(https://KimTein.github.io)

Ch_3_Designing and Using Functions

In [1]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

Functions That Python Provides

In [5]:
```
# Python comes with many built-in functions that perform common operation
abs(-9)

print("#"*30)

abs(3.3)

print("#" * 30)

# General form of a function call
day_temperature = 3
night_temperature = 10
abs(day_temperature - night_temperature)

print("#" * 30)

# Examples

abs(-7) + abs(3.3)

print("#" * 30)

pow(abs(-2), round(4.3))

print("#" * 30)

int(34.6) # Some of the most useful built-in functions are ones that conv

print("#" * 30)

int(-4.3) # We see that when a floating-point number is converted to an i

print("#" * 30)

float(21)

print("#" * 30)

help(abs)

print("#" * 30)

# Examples
help(pow)

print("#" * 30)

pow(2, 4) # This call calculates 2^4.

print("#" * 30)

pow(2, 4, 3) # We know that 2^4 is 16, and evaluation of 16 % 3 produces
```

Out[5]:
```
9
##############################
```

```
Out[5]: 3.3
        ##############################
Out[5]: 7
        ##############################
Out[5]: 10.3
        ##############################
Out[5]: 16
        ##############################
Out[5]: 34
        ##############################
Out[5]: -4
        ##############################
Out[5]: 21.0
        ##############################
        Help on built-in function abs in module builtins:

        abs(x, /)
            Return the absolute value of the argument.

        ##############################
        Help on built-in function pow in module builtins:

        pow(base, exp, mod=None)
            Equivalent to base**exp with 2 arguments or base**exp % mod with 3 a
        rguments

            Some types, such as ints, are able to use a more efficient algorithm
        when
            invoked using the three argument form.

        ##############################
Out[5]: 16
        ##############################
Out[5]: 1
```

Memory Addresses: How Python Keeps Track of values

In [8]:
```
# You can discover the actual memory address of an object using built-in
help(id)

print("#" * 30)

id(-9)

print("#" * 30)

id(23.1)

print("#" * 30)

shoe_size = 8.5
id(shoe_size)

print("#" * 30)

fahrenheit = 77.7
id(fahrenheit)

print("#" * 30)
# Function objects also have memory addresses.

id(abs)

print("#" * 30)

id(round)

print("#" * 30)
```

```
Help on built-in function id in module builtins:

id(obj, /)
    Return the identity of an object.

    This is guaranteed to be unique among simultaneously existing object
s.
    (CPython uses the object's memory address.)

##############################
```

Out[8]:
```
4408793232
##############################
4408311216
##############################
4408792464
##############################
4408311280
##############################
4346972416
##############################
```

Out[8]: 4346979408

##############################

Defining Our Own Functions

In [10]:
```
# Let's write our own function that convert Fahrenheit to Celsius.
convert_to_celsius(212) # The function convert_to_celsius doesn't exist y
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call la
st)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_as
signment.ipynb Cell 9 in <cell line: 2>()
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_assignment.ipynb#X23sZmlsZQ%3D%3D?line
=0'>1</a> # Let's write our own function that convert Fahrenheit to Cels
ius.
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_assignment.ipynb#X23sZmlsZQ%3D%3D?line
=1'>2</a> convert_to_celsius(212)

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

In [11]:
```
# General form
def convert_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5 / 9

convert_to_celsius(80)
```

Out[11]: 26.666666666666668

Using Local Variables For Temporary Storage

In [13]:
```
# Some computations are complex, and breaking them down into separate ste
def quadratic(a, b, c, x):
    first = a * x ** 2
    second = b * x
    third = c
    return first + second + third

quadratic(2, 3, 4, 0.5)

print("#" * 30)

quadratic(2, 3, 4, 1.5)

print("#" * 30)

quadratic(2, 3, 4, 1.3)

print("#" * 30)
```

Out[13]: 6.0

```
##############################
```

Out[13]: 13.0

```
##############################
```

Out[13]: 11.280000000000001

```
##############################
```

In [14]:
```
# Trying to access a local variable from outside the function is an error
first
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb Cell 13 in <cell line: 2>()
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb#X31sZmlsZQ%3D%3D?line=0'>1</a> # Trying to access a local variable from outside the function is an error, just like trying to access a variable that has never been defined is an error.
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb#X31sZmlsZQ%3D%3D?line=1'>2</a> first

