

# Contents

<b>Contents</b>	<b>1</b>
1. Curso . . . . .	2
2. Información general . . . . .	2
3. Profesores . . . . .	2
4. Introducción al curso . . . . .	2
5. Objetivos . . . . .	2
6. Competencias . . . . .	2
7. Temas . . . . .	2
8. Plan de trabajo . . . . .	7
8.1 Metodología . . . . .	7
8.2 Sesiones Teóricas . . . . .	8
8.3 Sesiones Prácticas . . . . .	8
9. Planificación . . . . .	8
10. Sistema de Evaluación . . . . .	8
11. Bibliografía básica . . . . .	8

# Universidad de Piura (UDEP)

## Sílabo 2022-I

### 1. CURSO

CS111. Introducción a la Ciencia de la Computación (Obligatorio)

### 2. INFORMACIÓN GENERAL

2.1 Créditos	:	4
2.2 Horas de teoría	:	2 (Semanal)
2.3 Horas de práctica	:	2 (Semanal)
2.4 Duración del periodo	:	16 semanas
2.5 Condición	:	Obligatorio
2.6 Modalidad	:	Presencial
2.7 Prerrequisitos	:	Ninguno

### 3. PROFESORES

Atención previa coordinación con el profesor

### 4. INTRODUCCIÓN AL CURSO

Este es el primer curso en la secuencia de los cursos introductorios a la Ciencia de la Computación. En este curso se pretende cubrir los conceptos señalados por la Computing Curricula IEEE-CS/ACM 2013. La programación es uno de los pilares de la Ciencia de la Computación; cualquier profesional del Área, necesitará programar para concretizar sus modelos y propuestas. Este curso introducción a los participantes en los conceptos fundamentales de este arte. Lo tópicos incluyen tipos de datos, estructuras de control, funciones, listas, recursividad y la mecánica de la ejecución, prueba y depuración.

### 5. OBJETIVOS

- Introducir los conceptos fundamentales de programación.
- Desarrollar su capacidad de abstracción utilizar un lenguaje de programación.

### 6. COMPETENCIAS

Nooutcomes

Nospecificoutcomes

### 7. TEMAS

Unidad 1: Historia (5)	
Competencias esperadas: a	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"><li>• Pre-historia – El mundo antes de 1946.</li><li>• Historia del hardware, software, redes.</li><li>• Pioneros de la Computación.</li><li>• Historia de Internet.</li></ul>	<ul style="list-style-type: none"><li>• Identificar importantes tendencias en la historia del campo de la computación [Familiarizarse]</li><li>• Identificar las contribuciones de varios pioneros en el campo de la computación [Familiarizarse]</li><li>• Discutir el contexto histórico de los paradigmas de diversos lenguajes de programación [Familiarizarse]</li><li>• Comparar la vida diaria antes y después de la llegada de los ordenadores personales y el Internet [Evaluar]</li></ul>
Lecturas : [BB19], [Gut13], [Zel10]	

## Unidad 2: Sistemas de tipos básicos (2)

### Competencias esperadas: a

Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"><li>• Tipos como conjunto de valores junto con un conjunto de operaciones.<ul style="list-style-type: none"><li>– Tipos primitivos (p.e. números, booleanos)</li><li>– Composición de tipos contruídos de otros tipos (p.e., registros, uniones, arreglos, listas, funciones, referencias)</li></ul></li><li>• Asociación de tipos de variables, argumentos, resultados y campos.</li><li>• Tipo de seguridad y los errores causados por el uso de valores de manera incompatible dadas sus tipos previstos.</li></ul>	<ul style="list-style-type: none"><li>• Tanto para tipo primitivo y un tipo compuesto, describir de manera informal los valores que tiene dicho tipo [Familiarizarse]</li><li>• Para un lenguaje con sistema de tipos estático, describir las operaciones que están prohibidas de forma estática, como pasar el tipo incorrecto de valor a una función o método [Familiarizarse]</li><li>• Describir ejemplos de errores de programa detectadas por un sistema de tipos [Familiarizarse]</li><li>• Para múltiples lenguajes de programación, identificar propiedades de un programa con verificación estática y propiedades de un programa con verificación dinámica [Usar]</li><li>• Usar tipos y mensajes de error de tipos para escribir y depurar programas [Usar]</li><li>• Definir y usar piezas de programas (tales como, funciones, clases, métodos) que usan tipos genéricos, incluyendo para colecciones [Usar]</li></ul>
<b>Lecturas :</b> [Gut13], [Zel10]	

