Further on Python

Day9_FurtherPy.md

LAST TIME TOPICS

by Nathan Hailu

Topics

- Functions,recursion
- lambda → Map/filter
- OOP & POP
- Class & objects
- user-built Module

Functions

- A function is a block of code that performs a specific task.
- Suppose, you need to create a program to create a blue circle.
- You can create two functions to solve this problem:
  a. A function that create a circle function
  b. A function that create a color function
- Dividing a complex problem into smaller chunks makes our program easy to understand and reuse.

Types of function

- There are two types of function in Python programming:
-
  - **Standard library functions** - These are built-in functions in Python that are available to use.
    - Almost all keywords are functions
    - print(),len(),input()....
  - **User-defined functions** - We can create our own functions based on our requirements.

Creating Functions

Syntax:

```
def function_name(arguments):
    # function body
```

It have body like if else statements also have body.

```
def greet():
    print('Hello World!')
```

Example:

- Functions have to be called to word. Thing functions like some skilled person, plumbers can do water pipe problems, their specific task is fixing broken pipes. So, if i need to fix my plumber ... right! | => SO python functions also have to be called.

```
# call the function
greet()
```

Cont...

- We have 2 functions
  - gtst
  - Nathan
- Both do different tasks.
- We have called their name so they are on the output.

```python
def gtst():
    print("learn.")
    print("DO exercise.")
    print("Ask questions.")


def Nathan():
    print("Teach.")
    print("Answer Questions.")
    print("Prepare modules.")


Nathan()
gtst()
```

```
# Output: Teach.
#         Answer Questions.
#         Prepare modules.
#         Learn.
#         DO exercise.
#         Ask questions.
```

Function Arguments

- They are used to take value while calling and insert it inside the function.
-

```python
# function with two arguments
def add_numbers(num1, num2):
    sum = num1 + num2
    print('Sum: ',sum)
```

```python
def add_numbers(num1=1,num2=1):
```

```python
# function call with two values
add_numbers(5, 4)
```

```python
# Output: Sum: 9
```

Cont…

```python
def users(fname, lname):
    print(f"Hello {fname}!, your father name is: {lname}")


users('Nathan','Hailu')


# Output: Hello Nathan!, your father name is: Hailu
```

The function takes fname & name as a variable for the function.

```python
def display(number1):
    print(f'the value u entered is: {number1}')


user_Input = input("Enter number: ")
display(user_Input)


# Output: Enter number: 23
#         the value u entered is: 23
```

Return statement

```python
def add_numbers():
    ...
    return sum
```

- A Python function may or may not return a value.
- If we want our function to return some value to a function call, we use the 'return' statement.

```python
def add(number1,number2):
    return number1+number2


add(2,3)
# Output: NOTHING...
```

```python
def add(number1,number2):
    return number1+number2


print(add(2,3))
# Output: 5
```

```python
def display(number1):
    return f'the value u entered is: {number1}'


user_Input = input("Enter number: ")


print(display(user_Input))
```

```python
def add(number1,number2):
    return number1+number2


sum = add(2,3)
print(sum)
# Output: 5
```

Cont...

- As i told you every things are functions on python
  - So print() is function also input() is len(), everything
  - When we do `print(hello)` we are calling the function and giving it an argument.
- You can give default values too

```python
def display(number1=100):
    print(f'the value u entered is: {number1}')


display()


# Output: 100
```

Quiz coding.

Write a code that can replace this pow() keyword. It accepts 2 arguments a number and its power. It u do pow(4,2) => it gives 16 , means the power.

```
print(pow(5,2))


# Output: 25
```

1) Create A function
2) Accept to arguments
3) Do a calculation to power
4) Return the value
5) Call the function and print it.