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

In [16]:
```
# Trying to access a local variable from outside the function is an error
a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb Cell 14 in <cell line: 2>()
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb#X32sZmlsZQ%3D%3D?line=0'>1</a> # Trying to access a local variable from outside the function is an error, just like trying to access a variable that has never been defined is an error.
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb#X32sZmlsZQ%3D%3D?line=1'>2</a> a

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

In [21]:
```
# The area of a program that a variable can be used in is called the vari
def quadratic(a, b, c, x):
    first = a * x ** 2
    second = b * x
    third = c
    return first + second + third

quadratic(1, 2, 3) # If a function is defined to take a certain number of
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb Cell 15 in <cell line: 8>()
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb#X33sZmlsZQ%3D%3D?line=4'>5</a>     third = c
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb#X33sZmlsZQ%3D%3D?line=5'>6</a>     return first + second + third
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignment.ipynb#X33sZmlsZQ%3D%3D?line=7'>8</a> quadratic(1, 2, 3)

TypeError: quadratic() missing 1 required positional argument: 'x'
```

Tracing the Function Calls in the Memory Model

In [24]:
```python
# The x that is a parameter of function f is a different variable than th
def f(x):
    x = 2 * x
    return x

# Whenever Python executes a function call, it creates a namespace (liter
x = 1
x = f(x + 1) + f(x + 2)
x
```

Out[24]: 10

Designing New Functions: A Recipe

In [30]:
```python
# Python uses three double quotes to start and end this documentation; ev

def days_differnce(day1: int, day2: int) -> int: # The parameter types an
    """
    Return the number of days between day1 and day2, which are
    both in the range 1-365 (thus indicating the day of the
    year).

    >>> days_difference(200, 224)
    24
    >>> days_difference(50, 50)
    0
    >>> days_difference(100, 99)
    -1
    """
    return day2 - day1
```

Designing Three Birthday-Related Functions

Day of the Week	Number
Sunday	1
Monday	2
Tuesday	3
Wednesday	4
Thursday	5
Friday	6
Saturday	7

In [35]:
```python
# Range from 1 to 7
def get_weekday(current_weekday: int, days_ahead: int) -> int:
    return current_weekday + days_ahead % 7

get_weekday(3, 1)
get_weekday(6, 1)
get_weekday(7, 1)

print("#" * 30)

# Range from 0 to 6
# Sunday -> 0, Saturday -> 6

def get_weekday(current_weekday: int, days_ahead: int) -> int:
    return (current_weekday + days_ahead - 1) % 7 + 1

get_weekday(3, 1)
get_weekday(6, 1)
get_weekday(7, 1)
```

Out[35]: 4

Out[35]: 7

Out[35]: 8

```
##############################
```

Out[35]: 4

Out[35]: 7

Out[35]: 1

What Day Is My Birthday On?

In [38]:
```python
def get_birthday_weekday(current_weekday: int, current_day: int, birthday
    days_diff = days_differnce(current_day, birthday_day)
    return get_weekday(current_weekday, days_diff)

get_birthday_weekday(5, 3, 4)
get_birthday_weekday(5, 3, 116)
get_birthday_weekday(6, 116, 3)
```

Out[38]: 6

Out[38]: 6

Out[38]: 5

Omitting a return Statement:None

In [40]:
```
# If you don't have a return statement in a function, nothing is produced
def f(x):
    x = 2 * x

res = f(3)
res

print("#" * 30)

print(res)

print("#" * 30)

id(res)

print("#" * 30)

# Variable res has a value: it's None! And None has a memory address. If

def f(x):
    x = 2 * x
    return None

print(f(3))
```

```
##############################
None
##############################
```

Out[40]: 4344743376

```
##############################
None
```

Dealing with Situations That Your Code Doesn't Handle

In [43]:
```
# You'll often write a function that works only in some situations.
def pie_percent(n: int) -> int:
    return int(100 / n)

pie_percent(5)
pie_percent(2)
pie_percent(1)
```

Out[43]: 20

Out[43]: 50

Out[43]: 100

Ch_4_Working with text

Creating Strings of Characters

The opening and closing quotes must match.

In [4]:
```
'Aristotle'
"Issac Newton"
```

Out[4]: 'Aristotle'

Out[4]: 'Issac Newton'

In [5]:
```
# The error indicates that the end of the line was reached before the end
'Charles Darwin"
```

```
  Input In [5]
    'Charles Darwin"
                    ^
SyntaxError: EOL while scanning string literal
```

Operations on Strings

In [6]:
```
# Python has a built-in function, len, that returns the number of charact
len('Albert Einstein')
print("#"*30)

len('123!')
print("#"*30)

len(' ')
print("#"*30)

len('')
```

Out[6]: 15

##############################

Out[6]: 4

##############################

Out[6]: 1

##############################

Out[6]: 0

In [8]:
```
# We can add two strings using the + operator, which produces a new strin
# When + has two string operands, it is referred to as the concatenation
'Albert' + ' Einstein'
```

Out[8]:
```
'Albert Einstein'
```

In [9]:
```
# Adding an empty string to another string produces a new string that is
"Alan Turing" + ''
print("#"*30)

