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

* Why should we use functions at all
* It helps save times as 1 function can be used several times after you declare it. In addition, by declaring function separately <normally in a different file>, our code will be much more easier to understand, to control or even to plan.
* How to define/declare a function?
* Functionname (arguments/parameters) {statement}
* How to call/use a function?
* Use function name and give it arguments.
* What is return, why and how do we use it?
* The return statement ends function execution and specifies a value to be returned to the function caller.
* Do we have to use return in **every** function?
* No.
* What are function arguments/parameters, why and how we use it?
* + Function **parameters** are the **names** listed in the function definition.

+ Function **arguments** are the real **values** passed to (and received by) the function.

* How to use function from a different file other than our currently working file?
* Load functions from all file by “window.onload”

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

|  |  |
| --- | --- |
| *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)

function print(){

for(i=0;i<3;i++){

console.log("hello work");}

}

1. Write a function that takes **2 numbers as arguments** and print out sum of them (note: has arguments, no return)

function add(x,y){

let sum = x+y;

console.log(sum);

}

add(5,6);

1. 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)

function drawSquare(length,colr){

color(colr)

for(i=0;i<4;i++){

fd(length)

rt(90)

}

}

//drawSquare(100,"red") test

1. Now, another programmer named ‘T.Anh’ will use your code in exercise 1. He writes as follows:

**for** (let i = 0; i < 30; i ++){

**drawSquare**(i \* 5, **'red'**)

lt(17)

penup()  
 forward(i \* 2)  
 pendown()

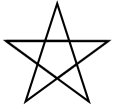
}

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*

* https://turtle-js.herokuapp.com/?turtle=jgppbp6XZW

1. Write a Python function that draws a star, named draw\_star, take 1 parameters: length



* <https://turtle-js.herokuapp.com/?turtle=fsRDCkupGV>

|  |  |
| --- | --- |
| 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 “Javascript 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"){**  
 console.log(**"Your function is correct"**)}  
**else**{  
 console.log(**"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]
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, 9, -10, 6])  
  
**if** (even\_list == [2, -10, 6]){  
 print(**"Your function is correct"**)};  
**else**{  
 console.log(**"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, ])

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