Recursion

- Recursion is process of defining something in terms of itself.
  - In Python, we know that a function can call other functions. It is even possible for the function to call itself. These types of construct are termed as recursive functions.

```python
def recurse():
    ...
    recurse()        recursive call
    ...

recurse()
```

```python
def factorial(x):
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))


num = 3
print("The factorial of", num, "is", factorial(num))
```

The factorial of 3 is 6
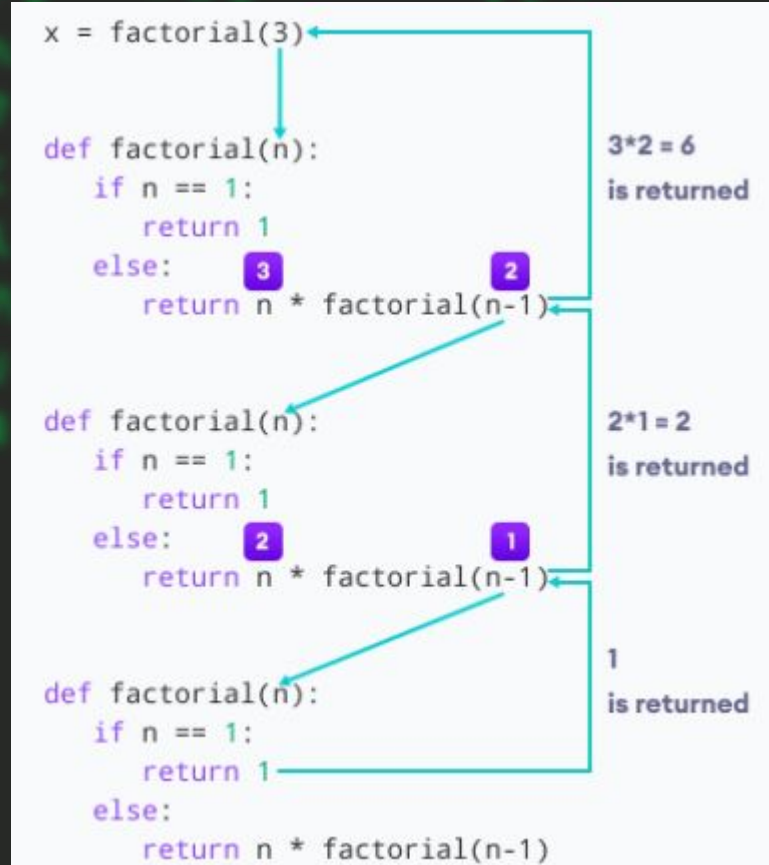
Cont...

- So, This helps to call the factorial in

```
factorial(3)            # 1st call with 3
3 * factorial(2)        # 2nd call with 2
3 * 2 * factorial(1)    # 3rd call with 1
3 * 2 * 1               # return from 3rd call as number=1
3 * 2                   # return from 2nd call
6                       # return from 1st call
```

Advantages of Recursion

1. Recursive functions make the code look clean and elegant.
2. A complex task can be broken down into simpler sub-problems using recursion.
3. Sequence generation is easier with recursion than using some nested iteration.

```
x = factorial(3)

def factorial(n):
    if n == 1:
        return 1
    else:  [3]        [2]
        return n * factorial(n-1)


def factorial(n):
    if n == 1:
        return 1
    else:  [2]        [1]
        return n * factorial(n-1)


def factorial(n):
    if n == 1:
        return 1
    else:
        return n * factorial(n-1)
```

3*2 = 6
is returned

2*1 = 2
is returned

1
is returned

Disadvantages of Recursion

1. Sometimes the logic behind recursion is hard to follow through.

2. Recursive calls are expensive (inefficient) as they take up a lot of memory and time.

3. Recursive functions are hard to debug.

Exercise 1

```
Enter Your name: Nathan
Enter Your Age: 20
==========================
Hello Nathan!
You are 20 years old.
You was Born in 2003
```

1) Create A function That accepts username and age, then display the name and the age.
   a) BONUS: try to also display, the year which the user born
      i) HINT: use 2023 as the year now.
2) Create 4 Functions add,sub,multi,div and those functions return the result of two arguments num1, num2. And accept 2 numbers user and do the calculation and display:

```
Enter the 1st number: 4
Enter the 2nd number: 2
==========================
The sum of 4 and 2 is: 6
The difference of 4 and 2 is: 2
The product of 4 and 2 is: 8
The quotient of 4 and 2 is: 2
```