"" + 'Grace Hopper'
```

Out[9]:
```
'Alan Turing'
##############################
'Grace Hopper'
```

In [10]:
```
# Because the first operand was a string, Python expected the second oper
'NH' + 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call la
st)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_4_
assignment.ipynb Cell 38 in <cell line: 2>()
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_4_assignment.ipynb#X60sZmlsZQ%3D%3D?li
ne=0'>1</a> # Because the first operand was a string, Python expected th
e second operand to also be a string but instead it was an integer.
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_4_assignment.ipynb#X60sZmlsZQ%3D%3D?li
ne=1'>2</a> 'NH' + 3

TypeError: can only concatenate str (not "int") to str
```

In [11]:
```
# Because Python saw a 9 first, it expected the second operand to also be
9 + 'planets'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call la
st)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_4_
assignment.ipynb Cell 39 in <cell line: 2>()
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_4_assignment.ipynb#X61sZmlsZQ%3D%3D?li
ne=0'>1</a> # Because Python saw a 9 first, it expected the second opera
nd to also be numeric.
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_4_assignment.ipynb#X61sZmlsZQ%3D%3D?li
ne=1'>2</a> 9 + 'planets'

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

In [12]:
```
# If you want to join a string with a number, you could apply function st
'Four score and ' + str(7) + ' years ago'
```

Out[12]:
```
'Four score and 7 years ago'
```

In [13]:
```
# Function int can be applied to a string whose contents look like an int
int('0')
print("#" * 30)

int("11")
print("#" * 30)

int('-324')
print("#" * 30)

float('-324')
print("#" * 30)

float("56.34")
```

Out[13]:
```
0
##############################
11
##############################
-324
##############################
-324.0
##############################
56.34
```

In [14]:
```
# It isn’t always possible to get an integer or a floating- point represe
int('a')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call la
st)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_4_
assignment.ipynb Cell 42 in <cell line: 2>()
      <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_4_assignment.ipynb#X64sZmlsZQ%3D%3D?li
ne=0'>1</a> # It isn’t always possible to get an integer or a floating-
point representation of a string.
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics
_programming/assignment/Ch_3/ch_3_4_assignment.ipynb#X64sZmlsZQ%3D%3D?li
ne=1'>2</a> int('a')

ValueError: invalid literal for int() with base 10: 'a'
```

In [15]:
```
float('b')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_4_assignment.ipynb Cell 43 in <cell line: 1>()
----> <a href='vscode-notebook-cell:/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_4_assignment.ipynb#X65sZmlsZQ%3D%3D?line=0'>1</a> float('b')

ValueError: could not convert string to float: 'b'
```

In [16]:
```
# In addition to +, len, int, and float, operator * can be applied to str
'AT' * 5
print("#"*30)

4 * '-'
print("#"*30)

'Gc' * 0
print("#"*30)

'TATATA' * -3
```

Out[16]:
```
'ATATATATAT'
##############################
'----'
##############################
''
##############################
''
```

In [17]:
```
# Strings are values, so you can assign a string to a variable.
sequence = 'ATTGTCCCCC'
len(sequence)
print("#"*30)

new_sequence = sequence + 'GGCCTCCTGC'
new_sequence
print("#"*30)

new_sequence * 2
```

Out[17]:
```
10
##############################
'ATTGTCCCCCGGCCTCCTGC'
##############################
'ATTGTCCCCCGGCCTCCTGCATTGTCCCCCGGCCTCCTGC'
```

Using Special Characters in Strings

In [18]:
```
# Suppose you want to put a single quote inside a string. If you write it
# When Python encounters the second quote—the one that is intended to be
# It doesn't know what to do with the text that comes after the second qu
'that's not to going to work'
```

```
  Input In [18]
    'that's not to going to work'
          ^
SyntaxError: invalid syntax
```

In [19]:
```
# One simple way to fix this is to use double quotes around the string
"that's better"
print("#"*30)

# We can also put single quotes around a string containing a double quote
'She said, "That is better."'
```

Out[19]:
```
"that's better"
```

```
##############################
```

Out[19]:
```
'She said, "That is better."'
```

In [20]:
```
# The backslash is called an escape character, and the combination of the
# The name comes from the fact that we're "escaping" from Python's usual
'She said, "That' + "'" + 's hard to read."'
```

