Course Information

Course Name: PythonPlus

Description: Python instruction developed with faculty from Cornell University's Engineering Department using robots as a teaching platform.

Sell Date: May 1, 2020

Ages: Students 9 to 12 Years. Students under 9 years of age with prior coding experience.

Pricing: Monthly Course Fee $99.00 for one (1) 60 minute classes per week or $178 for 2 60 minute classes per week.

Billing: You will be billed 30 days after your first billing.

Additional Materials: No Purchase Necessary

Instructor/Student Ratio: 1 to 8

Course Description

Working with faculty at Cornell University School of Engineering, UCode developed a curriculum to teach two related disciplines: **1) Computational Thinking** and **2) Coding**.

Computational thinking consists of breaking down a complex problem into smaller, more manageable parts (*known as decomposition*); looking for similarities among and within problems (*pattern recognition*); focusing on the important information only while ignoring irrelevant details (*abstraction*) and developing a step-by-step solution to the problem, or the rules to follow to solve the problem (*algorithms*).

Coding is a structured method of **communicating the solution (a set of instructions) to a computer**. While computational thinking enables you to work out exactly what to tell the computer to do, coding tells a computer what to do and how to do it.

Experiential Curriculum

To teach students these skills, UCode has developed an experiential approach to learning that removes students from the traditional classroom setting and allows them to immerse themselves in solving real-world problems.

We call these real-world problems "Challenges" - tasks that their robot must complete. The initial Challenges are quite simple. For example, having a robot stop at a certain distance or travel a path that completes a square. However, as a student progresses the Challenges become increasingly difficult requiring development of their Computational Thinking skills and the integration of more advanced logic and problem-solving skills.

Through solving these real-world problems, students develop a deeper and more sophisticated understanding of traditional school subjects such as Science and Math. Students who learn how to think computationally increase their ability to think logically, think spatially and to think creatively making them better test-takers, better problem solvers and better students.

Python

The core language of instruction is Python a programming language that has become the standard in education. Python uses intuitive commands such as 'if', 'for', 'while', 'try', 'with' and 'print' that follow the syntax of basic English composition and are easily recognizable by young learners. And unlike Scratch, or Mindstorms - which are "educational programming languages", Python is extensively used in the commercial world. Google and Facebook are built on Python, as well as most of the natural language processing libraries (NLP) driving machine learning. The ability to solve problems in Python has real value for our students outside of the classroom.

Through this learning journey, **students are guided by UCode Instructors** whose role is to support students by clarifying the Challenge and suggesting solution strategies.

Unlike in a school classroom, the instructor does not provide answers as students must take responsibility for their own learning. The instructor's role is more like that of coach making sure that students build their computational ability through *quality of practice*. You get good at coding, through practice. There are no grades when solving a Challenge – there is no wrong code or right.

Building new skills is difficult. Who can play great tennis or basketball the first time on a court? It takes practice. Students learn to take ownership for their learning, just like they will need to do at university or in the work place.

Most students flourish with this approach. They take ownership of their learning and have a great sense of achievement when they solve a Challenge on their own. They progress very quickly.

Concepts Covered

168 concepts are covered in the first 20 Units. These are outlined in the Python Periodic Table.

Assessment: Progress is measured by the completion of a Challenge in that the Challenges are "progressive" and increase in difficulty thereby requiring greater application of coding and computational thinking skills. Understanding of Core concepts is measured through a on-line test at periodic points in the student's learning progression.

Additional Resources:

1. Python Periodic Table
2. Course Overview (Units 1-20)
3. Python White Paper
4. Computational Thinking White Paper
5. Experiential Learning White Paper

Version.3.0