Anonymous / lambda Function

- If function doesnt have name it is called lambda function / Anonymous
- If you have 1 line code to return you dont need to def a function,
- a lambda function is a special type of function without the function name.
- Syntax: `lambda argument(s) : expression`
- We use lambda keyword instead of def

```
1   def greet():
2       return "Hello World"
3
4   print(greet())
```

```
def numbers(a,b):
    return a+b

print(numbers(2,3))
```

```
greet = lambda : print('Hello World')
```

```
# call the lambda
greet()
```

```
numbers = Lambda a,b : a+b
print(numbers(2,3))
```

Function takers function

- Filter, map & reduce takes a function as an argument.
-

```
filter()
(class) filter(__function: None, __iterable: Iterable[Any | None], /)

filter(__function: (_S@__init__) -> TypeGuard, __iterable:
Iterable[_S@__init__], /)

filter(__function: (Any) -> Any, __iterable: Iterable, /)
```

filter(function or None, iterable) --> filter object

Return an iterator yielding those items of iterable for which function(item) is true. If function is None, return the items that are true.

Filter Function

- Filters are used to filter or search some function from sequences.

```python
def is_even (n):
    return n%2==0

nums=[3,2,6,8,4,6,2,91]

evens= list(filter(is_even, nums))

print (evens)
# Output: <filter object at 0x00000136BA2569E0>
    # Output: [2, 6, 8, 4, 6, 2]
```

```python
nums=[3,2,6,8,4,6,2,91]

evens= list(filter(lambda n:n%2==0, nums))

print (evens)
# Output: [2, 6, 8, 4, 6, 2]
```

Map function

- Used to do some operations on Sequences

```
(__func: (_T1@__init__) -> _S@map, __iter1:
Iterable[_T1@__init__], /) -> None

map(func, *iterables) --> map object

Make an iterator that computes the function using arguments
from each of the iterables. Stops when the shortest iterable is
exhausted.
```

Cont...

- If you want to do a doubling equations on a list of numbers.

```python
1  nums=[3,2,6,8,4,6,2,91]
2
3  def doub(n):
4      return n*2
5  doubles = list(map(doub,nums))
6  print (doubles)
7  # Output: [6, 4, 12, 16, 8, 12, 4, 182]
```

```python
nums=[3,2,6,8,4,6,2,91]
evens = list(filter(lambda n:n%2==0,nums))
doubles = list(map(lambda n:n*2,evens))
print (doubles)
# Output: [4, 12, 16, 8, 12, 4]
```

```python
1  nums=[3,2,6,8,4,6,2,91]
2  doubles = list(map(lambda n:n*2,nums))
3  print (doubles)
4  # Output: [6, 4, 12, 16, 8, 12, 4, 182]
```

Exercise 2

1) Create a program that accepts 10 numbers, then filter the evens , then add 15 to them.
   a) DISPLAY: "Your Data: [.....]"
   b) HINT: Don't forget to accept int() only, use loops,filter,map

Object-Oriented Programming / OOP

- Python is an object oriented programming. This means mores things in python are objects.
- Objects are anything which can have action and name.
- Objects have attributes(properties) and methods(action or functions)
- Example : My Computer is an object because, it have..
    - Attribute: name,size,cpu,ram…
    - Behaviour: Running games, playing music, displaying texts…

