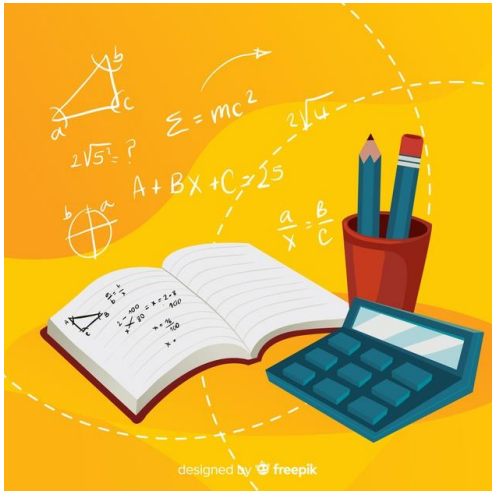


School Nova

Computer Science

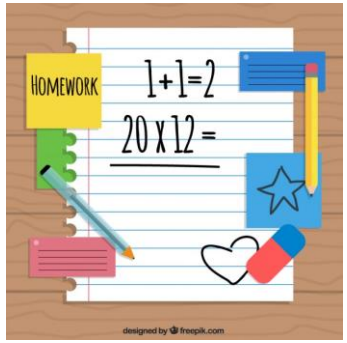
CS 201

User-defined functions: Part 2

1/19/2020

By Oleg Smirnov

Homework, tasks 1 - 4



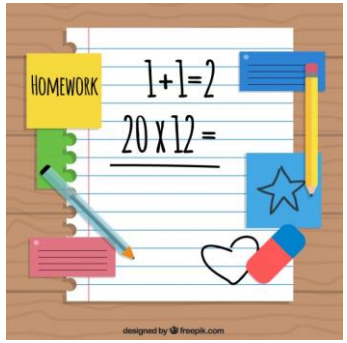

```
def print_product(a, b):
    print(f"The product of {a} and {b} is {a * b}.")

def calc_product(a, b):
    return(a * b)

def dict_from_lists(a, b):
    return(dict(zip(a, b)))

def safe_division(a, b):
    if (type(a)==int or type(a)==float) and (type(b)==int or type(b)==float) \
    and b != 0:
        return(a/b)
    else:
        return("NAN")
```

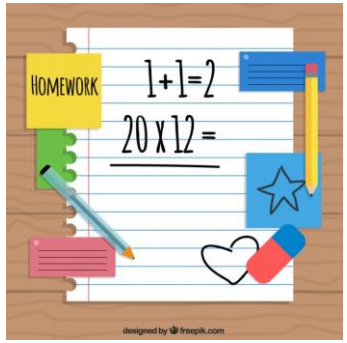
Homework, task 5




```
def int_input():

    while True:
        usernum = input("Please, enter an integer: ")
        if usernum == "quit" or usernum == "exit":
            usernum = "NAN"
            break
        try:
            usernum = int(usernum)
            break
        except:
            print("Not an integer")
            continue
    return(usernum)
```

Homework, tasks 6 - 8




```
def listval(*x):
    return(list(x))

def uniques(*x):
    return(list(set(x)))

import datetime
def day():
    return(int(str(datetime.date.today())[-2:]))
```

User-defined functions (more arguments)

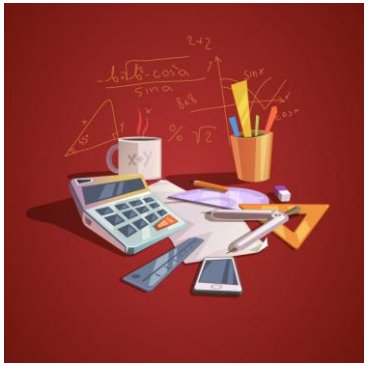
```
# this function returns two separate objects
def sumprod(x, y):
    return(x + y, x * y)

a, b = sumprod(3, 4)
print(a)                                    7
print(b)                                    12
```

```
# this function accepts ***keyworded arguments*** and returns a dictionary
def create_dictionary(**x):
    return(x)

capitals = create_dictionary(France = “Paris”, Italy = “Rome”) # using the function
print(capitals)
>>> {'France': 'Paris', 'Italy': 'Rome’}
```

