|  |  |
| --- | --- |
|  | ***Review*** |

Answer the following questions, if you’re not sure or even don’t remember, revisit our videos, refer to our book or ask your instructors or your TAs:

Remember that there isn’t only one right answer, only good ones :)

* Why should we use functions at all?

Creating a new function gives us an opportunity to name a group of statements. Functions can simplify a program by hiding a complex computation behind a single command. The function (including its name) can capture our mental chunking, or abstraction, of the problem.  
Creating a new function can make a program smaller by eliminating repetitive code

* How to define/declare a function?

**def** NAME( PARAMETERS ):  
STATEMENTS

A header line which begins with a keyword and ends with a colon.  
2. A body consisting of one or more Python statements, each indented the same amount — *the Python style guide recommends 4 spaces* — from the header line.

* How to call/use a function?

Function calls contain the name of the function being executed followed by a list of values, called arguments, which are assigned to the parameters in the function definition.

* What is return, why and how do we use it?
* Do we have to use return in **every** function?
* What are function arguments/parameters, why and how we use it?
* How to use function from a different file other than our currently working file?

Note: Function **arguments** are sometimes **also** called function **parameters**

Reference:

* [How to think like a computer scientist](http://www.ict.ru.ac.za/Resources/cspw/thinkcspy3/thinkcspy3.pdf), chapter 4
* [Python cheat sheet](https://www.dropbox.com/s/qgws0sammkb6fi6/1.%20beginners_python_cheat_sheet_pcc_all.pdf?dl=0)

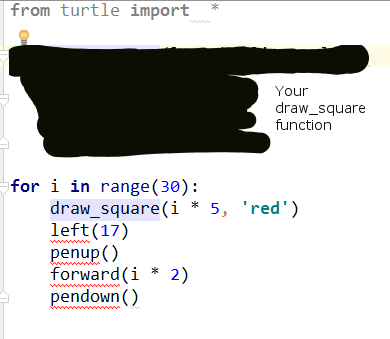
|  |  |
| --- | --- |
| *http://www.bestappsforkids.com/wp-content/uploads/2012/04/save-turtle.png* | ***Turtle exercise*** |

1. Write a function that prints out “Hello world” 3 times (note: no arguments, no return)
2. Write a function that takes **2 numbers as arguments** and print out sum of them (note: has arguments, no return)
3. Write a Python function that **draws a square**, named draw\_square, takes **2 arguments**: length and color, where length is the length of its side and color is the color of its bound (line color)
4. Now, another programmer named ‘T.Anh’ will use your code in exercise 1. He writes as follows:

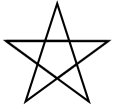
**for** i **in** range(30):  
 **draw\_square**(i \* 5, **'red'**)  
 left(17)  
 penup()  
 forward(i \* 2)  
 pendown()

Copy this code into your editor, run the whole program and see what it draws:

*Note: If your code does not run, try not to modify* T.Anh*’s code, modify your function instead*



1. Write a Python function that draws a star, named draw\_star, take 3 parameters: x, y, and length. Where x, y are the location of the star, length is the length of its side

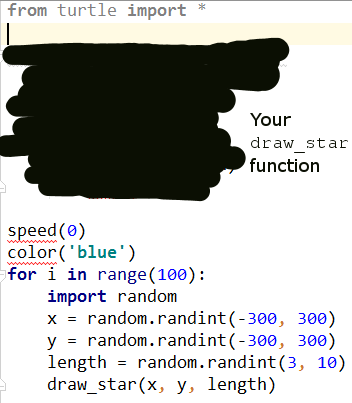


*Hint: Turn 144 degree at each point, Google ‘python 3 turtle go to position’*

1. Again, your function will be used by other programmers, they write as follows:

speed(0)  
color(**'blue'**)  
**for** i **in** range(100):  
 **import** random  
 x = random.randint(-300, 300)  
 y = random.randint(-300, 300)  
 length = random.randint(3, 10)  
 draw\_star(x, y, length)

Copy this code into your editor, run the whole program and see what it draws:



|  |  |
| --- | --- |
| http://images.8tracks.com/cover/i/001/358/131/7357.original-3735.jpg?rect=0,29,289,289&q=98&fm=jpg&fit=max&w=100&h=100 | ***Serious exercise*** |

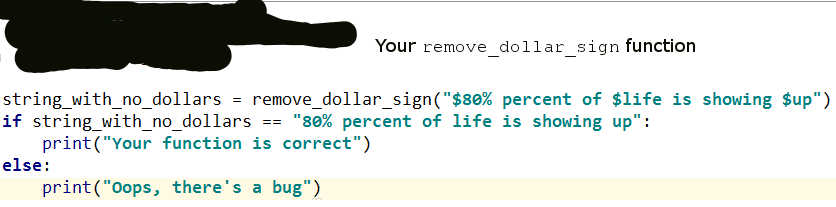
1. Write a function that removes the dollar sign (“$”) in a string, named remove\_dollar\_sign, takes 1 arguments: s, where s is the input string, **returns** the new string with no dollar sign in it

