CS1010S Tutorial 1

Sean Ng

AY2018/19 Sem 2, Week 3
Updated 2019-01-24 at 09:55:39

Introduction

About Me

- Year 2, Computer Science

About Me

- Year 2, Computer Science
- Took CS1010X (CS1010S + C programming + CS students)

About Me

- Year 2, Computer Science
- Took CS1010X (CS1010S + C programming + CS students)
- ...still use python for interviews, projects

About Me

- Year 2, Computer Science
- Took CS1010X (CS1010S + C programming + CS students)
- ...still use python for interviews, projects

Slides and/or tutorial materials at `pengnam.github.io/1010S/`

Contact me at `seanngpengnam@u.nus.edu`

About The Tutorial

- Attendance is taken (about 200xp)

About The Tutorial

- Attendance is taken (about 200xp)
- Participation is encouraged

About The Tutorial

- Attendance is taken (about 200xp)
- Participation is encouraged
- Some questions may be skipped

About The Tutorial

- Attendance is taken (about 200xp)
- Participation is encouraged
- Some questions may be skipped
- Lectures/recitations are ‘canonical’ over tutorial

Feedback From Coursemology

Functions: How Do They Work?

```
import math

def get_hyp(opp, adj):
    return math.sqrt(opp**2 + adj**2)

x = 4
y = 3
hyp = get_hyp(x, y)
```

Functions: How Do They Work?

1. `get_hyp(x, y)`

Functions: How Do They Work?

1. `get_hyp(x, y)`
2. `get_hyp(4, 3)`

Functions: How Do They Work?

1. `get_hyp(x, y)`
2. `get_hyp(4, 3)`

3.
   ```
   def get_hyp(opp, adj):
       return math.sqrt(opp**2 + adj**2)
   ```

Functions: How Do They Work?

1. `get_hyp(x, y)`
2. `get_hyp(4, 3)`

3.
   ```
   def get_hyp(opp, adj):
       return math.sqrt(opp**2 + adj**2)
   ```

4.
   ```
   def get_hyp(4, 3):
       return math.sqrt(4**2 + 3**2)
   ```

Functions: How Do They Work?

1. `get_hyp(x, y)`
2. `get_hyp(4, 3)`

3.
   ```
   def get_hyp(opp, adj):
       return math.sqrt(opp**2 + adj**2)
   ```

4.
   ```
   def get_hyp(4, 3):
       return math.sqrt(4**2 + 3**2)
   ```

5.
   ```
   def get_hyp(4, 3):
       return math.sqrt(16 + 9)
   ```

Functions: How Do They Work?

1. `get_hyp(x, y)`
2. `get_hyp(4, 3)`

3.
   ```
   def get_hyp(opp, adj):
       return math.sqrt(opp**2 + adj**2)
   ```

4.
   ```
   def get_hyp(4, 3):
       return math.sqrt(4**2 + 3**2)
   ```

5.
   ```
   def get_hyp(4, 3):
       return math.sqrt(16 + 9)
   ```

6.
   ```
   def get_hyp(4, 3):
       return math.sqrt(25)
   ```

Functions: How Do They Work?

1. `get_hyp(x, y)`
2. `get_hyp(4, 3)`

3.
   ```
   def get_hyp(opp, adj):
       return math.sqrt(opp**2 + adj**2)
   ```

4.
   ```
   def get_hyp(4, 3):
       return math.sqrt(4**2 + 3**2)
   ```

5.
   ```
   def get_hyp(4, 3):
       return math.sqrt(16 + 9)
   ```

6.
   ```
   def get_hyp(4, 3):
       return math.sqrt(25)
   ```

7.
   ```
   def get_hyp(4, 3):
       return 5
   ```

Functions: How Do They Work?

1. `get_hyp(x, y)`
2. `get_hyp(4, 3)`

3.
   ```
   def get_hyp(opp, adj):
       return math.sqrt(opp**2 + adj**2)
   ```

4.
   ```
   def get_hyp(4, 3):
       return math.sqrt(4**2 + 3**2)
   ```

5.
   ```
   def get_hyp(4, 3):
       return math.sqrt(16 + 9)
   ```

6.
   ```
   def get_hyp(4, 3):
       return math.sqrt(25)
   ```

7.
   ```
   def get_hyp(4, 3):
       return 5
   ```
8. 5

Functions: Quiz

Program A:

```
monthly_cost = 300
months = 12

def multiply(monthly_cost, months):
    return monthly_cost * months

yearly_cost = multiply(monthly_cost, months)
```

Program B:

```
monthly_cost = 300
months = 12

def multiply(x, y):
    return x * y

yearly_cost = multiply(monthly_cost, months)
```

Functions: Quiz

```
a, b = 12, 8

def sum(a, b):
    return a + b

a, b = 4, 6
sum(1, 1)
```

The output of this program is...

1. 20
2. 10
3. 2

Tutorial

Tutorial Question 2

First, using if-else, define a function `odd(x)` that returns *True* when its integer argument is an odd number and *False* otherwise.

Now, without using if-else, define the function `new_odd(x)` that does the same.

Tutorial Question 3

Write a function that will return the number of digits in an integer. You can safely assume that the integers are non-negative and will not begin with the number 0 other than the integer 0 itself.

Tutorial Question 4

Define a function that takes three numbers as arguments and returns the sum of the squares of the two larger numbers.

Tutorial Question 5

Write a function `is_leap_year` that takes one integer parameter and decides whether it corresponds to a leap year, i.e. the `is_leap_year` returns `True` if the input parameter is true, and `False` otherwise.