WHAT IF I TOLD YOU

EVERYTHING IS AN OBJECT

OBJECT ORIENTED

PROGRAMMING
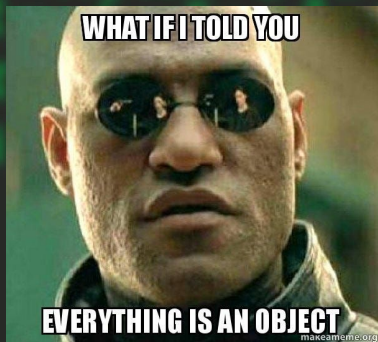
Python class

- Class is simply a place where we create our Object's attribute and behaviour
- It is like a template
- a class is a blueprint for that object.
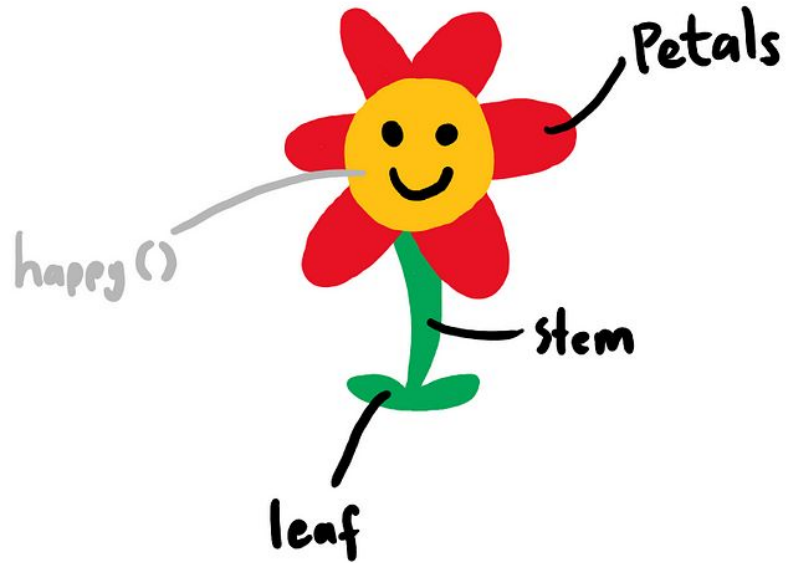- Syntax:

```python
class Computer:

    # Creating Attributes
    name = ""
    cpu = ""
```

- After you create the Blueprint, now you can create the objects based on your class
- On the above example we created Class computer and gave it an attributes of name,cpu
- Conventionally Class Names Start with capital letter

Class

Creating a Class called Flower, with an attributes of the flower like petals, stem,leaf…