**Unidad 3: Conceptos Fundamentales de Programación (9)****Competencias esperadas: a**

<b>Temas</b>	<b>Objetivos de Aprendizaje</b>
<ul style="list-style-type: none"><li>• Sintaxis y semántica básica de un lenguaje de alto nivel.</li><li>• Variables y tipos de datos primitivos (ej., números, caracteres, booleanos)</li><li>• Expresiones y asignaciones.</li><li>• Operaciones básicas I/O incluyendo archivos I/O.</li><li>• Estructuras de control condicional e iterativas.</li><li>• Paso de funciones y parámetros.</li><li>• Concepto de recursividad.</li></ul>	<ul style="list-style-type: none"><li>• Analiza y explica el comportamiento de programas simples que involucran estructuras fundamentales de programación variables, expresiones, asignaciones, E/S, estructuras de control, funciones, paso de parámetros, y recursividad [Evaluar]</li><li>• Identifica y describe el uso de tipos de datos primitivos [Familiarizarse]</li><li>• Escribe programas que usan tipos de datos primitivos [Usar]</li><li>• Modifica y expande programas cortos que usen estructuras de control condicionales e iterativas así como funciones [Usar]</li><li>• Diseña, implementa, prueba, y depura un programa que usa cada una de las siguientes estructuras de datos fundamentales: cálculos básicos, E/S simple, condicional estándar y estructuras iterativas, definición de funciones, y paso de parámetros [Usar]</li><li>• Escribe un programa que usa E/S de archivos para brindar persistencia a través de ejecuciones múltiples [Usar]</li><li>• Escoje estructuras de condición y repetición adecuadas para una tarea de programación dada [Familiarizarse]</li><li>• Describe el concepto de recursividad y da ejemplos de su uso [Evaluar]</li><li>• Identifica el caso base y el caso general de un problema basado en recursividad [Familiarizarse]</li></ul>
<b>Lecturas :</b> [Gut13], [Zel10]	

Unidad 4: Análisis Básico (2)	
Competencias esperadas: a,b	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>• Diferencias entre el mejor, el esperado y el peor caso de un algoritmo.</li> <li>• Definición formal de la Notación Big O.</li> <li>• Clases de complejidad como constante, logarítmica, lineal, cuadrática y exponencial.</li> <li>• Uso de la notación Big O.</li> <li>• Análisis de algoritmos iterativos y recursivos.</li> </ul>	<ul style="list-style-type: none"> <li>• Explique a que se refiere con “mejor”, “esperado” y “peor” caso de comportamiento de un algoritmo [Familiarizarse]</li> <li>• En el contexto de a algoritmos específicos, identifique las características de data y/o otras condiciones o suposiciones que lleven a diferentes comportamientos [Familiarizarse]</li> <li>• Indique la definición formal de Big O [Familiarizarse]</li> <li>• Use la notación formal de la Big O para dar límites superiores asintóticos en la complejidad de tiempo y espacio de los algoritmos [Usar]</li> <li>• Usar la notación formal Big O para dar límites de casos esperados en el tiempo de complejidad de los algoritmos [Usar]</li> </ul>
<b>Lecturas :</b> [Gut13], [Zel10]	

