In [ ]:

```
Built-in Data Types

In programming, data type is an important concept.

Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type:  str
Numeric Types:  int, float, complex
Sequence Types: list, tuple, range
Mapping Type:   dict
Set Types:  set, frozenset
Boolean Type:   bool
Binary Types:   bytes, bytearray, memoryview
```

In [5]:

```
x = 5j
print(type(x))
```

```
<class 'complex'>
```

In [7]:

```
x = "Hello World"

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
Hello World
<class 'str'>
```

In [3]:

```
x = 20

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
20
<class 'int'>
```

In [2]:

```
x = 20.0

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
20.0
<class 'float'>
```

In [11]:

```
x = 1j

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
1j
<class 'complex'>
```

In [13]:

```
x = ["apple", "banana", "cherry"]

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
['apple', 'banana', 'cherry']
<class 'list'>
```

In [7]:

```
x = ("apple", "banana", "cherry")

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
('apple', 'banana', 'cherry')
<class 'tuple'>
```

In [5]:

```
x = range(10)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
range(0, 10)
<class 'range'>
```

In [3]:

```
x = {"name" : "John", "age" : 36}

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
{'name': 'John', 'age': 36}
<class 'dict'>
```

In [10]:

```
x = {"apple", "banana", "cherry"}

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
{'apple', 'banana', 'cherry'}
<class 'set'>
```

In [11]:

```
x = frozenset({"apple", "banana", "cherry"})

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
frozenset({'apple', 'banana', 'cherry'})
<class 'frozenset'>
```

In [12]:

```
x = True

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
True
<class 'bool'>
```

In [13]:

```
x = b"Hello"

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
b'Hello'
<class 'bytes'>
```

In [14]:

```
x = bytearray(5)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
bytearray(b'\x00\x00\x00\x00\x00')
<class 'bytearray'>
```

In [15]:

```
x = memoryview(bytes(5))

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
<memory at 0x000001FE9FD7A340>
<class 'memoryview'>
```

In [14]:

```
x = str("Hello World")

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
Hello World
<class 'str'>
```

In [15]:

```
x = int(20)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
20
<class 'int'>
```

In [18]:

```
x = float(20.5)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
20.5
<class 'float'>
```

In [19]:

```
x = complex(1j)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
1j
<class 'complex'>
```

In [20]:

```
x = list(("apple", "banana", "cherry"))

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
['apple', 'banana', 'cherry']
<class 'list'>
```

In [21]:

```
x = tuple(("apple", "banana", "cherry"))

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
('apple', 'banana', 'cherry')
<class 'tuple'>
```

In [22]:

```
x = range(6)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
range(0, 6)
<class 'range'>
```

In [23]:

```
x = dict(name="John", age=36)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
{'name': 'John', 'age': 36}
<class 'dict'>
```

In [24]:

```
x = set(("apple", "banana", "cherry"))

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
{'apple', 'banana', 'cherry'}
<class 'set'>
```

In [25]:

```
x = frozenset(("apple", "banana", "cherry"))

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
frozenset({'apple', 'banana', 'cherry'})
<class 'frozenset'>
```

In [26]:

```
x = bool(5)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
True
<class 'bool'>
```

In [27]:

```
x = bytes(5)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
b'\x00\x00\x00\x00\x00'
<class 'bytes'>
```

In [28]:

```
x = bytearray(5)

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
bytearray(b'\x00\x00\x00\x00\x00')
<class 'bytearray'>
```

In [29]:

```
x = memoryview(bytes(5))

#display x:
print(x)

#display the data type of x:
print(type(x))
```

```
<memory at 0x000001FE9FD7A280>
<class 'memoryview'>
```

In [ ]:

```
Python Numbers

There are three numeric types in Python:

int
float
complex
```

In [30]:

```
x = 1
y = 2.8
z = 1j

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

In [ ]:

```
Int

Int, or integer, is a whole number, positive or negative, without decimals, of unlimited le
```

In [31]:

```
x = 1
y = 35656222554887711
z = -3255522

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'int'>
<class 'int'>
<class 'int'>
```

In [ ]:

```
Float

Float, or "floating point number" is a number, positive or negative, containing one or more
```

In [32]:

```
x = 1.10
y = 1.0
z = -35.59

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
```

In [33]:

```
x = 35e3
y = 12E4
z = -87.7e100

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'float'>
<class 'float'>
<class 'float'>
```

In [ ]:

```
Complex

Complex numbers are written with a "j" as the imaginary part:
```

In [34]:

```
x = 3+5j
y = 5j
z = -5j

print(type(x))
print(type(y))
print(type(z))
```

```
<class 'complex'>
<class 'complex'>
<class 'complex'>
```

In [ ]:

```
Type Conversion

You can convert from one type to another with the int(), float(), and complex() methods:
```

In [35]:

```
#convert from int to float:
x = float(1)

#convert from float to int:
y = int(2.8)

#convert from int to complex:
z = complex(x)

print(x)
print(y)
print(z)

print(type(x))
print(type(y))
print(type(z))
```

```
1.0
2
(1+0j)
<class 'float'>
<class 'int'>
<class 'complex'>
```

In [ ]:

```
Random Number

Python does not have a random() function to make a random number,
but Python has a built-in module called random that can be used to make random numbers:
```

In [36]:

```
import random

print(random.randrange(1, 10))
```

```
1
```

In [ ]:

```
Specify a Variable Type

There may be times when you want to specify a type on to a variable. This can be done with
Python is an object-orientated language, and as such it uses classes to define data types,
its primitive types.

Casting in python is therefore done using constructor functions:

int() - constructs an integer number from an integer literal, a float
literal (by removing all decimals), or a string literal (providing the string represents a
float() - constructs a float number from an integer literal, a float literal or a string
literal (providing the string represents a float or an integer)
str() - constructs a string from a wide variety of data types, including strings, integer
literals and float literals
```

In [37]:

```
x = int(1)
y = int(2.8)
z = int("3")
print(x)
print(y)
print(z)
```

```
1
2
3
```

In [38]:

```
x = float(1)
y = float(2.8)
z = float("3")
w = float("4.2")
print(x)
print(y)
print(z)
print(w)
```

```
1.0
2.8
3.0
4.2
```

In [39]:

```
x = str("s1")
y = str(2)
z = str(3.0)
print(x)
print(y)
print(z)
```

```
s1
2
3.0
```