Quiz 0 Review Session

Content on Quiz 0

- Objects & data types
- Expressions
- Functions
- Memory diagrams

Disclaimer: We haven’t seen the quiz; this review session covers the main topics in the unit.

Objects and Data Types

Data Types

- Data Types
  - `float` (decimal, e.g. `2.0`)
  - `int` (whole number, e.g. `2`)
  - `str` (string of characters, e.g. `"Hello"`)
  - `bool` (evaluates to True or False, e.g, `True`)
- Check the type of a value
  - `type()`
- Change the type - also known as **casting**
  - `str(), float(), int()`

Practice with Types

1. What is the **type** of the following expression? `float(str(1.5))`
2. What is the difference between an `int` and `float`?
3. What is the type of the following expression? `"1"`
4. What would `type(False)` evaluate to?

1. `float`
2. A int is an integer with no decimal. A float is a floating point number, and has a decimal point
3. `str`
4. `bool`

Indexing/Subscription Notation

- **Subscription –** using square brackets [ ] to get one item in a *sequence*
- Using subscription to access an item is called **indexing**
- Indexing in Python starts at **0**

`str` is a *sequence*, so you can use subscription to index a string and get one letter

ex: `"COMP110"[0]` would evaluate to `"C"`

```
"C O M P 1 1 0"
 0 1 2 3 4 5 6
```

Practice

1. What would `"happy"[1]` evaluate to?
2. What is the value of `len("happy")`?
3. Evaluate `"pie"[len("pie") - 1]`.
4. What would `"happy"[5]` evaluate to?

1. `"a"`
2. 5
3. `"e"`
4. Error - the last letter of "happy" is at index 4

Expressions

Expressions

- An **expression** evaluates to a value of some *type*
- Numerical operators
  - Mathematical operations
- Relational operators
  - Always result in a True or False boolean value

Numerical Operators

Symbol	Operator Name	Example
`**`	Exponentiation	`2 ** 8` equivalent to 2^8
`*`	Multiplication	`10 * 3`
`/`	Division	`7 / 5` result is `1.4`
`//`	Integer Division	`7 // 5` result is `1`
`%`	Remainder “modulo”	`7 % 5` result is `2`
`+`	Addition	`1 + 1`
`-`	Subtraction	`111 - 1`
`-`	Negation	`-(1 + 1)` result is `-2`

Order Of Operations

- P ()
- E **
- MD * / %
- AS + -
- Tie? Evaluate *Left to Right*

Relational Operators

Operator Symbol	Verbalization	`True` Ex.	`False` Ex.
`==`	Is equal to?	`1 == 1`	`1 == 2`
`!=`	Is NOT equal to?	`1 != 2`	`1 != 1`
`>`	Is greater than?	`1 > 0`	`0 > 1`
`>=`	Is at least?	`1 >= 0` or `1 >= 1`	`0 >= 1`
`<`	Is less than?	`0 < 1`	`1 < 0`
`<=`	Is at most?	`0 <= 1` or `1 <= 1`	`1 <= 0`

Order Of Operations

- P ()
- E **
- MD * / %
- AS + -
- Tie? Evaluate *Left to Right*

Relational Operators should be evaluated last, simplify both sides first!

Practice

1. `2 ** 3`
2. `13 % 5`
3. `3 / 1.5`
4. `7 == 7`
5. `9 <= 2 + 2 * 3`
6. `12 > int(4.0) * int(3.0)`
7. What would be the result of the following expression? `20 + "20"`

1. `8`
2. `3`
3. `2.0`
4. `True`
5. `False`
6. `False`
7. TypeError - `20` is an `int`, and `"20"` is a `str`. The types of these values do not match, so we cannot perform a numerical operation on them

Functions

Function Definitions

- A function **definition** is made up of function **signature** and function **body**
- The **signature** tells you how the function should be used
  - function's name
  - what (if any) parameters it has
  - what type of value will be returned
- The **body** of a function specifies steps that will be followed when the function is called
  - **return** statements are special statements that tell the computer to stop evaluating the function, and return back to the place the function was called

```
def name_of_function(parameter: type) -> returnType:
    """Docstring."""
    return expression_of_returnType
```

Function Calls

- A function **call** is made up of the function name, and arguments for each parameter of the function definition

Given the following *function definition:*

```
def int_plus_four(input_number: int) -> int:
    """Docstring."""
    return input_number + 4
```

An example matching *function call* would be:

```
int_plus_four(input_number=5)
```

