Lecture 2

Today: variables, types, defining your own functions, if-else

Variables

How to use variables in Python.

In [1]:
```
# this is how you assign a variables
x = 2
y = 3
```

In [2]:
```
x+y
```

Out[2]: 5

In [3]:
```
x**2 + x + 1
```

Out[3]: 7

In [4]:
```
# you can re-assign and lose the old value
x = 3
```

In [5]:
```
x+y
```

Out[5]: 6

In [6]:
```
# variables can hold strings as well
strng = "will print"
```

In [7]:
```
# by the way, print is a handy function, if you want to print more tha
n one value
print("hellp")
print(x)
print("print " + strng, 1, 2, x, x+1)
```

```
hellp
3
print will print 1 2 3 4
```

Types

Every variable (and every expression has a type)

In [8]:
```
# take a look at this
x = 2/2
print(x)
y = 1
print(y)
```

```
1.0
1
```

what's going on? It's because 2/2 has a different type than 1.

In [9]:
```
# division by integer has float type
type(2/2)
```

Out[9]:
```
float
```

In [10]:
```
type(1.0)
```

Out[10]:
```
float
```

In [11]:
```
type(1)
```

Out[11]:
```
int
```

In [12]:
```
# let's look at some other types
type("hellooo")
```

Out[12]:
```
str
```

In [13]:
```
# even functions have a type
type(abs)
```

Out[13]:
```
builtin_function_or_method
```

In [14]:
```
import math
type(math.cos)
```

Out[14]:
```
builtin_function_or_method
```

In [15]:
```
# even modules have type
type(math)
```

Out[15]:
```
module
```

types are important in programming. you must **always** think about the types of the objects you are working with.

Defining functions

In [16]:
```
def f(x):
    return x*x
```

In [17]:
```
f(3)
```

Out[17]: 9

In [18]:
```
# what is this going to do?
def g(x,y):
    print("One day, I will pass the Turing test.")
    z = x + y
    return z
    z = 1/0        # we never get to this line
    return z+1     # we never get to this line
```

In [19]:
```
g(1,2)
```

```
One day, I will pass the Turing test.
```

Out[19]: 3

Every line inside the function is executed until return is called

Local vs global variables

Some things to be careful about when working with functions

In [20]:
```
# What's going to happen?
def f(x,y):
    zzz = 0   # local variable
    return x+y
```

In [21]:
```
f(2,1)
```

Out[21]:
```
3
```

In [22]:
```
print(zzz)    # error because zzz is a local variable, not known outsi
de the function
```

```
---------------------------------------------------------------------
-------
NameError                                 Traceback (most recent cal
l last)
<ipython-input-22-dbb4ee89e69d> in <module>()
----> 1 print(zzz)    # error because zzz is a local variable, not k
nown outside the function

NameError: name 'zzz' is not defined
```

We didn't do the following during the lecture, but it's good to put here

In [23]:
```
ggg = 1
def f(x,y):
    ggg = 1000
    return x+y

print(f(1,2),ggg)
```

```
3 1
```

So ggg's value didn't change. Why? It was because ggg inside the function is a new, local variable. Not the global one.

Fix: use the `global` keyword

In [24]:
```
ggg = 1
def f(x,y):
    global ggg
    ggg = 1000
    return x+y

print(f(1,2),ggg)
```

```
3 1000
```

In [ ]:
```

```