*Hint: Google “Python string replace remove”*

1. Now, another programmer named Hiep will use your code in exercise 3. He writes as follows:

string\_with\_no\_dollars = remove\_dollar\_sign(**"$80% percent of $life is to show $up"**)  
**if** string\_with\_no\_dollars == **"80% percent of life is to show up"**:  
 print(**"Your function is correct"**)  
**else**:  
 print(**"Oops, there's a bug"**)

Copy this code into your editor, run the whole program and see what it prints out:



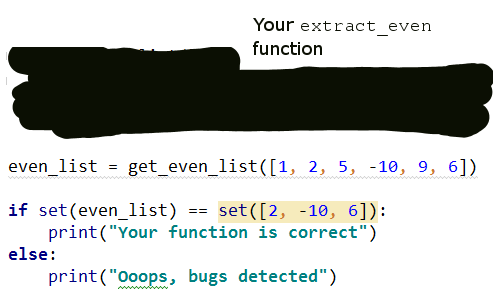
If it prints out **“Your function is correct”**, we’re good

If it prints out **“Oops, there’s a bug”**, you might want to come back and check your function

1. Write a function that extracts the even items in a given integer list, named get\_even\_list, takes 1 parameter: l, where l is the given integer list ([1, 4, 5, -1, 10] for example), returns a new list contains only even numbers ([4, 10] if the given list is [1,4,5,-1,10])
2. Let’s take your function to the test. The tester will write his/her test code as follows:

even\_list = get\_even\_list([1, 2, 5, -10, 9, 6])  
  
**if** set(even\_list) == set([2, -10, 6]):  
 print(**"Your function is correct"**)  
**else**:  
 print(**"Ooops, bugs detected"**)

Copy this code into your editor, run the whole program and see what it prints out:



If it prints out **“Your function is correct”**, we’re good

If it prints out **“Oops, bugs detected”**, you might want to come back and check your function

*Note:* ***set*** *is an unordered data structure, meaning set of (1, 2,3) equals set of (3, 1, 2)*

1. Write a function named is\_inside that checks if a point is inside a rectangle, takes 2 parameters, the first is a list with 2 elements respectively represents x and y coordinates of the given point, the second is a list with 4 elements respectively represents x, y coordinates and width height of the given rectangle

For example:

is\_inside([100, 120], [140, 60, 100, 200])

should return False

**140**

**60**

**width = 100**

**height = 200**

X

Y

**120**

**100**

and

is\_inside([200, 120], [140, 60, 100, 200])

should return True

**140**

**60**

**width = 100**

**height = 200**

X

Y

**120**

**200**

1. (**Optional**) Write test cases (as we did in exercices 8 and 10) to check if your is\_inside function is correct

12. (**Optional**) Download [this starter code](https://drive.google.com/open?id=1MmTXikygr4_IStHwySGYXYmdDSYI3TJE) and unzip it. Inside you will find the BackColor Game with UI and without core logic.

You can try running it by typing “python app.py”.

Now Open file backcolor.py and have a look at the shapes data structure, what type of data is this, dictionary, list or both?

If you don’t know what ‘#C62828’ is, scroll to the last page

Inside an item of shapes is the information about a colored rectangle drawn in the UI:

(Note: the **text** is not shown here but in the quiz)

**#C62828**

**x = 140**

**y = 60**

**width = 100**

**height = 100**

X

Y

Your job is to write the generate\_quiz and mouse\_press (though you can play a little bit, but you SHOULD NOT touch get\_shapes() function) to serve the following functionality:

generate\_quiz()

Each time the UI developer wants to generate quiz, she will call generate\_quiz and expect a list in return, this list, must contain 3 elements (in this order):

text: The text to be shown to users

color: The color of the shown text

quiz\_type:

**0** if users must select one the rectangles by the **Meaning** of the text as the answer

**1** if users must select one the rectangles by the **Color** of the text as the answer

mouse\_press()

Each time users click on the screen, the UI developer will call mouse\_press to check if users have just answered correctly

x, y: The coordinates of the position that users have just clicked

text: The Quiz’s text being shown to users

color: The Quiz’s color being shown to users

quiz\_type: The current Quiz Type - users should select by Meaning or the Color of the Text

The return value must be True or False, which represents users’ correctness

To understand the functions’ arguments better, you should print them out, run and play with the UI before go for the coding

#C62828 is hex color, Google: “color picker” and try to move the sidebar left and right:

Drag this left and right