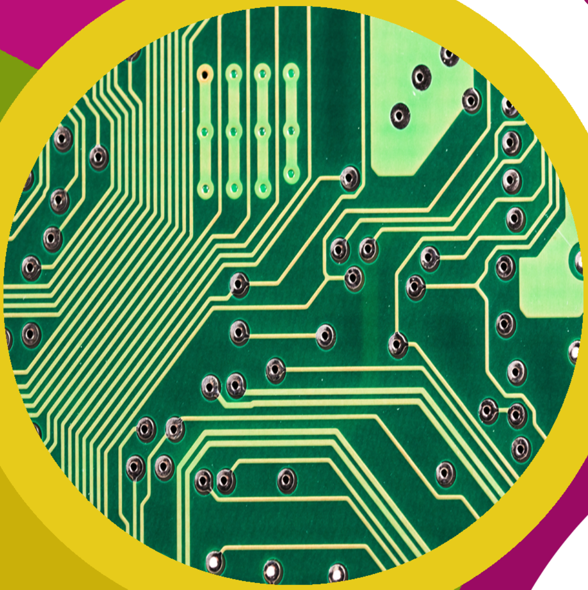
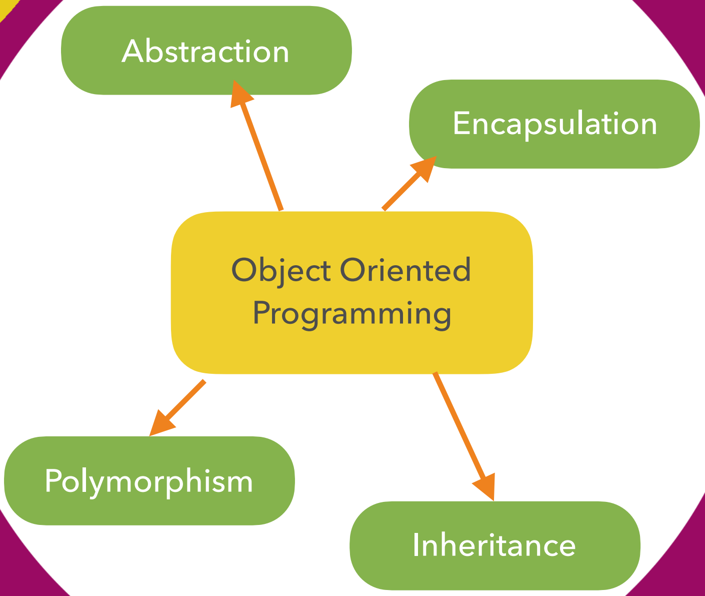
technocamps

Inspiring | Creative | Fun
Ysbrydoledig | Creadigol | Hwyl

Object Oriented GUI Workbook





Overview

In this workshop, we will be looking at the concepts of object oriented programming and how to write our own GUI program using these concepts.

Learning Outcomes

1. Improved knowledge of classes, methods and object in Python
2. Knowledge of the object oriented programming concept abstraction
3. Knowledge of the object oriented programming concept encapsulation
4. Knowledge of the object oriented programming concept inheritance
5. Knowledge of the object oriented programming concept polymorphism

Attendee Prerequisites

1. Experience of Python Programming.
2. Experience of programming GUIs in Python.

Python Review

What does each piece of code do? Write the answer at the side.

```
print(“Hello World!”)
print(27+3)
print(2*2)
print(2**2)
```

```
myName=“technocamps”
print(myName)
```

```
myLastName = input(“Please provide your last name: ”)
print(myLastName)
```

```
grade=70
if grade >= 70 :
    print(“You got an A”)
elif grade >= 60 :
    print(“You got a B”)
else :
    print(”You got a C”)
```

```
counter=1
while(counter<5):
    print(”The counter is ”, counter)
    counter+=1
print(“End”)
```

```
count=0
for count in range(0,5):
    print(“The count is “, count)
print(“End”)
```

Starter Activity: What do they mean?

