## Dictionaries

We use dictionaries to store key/value pairs.

```
customer = {
    "name": "John Smith",
    "age": 30,
    "is_verified": True
}
```

We can use strings or numbers to define keys. They should be unique. We can use any types for the values.

```
customer["name"]                # returns "John Smith"
customer["type"]                # throws an error
customer.get("type", "silver") # returns "silver"
customer["name"] = "new name"
```

## Functions

We use functions to break up our code into small chunks. These chunks are easier to read, understand and maintain. If there are bugs, it's easier to find bugs in a small chunk than the entire program. We can also re-use these chunks.

```
def greet_user(name):
    print(f"Hi {name}")


greet_user("John")
```

**Parameters** are placeholders for the data we can pass to functions. **Arguments** are the actual values we pass.

We have two types of arguments:

- Positional arguments: their position (order) matters
- Keyword arguments: position doesn't matter - we prefix them with the parameter name.

```
# Two positional arguments
greet_user("John", "Smith")

# Keyword arguments
calculate_total(order=50, shipping=5, tax=0.1)
```

Our functions can return values. If we don't use the return statement, by default **None** is returned. None is an object that represents the absence of a value.

```
def square(number):
    return number * number

result = square(2)
print(result)  # prints 4
```

## Exceptions

Exceptions are errors that crash our programs. They often happen because of bad input or programming errors. It's our job to anticipate and handle these exceptions to prevent our programs from cashing.

```
try:
    age = int(input('Age: '))
    income = 20000
    risk = income / age
    print(age)
except ValueError:
    print('Not a valid number')
except ZeroDivisionError:
    print('Age cannot be 0')
```

## Classes

We use classes to define new types.

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y
    def move(self):
        print("move")
```

When a function is part of a class, we refer to it as a **method**.

Classes define templates or blueprints for creating objects. An object is an instance of a class. Every time we create a new instance, that instance follows the structure we define using the class.

```
point1 = Point(10, 5)
point2 = Point(2, 4)
```

__init__ is a special method called constructor. It gets called at the time of creating new objects. We use it to initialize our objects.

## Inheritance

Inheritance is a technique to remove code duplication. We can create a *base class* to define the common methods and then have other classes inherit these methods.

```
class Mammal:
  def walk(self):
     print("walk")

class Dog(Mammal):
  def bark(self):
     print("bark")


dog = Dog()
dog.walk()   # inherited from Mammal
dog.bark()   # defined in Dog
```

## Modules

A module is a file with some Python code. We use modules to break up our program into multiple files. This way, our code will be better organized. We won't have one gigantic file with a million lines of code in it!

There are 2 ways to import modules: we can import the entire module, or specific objects in a module.

```
# importing the entire converters module
import converters
converters.kg_to_lbs(5)

# importing one function in the converters module
from converters import kg_to_lbs
kg_to_lbs(5)
```

## Packages

A package is a directory with **__init__.py** in it. It can contain one or more modules.

```
# importing the entire sales module
from ecommerce import sales
sales.calc_shipping()

# importing one function in the sales module
from ecommerce.sales import calc_shipping
calc_shipping()
```

## Python Standard Library

Python comes with a huge library of modules for performing common tasks such as sending emails, working with date/time, generating random values, etc.

### Random Module

```
import random

random.random()       # returns a float between 0 to 1
random.randint(1, 6)  # returns an int between 1 to 6

members = ['John', 'Bob', 'Mary']
leader = random.choice(members) # randomly picks an item
```

# Pypi

Python Package Index (pypi.org) is a directory of Python packages published by Python developers around the world. We use **pip** to install or uninstall these packages.

```
pip install openpyxl
```

```
pip uninstall openpyxl
```