Functions

- Make sure to know the difference between *defining* and *calling* a function
  - defining a function: writing out the "recipe", specifying how the function should work, but not actually evaluating it
  - calling a function: cooking from the "recipe", actually evaluating the code of the function definition
- `return` vs `print`
  - return:
    - sends you immediately out of a function, finishing the function
    - used for your computer to send the result of a function back to where the function was called
  - print:
    - outputs a value for us humans to see
  - return and print are not the same!

```python
"""A program defining two functions."""

def print_hello() -> None:
    """Outputs hello"""
    print("Hello")

def difference(a: int, b: int) -> int:
    """Returns the difference between two integers."""
    return a - b

print(difference(a=4, b=1))
```

Identify line number(s) with the following:

1. Docstring
2. Function calls
3. Return statement
4. Function definition
5. Function signature

1. 1, 4, 8
2. 5, 11
3. 9
4. 3 - 5, 7 - 9
5. 3, 7

```python
"""A program defining two functions."""

def print_hello() -> None:
    """Outputs hello"""
    print("Hello")

def difference(a: int, b: int) -> int:
    """Returns the difference between two integers."""
    return a - b
```

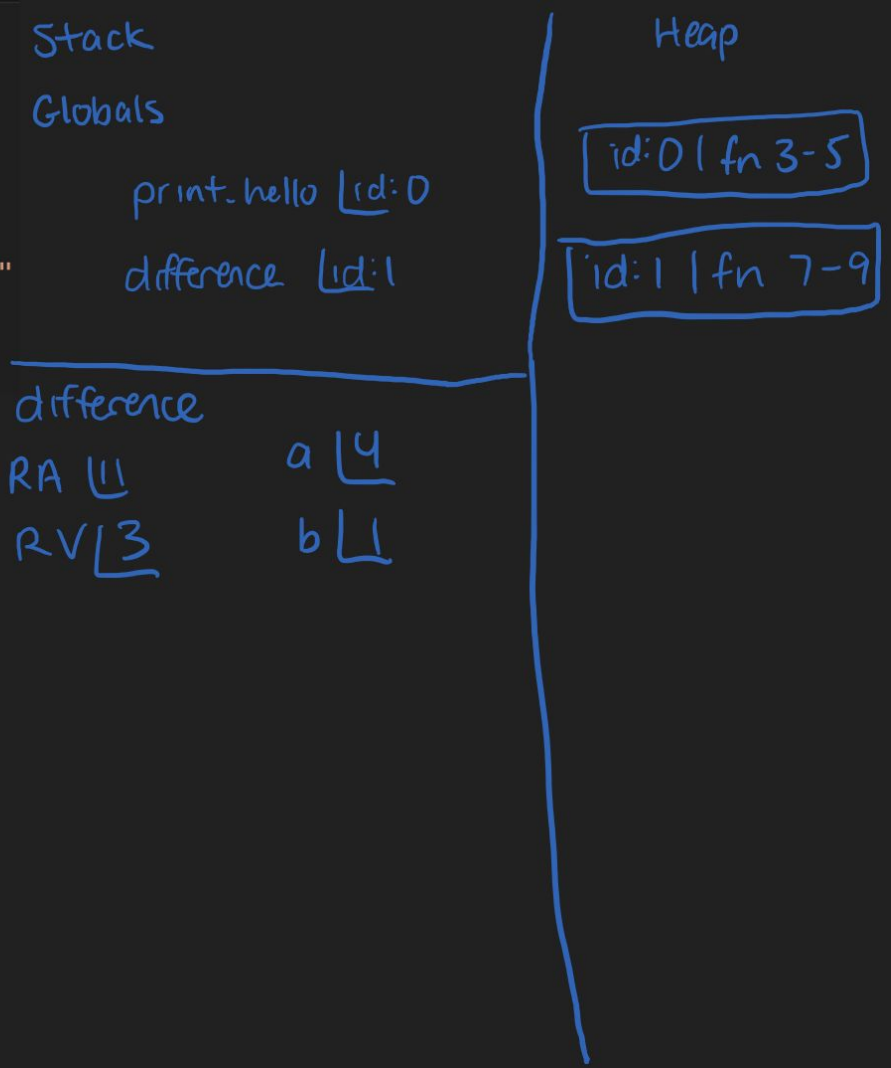
1. What is returned from the following function call? 5
   `difference(a=10, b=5)`
2. What would be the printed output of the following function call? "Hello"
   `print_hello()`
3. Does the `print_hello()` function have any parameters? No
4. What is returned by print_hello()? None

```python
"""A program defining two functions."""

def print_hello() -> None:
    """Outputs hello"""
    print("Hello")

def difference(a: int, b: int) -> int:
    """Returns the difference between two integers."""
    return a - b

print(difference(a=4, b=1))
```

```python
"""A program defining two functions."""

def print_hello() -> None:
    """Outputs hello"""
    print("Hello")

def difference(a: int, b: int) -> int:
    """Returns the difference between two integers."""
    return a - b

print(difference(a=4, b=1))
```