Write what you think each term means:

Abstraction:____

____

Encapsulation:____

____

Inheritance:____

____

Polymorphism:____

____

Collecting Information

Collect information about 2+ of your classmates including: name, age, date of birth, number of siblings and number of pets.

____

____

____

____

____

____

Classes, Objects and Methods

Classes

Write your own description of a class below:

___

___

___

Objects

Write your own description of an object below:

___

___

___

Methods

Write your own description of a method below:

___

___

___

GUI Code

```
from tkinter import *

months = ["01","02","03","04","05","06","07","08","09","10","11","12"]
days =
["01","02","03","04","05","06","07","08","09","10","11","12","13","14","15","16","17","18","19","20","21","22","23"
,"24","25","26","28","29","30","31"]
students = []

class GUI():
	def __init__(self,root):
		root.title("Student Register")

		labelName = Label(root,text="Student Name:")
		labelName.grid(column=0,row=0)
		studentName = Entry(root)
		studentName.grid(column=1,row=0)

		labelAge = Label(root,text="Student Age:")
		labelAge.grid(column=0,row=1)
		studentAge = Entry(root)
		studentAge.grid(column=1,row=1)

		labelDOB = Label(root,text="Please enter date of birth below (dd/mm/yyyy):")
		labelDOB.grid(column=1,row=2)

		d = StringVar(root)
		d.set(days[0])
		day = OptionMenu(root,d,*days)
		day.grid(column=0,row=3)
		m = StringVar(root)
		m.set(months[0])
		month = OptionMenu(root,m,*months)
		month.grid(column=1,row=3)
		y = StringVar(root)
		year = Entry(root,text=y)
		year.grid(column=2,row=3)

		button = Button(root, text="Add Student", command= lambda:
Student.addStudent(studentName.get(),studentAge.get(),d.get(),m.get(),y.get()))
		button.grid(column=1,row=4)

		button = Button(root, text="Print Students", command= lambda: Student.printStudents())
		button.grid(column=1,row=5)

def main():
	root = Tk()
	gui = GUI(root)
	root.mainloop()

if __name__ == '__main__':
	main()
```

Adding the GUI

GUI Code

Before you copy the code on the previous page into your program draw what you think the GUI will look like below:

Reflection

Write the code into your program and run it. Does the program look like your drawing?

Improving the GUI

Improvements

What can be done to improve the program? Write your ideas below:

____

____

____

Validation

Why is validation important?

____

____

____

Validation

```
import datetime

def validate(d,m,y):
    date_string = d+"-"+m+"-"+y
    date_format = '%d-%m-%Y'
    Result = True
    try:
          date_obj = datetime.datetime.strptime(date_string, date_format)
          except ValueError:
              Result = False
              print("Date of birth is not valid")
          return Result
```

Abstraction and Encapsulation

Abstraction

Look back at your initial thoughts of what abstraction meant, were you correct? Write a new improved definition of it below:

Think of 2 everyday objects you use/see but you don't know what happens behind the scenes. Write down the object, including how you interact with it and the outcome.

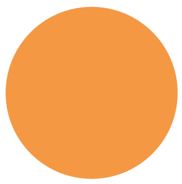
Encapsulation

Look back at your initial thoughts of what encapsulation meant, were you correct? Write a new improved definition of it below:

Inheritance Activity

Match the subclasses to the parent class. Think carefully about what the subclasses will inherit. Some parent classes may have their own parents!

Fruit



Stephen Hawkin

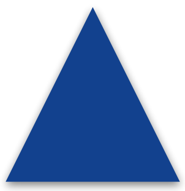
Scientist

David Beckham

Footballer

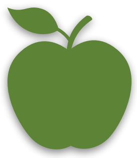
Einstein

Shape

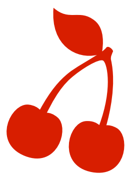


Person

Gareth Bale



Food



Marie Curie

Polymorphism

An interface is a class which has a method defined but no implementation. Parent classes are sometimes interfaces. The parent class defines the method but each subclass has its own implementation.