Unidad 5: Algoritmos y Estructuras de Datos fundamentales (8)	
Competencias esperadas: a,b	
Temas	Objetivos de Aprendizaje
<ul style="list-style-type: none"> <li>• Algoritmos numéricos simples, tales como el cálculo de la media de una lista de números, encontrar el mínimo y máximo.</li> <li>• Algoritmos de búsqueda secuencial y binaria.</li> <li>• Algoritmos de ordenamiento de peor caso cuadrático (selección, inserción)</li> <li>• Algoritmos de ordenamiento con peor caso o caso promedio en <math>O(N \lg N)</math> (Quicksort, Heapsort, Mergesort)</li> <li>• Tablas Hash, incluyendo estrategias para evitar y resolver colisiones.</li> <li>• Árboles de búsqueda binaria: <ul style="list-style-type: none"> <li>– Operaciones comunes en árboles de búsqueda binaria como seleccionar el mínimo, máximo, insertar, eliminar, recorrido en árboles.</li> </ul> </li> <li>• Grafos y algoritmos en grafos: <ul style="list-style-type: none"> <li>– Representación de grafos (ej., lista de adyacencia, matriz de adyacencia)</li> <li>– Recorrido en profundidad y amplitud</li> </ul> </li> <li>• Montículos (Heaps)</li> <li>• Grafos y algoritmos en grafos: <ul style="list-style-type: none"> <li>– Problema de corte máximo y mínimo</li> <li>– Búsqueda local</li> </ul> </li> <li>• Búsqueda de patrones y algoritmos de cadenas/texto (ej. búsqueda de subcadena, búsqueda de expresiones regulares, algoritmos de subsecuencia común más larga)</li> </ul>	<ul style="list-style-type: none"> <li>• Implementar algoritmos numéricos básicos [Usar]</li> <li>• Implementar algoritmos de búsqueda simple y explicar las diferencias en sus tiempos de complejidad [Evaluar]</li> <li>• Ser capaz de implementar algoritmos de ordenamiento comunes cuadráticos y <math>O(N \log N)</math> [Usar]</li> <li>• Describir la implementación de tablas hash, incluyendo resolución y el evitamiento de colisiones [Familiarizarse]</li> <li>• Discutir el tiempo de ejecución y eficiencia de memoria de los principales algoritmos de ordenamiento, búsqueda y hashing [Familiarizarse]</li> <li>• Discutir factores otros que no sean eficiencia computacional que influyan en la elección de algoritmos, tales como tiempo de programación, mantenibilidad, y el uso de patrones específicos de la aplicación en los datos de entrada [Familiarizarse]</li> <li>• Explicar como el balanceamiento del arbol afecta la eficiencia de varias operaciones de un arbol de búsqueda binaria [Familiarizarse]</li> <li>• Resolver problemas usando algoritmos básicos de grafos, incluyendo búsqueda por profundidad y búsqueda por amplitud [Usar]</li> <li>• Demostrar habilidad para evaluar algoritmos, para seleccionar de un rango de posibles opciones, para proveer una justificación por esa selección, y para implementar el algoritmo en un contexto en específico [Evaluar]</li> <li>• Describir la propiedad del heap y el uso de heaps como una implementación de colas de prioridad [Familiarizarse]</li> <li>• Resolver problemas usando algoritmos de grafos, incluyendo camino más corto de una sola fuente y camino más corto de todos los pares, y como mínimo un algoritmo de arbol de expansion minima [Usar]</li> <li>• Trazar y/o implementar un algoritmo de comparación de string [Usar]</li> </ul>
<b>Lecturas :</b> [Gut13], [Zel10]	

<b>Unidad 6: Algoritmos y Diseño (9)</b>	
<b>Competencias esperadas: a,b</b>	
<b>Temas</b>	<b>Objetivos de Aprendizaje</b>
<ul style="list-style-type: none"> <li>• Conceptos y propiedades de los algoritmos <ul style="list-style-type: none"> <li>– Comparación informal de la eficiencia de los algoritmos (ej., conteo de operaciones)</li> </ul> </li> <li>• Rol de los algoritmos en el proceso de solución de problemas</li> <li>• Estrategias de solución de problemas <ul style="list-style-type: none"> <li>– Funciones matemáticas iterativas y recursivas</li> <li>– Recorrido iterativo y recursivo en estructura de datos</li> <li>– Estrategias Divide y Conquistar</li> </ul> </li> <li>• Conceptos y principios fundamentales de diseño <ul style="list-style-type: none"> <li>– Abstracción</li> <li>– Descomposición de Program</li> <li>– Encapsulamiento y camuflaje de información</li> <li>– Separación de comportamiento y aplicación</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Discute la importancia de los algoritmos en el proceso de solución de un problema [Familiarizarse]</li> <li>• Discute como un problema puede ser resuelto por múltiples algoritmos, cada uno con propiedades diferentes [Familiarizarse]</li> <li>• Crea algoritmos para resolver problemas simples [Usar]</li> <li>• Usa un lenguaje de programación para implementar, probar, y depurar algoritmos para resolver problemas simples [Usar]</li> <li>• Implementa, prueba, y depura funciones recursivas simples y sus procedimientos [Usar]</li> <li>• Determina si una solución iterativa o recursiva es la más apropiada para un problema [Evaluar]</li> <li>• Implementa un algoritmo de divide y vencerás para resolver un problema [Usar]</li> <li>• Aplica técnicas de descomposición para dividir un programa en partes más pequeñas [Usar]</li> <li>• Identifica los componentes de datos y el comportamiento de múltiples tipos de datos abstractos [Usar]</li> <li>• Implementa un tipo de dato abstracto coherente, con la menor pérdida de acoplamiento entre componentes y comportamientos [Usar]</li> <li>• Identifica las fortalezas y las debilidades relativas entre múltiples diseños e implementaciones de un problema [Evaluar]</li> </ul>
<b>Lecturas :</b> [Gut13], [Zel10]	

<b>Unidad 7: Métodos de Desarrollo (1)</b>	
<b>Competencias esperadas: a,b</b>	
<b>Temas</b>	<b>Objetivos de Aprendizaje</b>
<ul style="list-style-type: none"> <li>• Entornos modernos de programación: <ul style="list-style-type: none"> <li>– Búsqueda de código.</li> <li>– Programación usando librería de componentes y sus APIs.</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>• