

## TYPES

**str** string of characters

```
a = "hello!"
a.upper()      # HELLO!
a.capitalize() # Hello!
a.strip("!")   # hello
a.index("e")   # 2
a.split("e")   # ["h", "llo!"]
```

**number types**

```
count = 3 # int
pi = 3.14 # float
```

**list** sequentially hold other data

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

**tuple** same as list, but immutable

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

**dict** associate “key” with “values”

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

**list methods**

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

**dict methods**

```
a = {"hi": "hola", "b": 3}
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
```

## KEY TERMS

**Variable** A named “bucket” that you can put data into.**Assignment** The act of putting data into a variable.**Function** A bit of code given a name. Functions attached to “objects” (data types) are called *methods*.**Invoke** To cause a function to run (aka *calling* the function). Written as parenthesis which optionally enclose *arguments*.**Arguments** Data provided to a function when calling a function.

## BRANCHING

**Basic if** Optionally execute indented code based on the truth value of the *condition*

```
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 == "Hefe":
    print("Hefeweizen")
elif beer == "Stonehenge":
    print("Stout")
else:
    print("Unknown beer")
```

**Pass** placeholder that does nothing

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

## ITERATION

**For loop** Execute the indented code for each item in a list or other “iterable”, temporarily putting that item in a given variable

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

**Range for-loop** Useful for looping through numbers

```
for x in range(0, 100):
    print("x:", x)
```

**While loop** Repeat indented code until condition is no longer true

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

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

```
for i in range(0, 50):
    choice = input("quit/skip? ")
    if choice == "quit":
        break
    elif choice == "skip":
        continue
    print("i", i, "i^2", i ** 2)
```

## FUNCTIONS

**Positional parameters**

```
def add(a, b):
    c = a + b
    print("the sum is", c)
add(1, 2)
```

**Keyword parameters**

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

**Return value**

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

**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** alt. syntax for one-liners

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

## PROGRAMMING CONCEPTS

**Pseudocode** A “rough draft”, fake code in English which serves as a middle-ground in the process of converting high-level thought into low-level, real programming code**State diagram** Diagramming what different variables should hold at different times in the execution of your code.**Scope** The idea that variables assigned within functions aren’t accessible outside that function.**Refactor** The process of converting ugly, repetitive code into clean code which follows D.R.Y. (Don’t Repeat Yourself) principles.