Select two parent classes from the inheritance activity and write at least two methods per class which can be implemented differently in the subclasses.

Zoo Activity

Use the space below to plan out your classes. Think of the four concepts whilst planning: Abstraction, Encapsulation, Inheritance, Polymorphism.

Key Words

Class	**Abstraction**	**Inheritance**
Method	**Encapsulation**	**Polymorphism**
Object		

Class - A Class is like an object constructor, or a "blueprint" for creating objects. In Python use the **class** keyword to define a class.

Method - A method is a like an instruction that can be called on the class or object. It only runs when it is called. In Python use the **def** keyword to define a function.

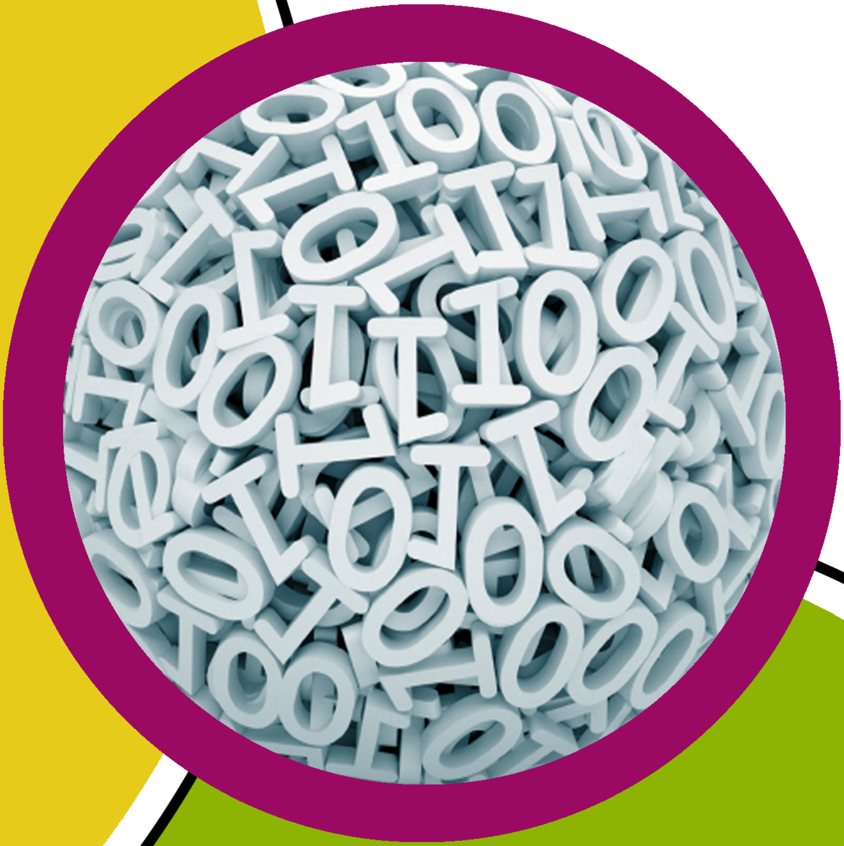
Object - An object is an instance of a class.

Abstraction - Abstraction is the process of removing unnecessary detail and simplifying.

Encapsulation - Encapsulation is achieved when each object keeps its state private, inside a class. Other objects don't have direct access to this state. Instead, they can only call a list of public functions — called methods.

Inheritance - Objects are often very similar. They share common logic. But they're not entirely the same. Inheritance enables new objects to take on the properties of existing objects. A class that is used as the basis for inheritance is called a superclass, base class or parent class. A class that inherits from a superclass is called a subclass, derived class or child class.

Polymorphism - Polymorphism gives a way to use a class exactly like its parent so there's no confusion with mixing types. But each child class keeps its own methods as they are. It is the characteristic of being able to assign a different meaning or usage to something in different contexts - specifically, to allow an entity such as a variable, a function, or an object to have more than one form.



technocamps

Inspiring | Creative | Fun
Ysbrydoledig | Creadigol | Hwyl

@Technocamps

Find us on Facebook