operations

```
a = {"a", 1, 4, "b"}
b = {"a", "b"}
print(a - b)  #  {1, 4}
```

list methods

```
a = ["a", "b", 3]
a.append(4) # ["a", "b", 3, 4]
a.reverse() # [4, 3, "b", "a"]
```

dict methods

```
a = {"a": 1, "b": 2}
a.get("c", 3) # 3 as default
a.update({"d": 4}) # add more
a.keys()   # iterable of keys
a.values() # ... of values
a.items()  # ... of both
```

Input/Output

Prompt user

```
name = input("Name? ")
print("Hi ", name)
```

Read from file and convert to `str`

```
a = open("file.txt").read()
print("data:", a.decode("utf-8"))
```

Write to file creating if none

```
a = "Some text for o.txt"
open("o.txt", "w+").write(a)
```

Branching

Basic if Conditionally execute indent

```
if cost < 10:
    print("impulse buy")
```

Boolean operators "and", "or"

```
if age > 17 and place == "UK":
    print("can buy alcohol")
if age < 18 or s == "student":
    print("can get discount")
```

If-elif-else

```
if beer == "Darkwing":
    print("IPA")
elif beer == "Stonehenge":
    print("Stout")
else:
    print("Unknown beer")
```

Pass placeholder that does nothing

```
if cost > 1.99:
    pass # TODO: finish this
```

Iteration

While loop Repeat indented code until condition is no longer true

```
i = 2
while i < 10000:
    print("square:", i)
    i = i ** 2
```

For loop Repeat for each item in iterable

```
names = ["John", "Paul", "G"]
for name in names:
    print("name:", name)
for x in range(0, 100):
    print("x:", x)
```

List comprehension Create a new list while looping

```
names = ["John", "Paul", "G"]
long_names = [
    n.lower() for n in names
    if len(n) > 2
]   # = ["john", "paul"]
```

Interruption Exit loops prematurely with `break`, skip to next iteration with `continue`

Functions

Return value w/ positional param

```
def in_file(name):
    path = "./src/" + name
    return path + ".html"
path = in_file("home")
html = open(path).read()
```

Keyword parameters

```
def greet(name="Jack"):
    print("Hello", name)
greet(name="Jill")
```

Variable length arguments

```
def do_all(*args, **kwargs):
    print(kwargs) # kwargs is dict
    return sum(args)
do_all(3, 5, b=3)
```

Comment aka "docstring"

```
def plural(word):
    """
    Return the plural of
    an English word.
    """
    if word.endswith("s"):
        return word + "es"
    return word + "s"
print("Many", plural("cat"))
```

Lambda alternative syntax for one-liners

```
cubed = lambda i: i ** 3
print("5^3 is ", cubed(5))
```

More

Try / except Handle or ignore errors.

```
try: big_number = 1 / 0
except Exception as e:
    print("It broke:", e)
```

With Execute code in a context

```
with open("file.txt") as f:
    f.write("test")
```

Unpacking assignment Assign to two or more, good for loops

```
x, y = [35, 15]
pairs = [(10, 5), (8, 100)]
for left, right in pairs:
    print(left * right)
```