Creating Objects

- You can Create many Objects based on 1 class.
- Here we created 2 Objects called
  - Nathan_Computer
  - Alemayew_Computer
- Syntax:
  - Var = classname()
  - Var.attribute = "value"

```python
# Creating an Object based on the blue print
Nathan_Computer = computer()
Nathan_Computer.name = "Hp laptop"
Nathan_Computer.cpu = "Intel Core i5"


# Creating another Object
Alemayew_Computer = computer()
Alemayew_Computer.name = "Dell Desktop"
Alemayew_Computer.cpu = "Intel Core i3"
```

```python
print(f"Nathan's Computer Name is called {Nathan_Computer.name}.\n It is {Nathan_Computer.cpu}")

# Output: Nathan's Computer Name is called Hp laptop.
#         It is Intel Core i5
```
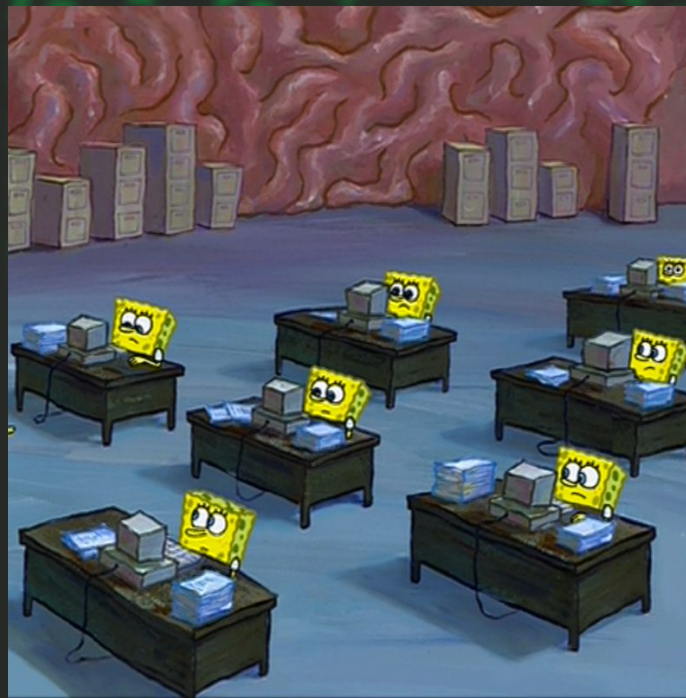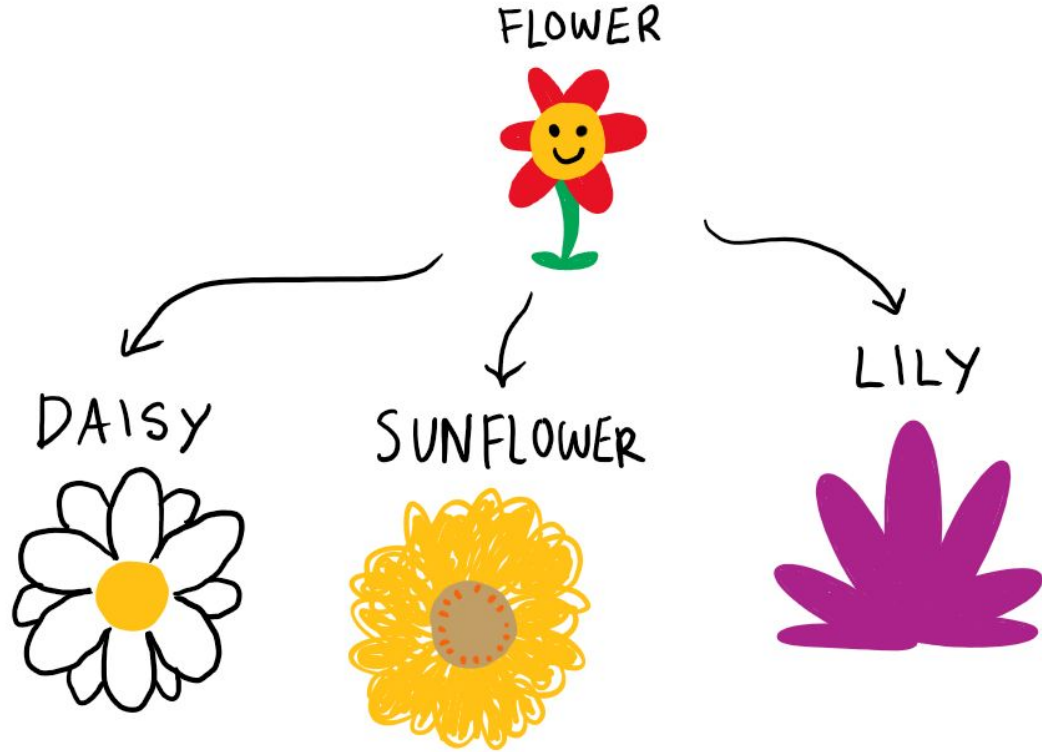
by Nathan Hailu

CLASS

Object

Based on our flower class we can create different types of flowers in this case that is called different objects.

Lets check the type() of our object

```
Nathan_Computer = computer()
print(type(Nathan_Computer))

# Output: <class '__main__.computer'>
```

```
a = 4
print(type(a))

# Output: <class 'int'>
```

- When we check Our type it is similar with type of int, both are classes,
- 'a' and 'Nathan_Computer' are objects
- computer and int are classes

Giving Behaviours == Creating Methods

- Functions are called methods on OOP
- Syntax on calling
  - classname .method(object)

- Creating a Behaviour run, which is 1 behaviour of my computer. And have parameter of 'self'
- Self is the object name on OOP you have to declare it.

```python
class computer:
    # Creating Attributes
    name = ""
    cpu = ""


    # Creating Behavior
    def run(self):
        return "BIOS is G00d!"
```

```python
# Creating an Object based on the blue print
Nathan_Computer = computer()
Nathan_Computer.name = "Hp laptop"
Nathan_Computer.cpu = "Intel Core i5"

print(f"Running: {computer.run(Nathan_Computer)}")
```

Exercise 3

1) Create a class of human.
   a) With properties of name,age,grade
   b) With behaviour of running, dancing, eating. Display "I can run!" "i can dance" "i can eat!" for the behaviours
2) Create an Object with human1 and give the attributes of yours.
3) Display This:

```
# Output: My name is Nathan Hailu
#         I am 20 years old.
#         I am Grade 2 student
#         My skills are: I can RUN! , I can dance and I can eat.
#         Thank You!
```

Python Constructors

```python
class computer:
    # Creating Attributes
    name = ""
    cpu = ""
```

- Earlier we assigned a default value to a class attribute,

However, we can also initialize values using the constructors.