Out[20]:
```
'She said, "That\'s hard to read."'
```

In [21]:
```
# The escape sequence \' is indicated using two symbols, but those two sy
len('\'')
print("#" * 30)

len('it\'s')
```

Out[21]:
```
1
```

```
##############################
```

Out[21]:
```
4
```

In [22]:
```
# The backslash is called an escape character, and the combination of the
# The name comes from the fact that we're "escaping" from Python's usual
'She said, "That' + "'" + 's hard to read."'
```

Out[22]:
```
'She said, "That\'s hard to read."'
```

Creating Multiline String

In [24]:
```
# If you create a string using single or double quotes, the whole string
'one
```

```
  Input In [24]
    'one
       ^
SyntaxError: EOL while scanning string literal
```

In [26]:
```
# To span multiple lines, put three single quotes or three double quotes
# The string can then span as many lines as you want.
'''one
two
three'''
```

Out[26]:
```
'one\ntwo\nthree'
```

Notice that the string Python creates contains a \n sequence everywhere our input started a new line. Each newline is a character in the string.

Printing Information

Built-in function print

In [28]:
```
# Function print doesn't allow any styling of the output: no colors, no i
print(1 + 1)
print("The Latin 'Oryctolagus cuniculus' means 'domestic rabbit'.\n")

print('In 1859, Charles Darwin revolutionized biology')
print('and our understanding of ourselves')
print('by publishing "On the Origin of species".\n')

print('one\ttwo\nthree\tfour')
```

```
2
The Latin 'Oryctolagus cuniculus' means 'domestic rabbit'.

In 1859, Charles Darwin revolutionized biology
and our understanding of ourselves
by publishing "On the Origin of species".

one     two
three   four
```

In [29]:
```
numbers = '''one
two
three'''
numbers
print("#" * 30)

print(numbers) # When a multiline string is printed, those \n sequences a
print("#" * 30)

print(1, 2, 3) # Function print takes a comma-separated list of values an
print("#" * 30)

print() # Print ends the current line, advancing to the next one.
```

Out[29]:
```
'one\ntwo\nthree'
```

```
##############################
one
two
three
##############################
1 2 3
##############################
```

In [30]:
```
# Function print can print values of any type, and it can even print valu
print(1, 'two', 'three', 4.0)
```

```
1 two three 4.0
```

In [31]:
```
# It is possible to call print with an expression as an argument.
radius = 5
print("The diameter of the circle is", radius * 2, "cm.")
```

```
The diameter of the circle is 10 cm.
```

In [32]:
```
# The parameters sep, end, file, and flush have assignment statements in
# These are called default parameter values: by default, if we call funct
help(print)
```

```
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdo
ut.
    sep:   string inserted between values, default a space.
    end:   string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.
```

In [37]:
```
# We separate each value with a comma and a space instead of just a space
print('a', 'b', 'c' '\n')
print('a', 'b', 'c', sep=',')

print('\n')
print('a', 'b', 'c', sep=',', end='')
```

```
a b c

a,b,c


a,b,c
```

In [38]:
```
def convert_to_celsius(fahrenheit: float) -> float:
    """ Return the number of Celsius degrees equivalent to fahrenheit deg
    >>> convert_to_celsius(75)
    23.888888888888889
    """

    return (fahrenheit - 32.0) * 5.0 / 9.0

print('80, 78.8 and 10.4 degrees Fahrenheit are equal to ', end='')
print(convert_to_celsius(80), end=', \n')
print(convert_to_celsius(78.8), end=', and ')
print(convert_to_celsius(10.4), end=' Celsius. \n')
```

```
80, 78.8 and 10.4 degrees Fahrenheit are equal to 26.666666666666668,
26.0, and -12.0 Celsius.
```

Getting Information from the Keyboard

In [39]:
```
# Another built-in function is input, which reads a single line of text f
# It returns whatever the user enters as a string, even if it looks like
species = input()
species
print()

population = input()
population
print()

type(population)
print()

species = input("Please enter a species: ")
print(species)
```

In [41]:
```
# If you are expecting the user to enter a number, you must use int or fl
population = input()
population
```

Out[41]:
```
'Homo sapiens'
```

In [51]:
```
population = input(population)
population
```

Out[51]:
```
'6973738433'
```

In [44]:
```
type(population)
```

Out[44]:
```
str
```

In [50]:
```
population = int(input())
population = population + 1
population
```

Out[50]:
```
6973738434
```

Reference

- Title: Physics Programming Lecture Note (INU)
- Author: Jeongwoo Kim, Ph.D.
- Availability: https://sites.google.com/view/jeongwookim