User-defined functions (more arguments)



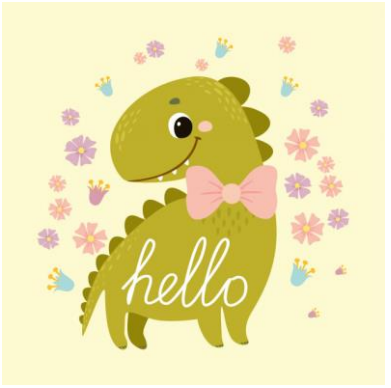
this function accepts ***one argument*** and then ***any number of arguments***
and it ***returns a list***

```
def power_list(power, *values):
    mylist = []
    for i in values:
        mylist.append(i**power)
    return(mylist)

a = power_list(2, 6, 7, 8, 9)
print(a)
```

```
>>> [36, 49, 64, 81]
```

User-defined functions default values




```
def greet(name = "John"):
    print(f"Hello, {name}!")

greet(“Alex")
greet()
```

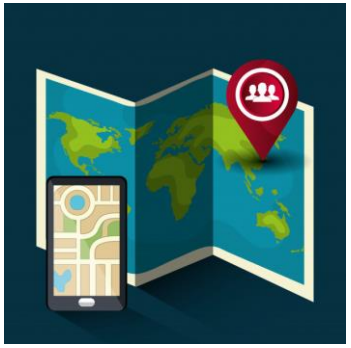
```
Hello, Alex!
Hello, John!
```

```
def sum(a = 3, b = 7):
    return(a + b)

print(sum())
print(sum(10, 20))
print(sum(10))
print(sum(b = 10))
```

```
10
30
17
13
```

User-defined functions, local and global variables 1



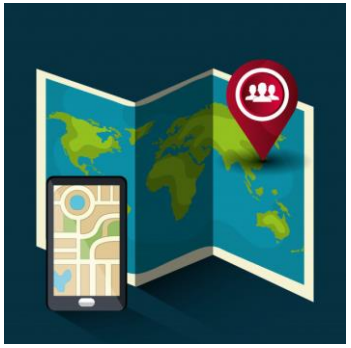
```
def secret_sum(a):
    b = 100
    return(a + b)

x = secret_sum(33)
#print(b)                    # NameError: name 'b' is not defined

def secret_sum2(a):
    global b
    b = 100
    return(a + b)

x = secret_sum2(33)
print(b)                     100
```

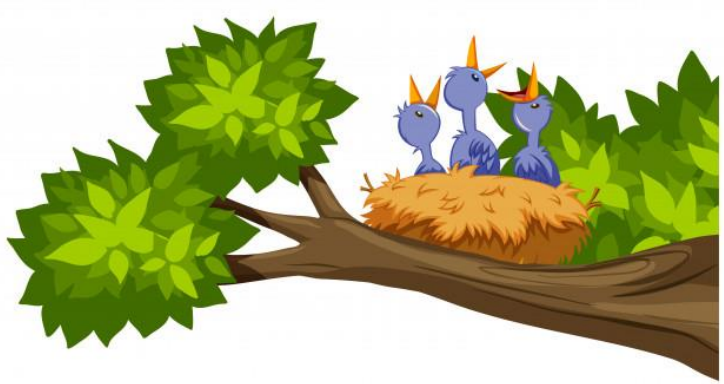
User-defined functions, local and global variables 2



y = 20
 	
def mysum(x):
  y = 10
  return(x + y)
 	
print(mysum(5))
print(y)
	y = 20
	
	def mysum(x):
	  global y
	  y = 10
	  return(x + y)
	
	print(mysum(5))
	print(y)

15 20	15 10

User-defined functions, nested functions



```
def safe_div(a, b):

    def is_a_num(x):
        return(type(x)==int or type(x)==float)

    if is_a_num(a) and is_a_num(b) and b != 0:
        return(a/b)
    else:
        return("NAN")
```

You cannot use is_a_num() outside of save_div()
Only safe_div() can use is_a_num()

If you need to use is_a_num() both inside AND outside of safe_div(), it should be a separate, not nested, function.