Output

3



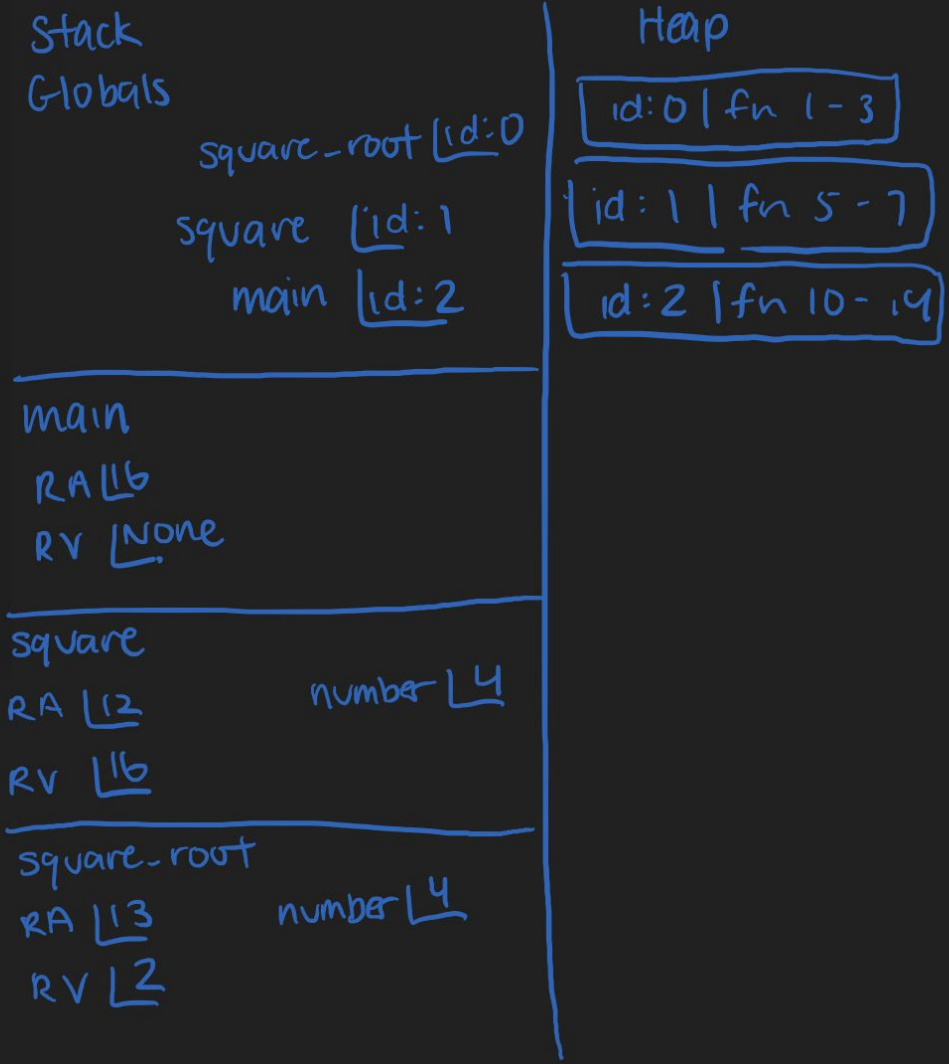
```python
def square_root(number: int) -> float:
    """Calculate the square root of a number."""
    return number ** 0.5

def square(number: int) -> int:
    """Calculate the square of a number."""
    return number ** 2


def main() -> None:
    """Entrypoint of the program"""
    print(square(number=4))
    print(square_root(number=4))
    return None

main()
```

```python
def square_root(number: int) -> float:
    """Calculate the square root of a number."""
    return number ** 0.5

def square(number: int) -> int:
    """Calculate the square of a number."""
    return number ** 2


def main() -> None:
    """Entrypoint of the program"""
    print(square(number=4))
    print(square_root(number=4))
    return None

main()
```

output

16

2



Code Writing Example

Write a function definition for a function named `amount_of_snow` that takes in one `int` parameter named `hours_snowed` and returns an `float` value representing the total amount of snow in inches. We can assumed that snow falls at an average of **0.9 inches per hour**.

Make sure to add a descriptive docstring describing the function!

Write a call to the function.

```
1  def amount_of_snow(hours_snowed: int) -> float:
2      """Calculates total inches of snow."""
3      return hours_snowed * 0.9
4
5  amount_of_snow(hours_snowed=3)
```

Questions?

Other Resources!

- Practice quiz on the course site with answers and explanations
  - We would recommend trying the problems out on your own, then checking your answers
- Tutoring
  - Thursday 3 - 5 in FB 141
- Office Hours
  - Tomorrow and Friday 11 - 5 in SN008