Here, `__init__()` is the constructor method that is called whenever a new object of that class is instantiated.

```python
class Bike:

    # constructor function
    def __init__(self, name = ""):
        self.name = name

bike1 = Bike()
```

The constructor above initializes the value of the `name` attribute. We have used the `self.name` to refer to the `name` attribute of the `bike1` object. -> it is like 2 variables 1 is from outside(`name`) the class and the another is from inside(`self.name`) the class

If we use a constructor to initialize values inside a class, we need to pass the corresponding value during the object creation of the class.

```python
Nathan_Computer = computer()
Nathan_Computer.name = "Hp laptop"
```

```python
bike1 = Bike("Mountain Bike")
```

```python
class computer:
    # Creating Attributes
    def __init__(self,name,cpu):
        self.name = name
        self.cpu = cpu

    # Creating Behavior
    def run(self):
        return "BIOS is G00d!"


# Creating an Object based on the blue print
Nathan_Computer = computer('Hp Laptop','Intel i5')
Alemayew_Computer = computer("Dell Desktop","Intel Core i3")


print(f"Nathan's Computer Name is called {Nathan_Computer.name}.\n It is {Nathan_Computer.cpu}")
```

Exercise 4

1) Rewrite your code with __init__

Exercise 3

1) Create a class of human.
   a) With properties of name,age,grade
   b) With behaviour of running, dancing, eating. Display "I can run!" "i can dance" "i can eat!" for the behaviours
2) Create an Object with human1 and give the attributes of yours.
3) Display This:

```
# Output: My name is Nathan Hailu
#         I am 20 years old.
#         I am Grade 2 student
#         My skills are: I can RUN! , I can dance and I can eat.
#         Thank You!
```

Python Inheritance

- Is a way of creating new class with some properties of existing class.
- Syntax:
  - class newclass(oldclass):
    - ...

```
I can eat!
I can sleep!
I can bark! Woof woof!!
```

```python
# base class
class Animal:

    def eat(self):
        print( "I can eat!")

    def sleep(self):
        print("I can sleep!")

# derived class
class Dog(Animal):

    def bark(self):
        print("I can bark! Woof woof!!")

# Create object of the Dog class
dog1 = Dog()

# Calling members of the base class
dog1.eat()
dog1.sleep()

# Calling member of the derived class
dog1.bark();
```

Python Encapsulation

- Encapsulation is Feature of OOP, That refers to bundling of attributes and methods inside a single class.
- Encapsulation allows for better control and protection of the data, ensuring that it is accessed and modified only through specified methods.

```python
class computer:
    # Creating Attributes
    def __init__(self,name,cpu):
        self.name = name
        self.cpu = cpu
        self.price = 1000

    # Creating Behavior
    def run(self):
        return "BIOS is G00d!"

    def setprice(self,birr):
        self.price = birr

# Creating an Object based on the blue print
Nathan_Computer = computer('Hp Laptop','Intel i5')
Alemayew_Computer = computer("Dell Desktop","Intel Core i3")

print(f"Nathan Computer price is: {Nathan_Computer.price} birr.")

# Ye PC waga chemere!
Nathan_Computer.setprice(2000)

print(f"Nathan Computer price is: {Nathan_Computer.price} birr.")

# Output: Nathan Computer price is: 1000
#         Nathan Computer price is: 2000
```

Package installing

As we have seen package installing on our linux tutorial

We use pip to install tools so, on terminal

    pip install packagename

Package using

- On python, there are a lot of packages to use them, we simply `import packagename`

- Each packages have their own classes and methods so we have to do more studies about those packages.

```
import sys

a = sys.argv[2]
```

Here, we imported sys package and from that package we added the method argv, and creating an object 'a'

- From now on when you want to learn website dev with python it is just learning the package. - Django,flask,panda,numpy....

We will see details on this when we try to create hacking tools in our future classes. DONT FORGET TO PRACTICE!!!

Class is over

1) Ask any question
2) DO notes
3) Practice

Finish Payment Until Nov,6,2023 You have 1 week