ch_3_assignment

March 7, 2023

0.1 ch_3_assignment

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

0.1.1 Functions That Python Provides

[ ]:
```
# Python comes with many built-in functions that perform common operations.
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 convert␣
 ↪from one type to another.
```

```
print("#" * 30)

int(-4.3) # We see that when a floating-point number is converted to an␣
 ↪integer, it is truncated, not rounded.

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 1.
```

[ ]: 9

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

[ ]: 3.3

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

[ ]: 7

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

[ ]: 10.3

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

[ ]: 16

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

[ ]: 34

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

[ ]:
```
-4
```

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

[ ]:
```
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 arguments

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

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

[ ]:
```
16
```

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

[ ]:
```
1
```

0.1.2 Memory Addresses: How Python Keeps Track of values

[ ]:
```
# You can discover the actual memory address of an object using built-in␣
  ↪function id.
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 objects.
    (CPython uses the object's memory address.)

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

[ ]:
```
4408793232

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

[ ]:
```
4408311216

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

[ ]:
```
4408792464

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

[ ]:
```
4408311280

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

[ ]:
```
4346972416

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

[ ]:
```
4346979408

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

0.1.3 Defining Our Own Functions

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

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignmen .
 ↪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␣
 ↪Celsius.
----> <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
```

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

convert_to_celsius(80)
```

[ ]:
```
26.666666666666668
```

0.1.4 Using Local Variables For Temporary Storage

[ ]:
```
# Some computations are complex, and breaking them down into separate steps can␣
 ↪lead to clearer code.
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)
```

[ ]:
```
6.0

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

13.0

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

11.280000000000001

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

[ ]:
```
# 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.
first
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignmen .
 ↪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 i␣
 ↪an error, just like trying to access a variable that has never been defined i␣
 ↪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
```

[ ]:
```
# 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
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignmen .
 ↪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 i
 ↪an error, just like trying to access a variable that has never been defined i
 ↪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
```

[ ]:
```
# The area of a program that a variable can be used in is called the variable's
 ↪scope. The scope of a local variable is from the line in which it is defined
 ↪up until the end of the function.
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
 ↪parameters, a call on that function must have the same number of arguments
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/Users/Kim_Tein/INU/inu_data/physics_programming/assignment/Ch_3/ch_3_assignmen
 ↪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'
```

0.1.5 Tracing the Function Calls in the Memory Model

[ ]:
```
# The x that is a parameter of function f is a different variable than the x in
 ↪the shell.
def f(x):
    x = 2 * x
    return x
```

```
# Whenever Python executes a function call, it creates a namespace (literally, a space for names) in which to store local variables for that call.
x = 1
x = f(x + 1) + f(x + 2)
x
```

[ ]: 10

0.1.6 Designing New Functions: A Recipe

[ ]:
```
# Python uses three double quotes to start and end this documentation; everything in between is meant for humans to read. This notation is called a docstring, which is short for documentation string.

def days_differnce(day1: int, day2: int) -> int: # The parameter types and return type form a type contract because we are claiming that if you call this function with the right types of values, we'll give you back the right type of value.
    """
    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

[ ]:
```
# 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)
```

[ ]: 4

[ ]: 7

[ ]: 8

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

[ ]: 4

[ ]: 7

[ ]: 1

What Day Is My Birthday On?

```
[ ]: def get_birthday_weekday(current_weekday: int, current_day: int, birthday_day:␣
      ↪int) -> int:
        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)
```

[ ]: 6

[ ]: 6

[ ]: 5

0.1.7 Omitting a return Statement:None

[ ]:
```
# 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 you␣
 ↪don't have a return statement in your function, your function will return␣
 ↪None.

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

print(f(3))
```

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

[ ]:
```
4344743376

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

0.1.8 Dealing with Situations That Your Code Doesn't Handle

[ ]:
```
# 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)
```

```
[ ]: 20
```

```
[ ]: 50
```

```
[ ]: 100
```

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

---