---- Python practice sets ----

----------BASIC DATA TYPES ----------

Fundamental types

In [19]:

```
# integers
x = 1
type(x)
```

Out[19]:

int

In [20]:

```
# float
x = 1.0
type(x)
```

Out[20]:

float

In [21]:

```
# boolean
b1 = True
b2 = False

type(b1)
```

Out[21]:

bool

In [22]:

```
# complex numbers: note the use of `j` to specify the imaginary part
x = 1.0 - 1.0j
type(x)
```

Out[22]:

complex

In [23]:

```
print(x)
```

(1-1j)

In [24]:

```
print(x.real, x.imag)
```

(1.0, -1.0)

Type utility functions

The module types contains a number of type name definitions that can be used to test if variables are of certain types:

In [25]:

```
import types

# print all types defined in the `types` module
print(dir(types))
```

['BooleanType', 'BufferType', 'BuiltinFunctionType', 'BuiltinMethodType', 'ClassType', 'CodeType', 'ComplexType', 'DictProxyType', 'DictType', 'DictionaryType', 'EllipsisType', 'FileType', 'FloatType', 'FrameType', 'FunctionType', 'GeneratorType', 'GetSetDescriptorType', 'InstanceType', 'IntType', 'LambdaType', 'ListType', 'LongType', 'MemberDescriptorType', 'MethodType', 'ModuleType', 'NoneType', 'NotImplementedType', 'ObjectType',

'SliceType', 'StringType', 'StringTypes', 'TracebackType', 'TupleType', 'TypeType', 'UnboundMethodType', 'UnicodeType', 'XRangeType', '__all__', '__builtins__', '__doc__', '__file__', '__name__', '__package__']

In [26]:

```
x = 1.0

# check if the variable x is a float
type(x) is float
```

Out[26]:

True

In [27]:

```
# check if the variable x is an int
type(x) is int
```

Out[27]:

False

We can also use the isinstance method for testing types of variables:

In [28]:

```
isinstance(x, float)
```

Out[28]:

True

Type casting

In [29]:

```
x = 1.5

print(x, type(x))
```

(1.5, <type 'float'>)

In [30]:

```
x = int(x)

print(x, type(x))
```

(1, <type 'int'>)

In [31]:

```
z = complex(x)

print(z, type(z))
```

((1+0j), <type 'complex'>)

In [32]:

```
x = float(z)
```

TypeError Traceback (most recent call last)

<ipython-input-32-e719cc7b3e96> in <module>()

----> 1 x = float(z)

TypeError: can't convert complex to float

Complex variables cannot be cast to floats or integers. We need to use z.real or z.imag to extract the part of the complex number we want:

In [33]:

```
y = bool(z.real)
```

```
print(z.real, " -> ", y, type(y))

y = bool(z.imag)

print(z.imag, " -> ", y, type(y))
```

```
(1.0, ' -> ', True, <type 'bool'>)
(0.0, ' -> ', False, <type 'bool'>)
```

Operators and comparisons

Most operators and comparisons in Python work as one would expect:

- Arithmetic operators +, -, *, /, // (integer division), '**' power

In [34]:

```
1 + 2, 1 - 2, 1 * 2, 1 / 2
```

Out[34]:

```
(3, -1, 2, 0)
```

In [35]:

```
1.0 + 2.0, 1.0 - 2.0, 1.0 * 2.0, 1.0 / 2.0
```

Out[35]:

```
(3.0, -1.0, 2.0, 0.5)
```

In [36]:

```
# Integer division of float numbers
3.0 // 2.0
```

Out[36]:

```
1.0
```

In [37]:

```
# Note! The power operators in python isn't ^, but **
2 ** 2
```

Out[37]:

```
4
```

Note: The / operator always performs a floating point division in Python 3.x. This is not true in Python 2.x, where the result of / is always an integer if the operands are integers. to be more specific, 1/2 = 0.5 (float) in Python 3.x, and 1/2 = 0 (int) in Python 2.x (but 1.0/2 = 0.5 in Python 2.x).

- The boolean operators are spelled out as the words and, not, or.

In [38]:

```
True and False
```

Out[38]:

```
False
```

In [39]:

```
not False
```

Out[39]:

```
True
```

In [40]:

```
True or False
```

Out[40]:

True

- Comparison operators >, <, >= (greater or equal), <= (less or equal), == equality, is identical.

In [41]:

```
2 > 1, 2 < 1
```

Out[41]:

(True, False)

In [42]:

```
2 > 2, 2 < 2
```

Out[42]:

(False, False)

In [43]:

```
2 >= 2, 2 <= 2
```

Out[43]:

(True, True)

In [44]:

```
# equality
[1,2] == [1,2]
```

Out[44]:

True

In [45]:

```
# objects identical?
l1 = l2 = [1,2]

l1 is l2
```

Out[45]:

True

----------# raw_input_example.py----------

```
name = raw_input("What is your name? ")
city = raw_input("What city do you live in? ")
state = raw_input("What state is that in? ")

print "Hello there! It is so great to meet you,"
# One way to do this is to print strings on separate lines
print name
print "from"
print city
print state

# We can also "glue together" pieces of a string by adding commas
#  between them.
print name, "from", city, state
```

```
# This doesn't work because raw_input returns a string
#age = raw_input("Pardon my rudeness, but how old are you? ")
#print "Wow! You look like you could be", age - (0.15*age), "!!"

age = input("Pardon my rudeness, but how old are you? ")

# Notice that we can "glue together" two strings and one integer into
#  one giant string.
print "Wow! You look like you could be", int(age - (0.15*age)), "!!"

# int(argument) forces the argument to be an integer by rounding down.
# So, int(5.1) = 5 and int(5.9) = 5
```

--- practicing if else conditions----

```
# conditional_examples.py

# Boolean math

# Examples of if statements
# General format:
# if <condition is True>:
#     <code to execute if condition is True>
if 9 > 5:
   print "Yes, 9 greater than 5"

if 9 != 5:
   print "Yes, 9 not equal to 5"

# An example of an if/else statemet
# General format:
# if <condition is True>:
#     <code to execute if condition is True>
# else:
#     <code to execute if condition is False>

#if 9 < 5:
#    print "Yes, 9 less than 5"
#else:
#    print "No, 9 is not less than 5"


# An example using "not" and "and" keywords
#if not (10 == 4) and 9 > 5:
#    print "Yay, basic math competency achieved!"
#else:
#    print ":("
```

```
# Traffic light example
#light_color = raw_input("What color is the traffic light? ")
#light_color = light_color.lower()
#print light_color
#if light_color == "red":
#    print "You should stop"
#elif light_color == "yellow":
#    print "Slow down!"
#elif light_color == "green":
#    print "Go ahead!"
#else:
#    print "What country are you in??"
```

PYTHON PROBLEM:

Create a program that asks the user to enter their name and their age. Print out a message addressed to them that tells them the year that they will turn 100 years old.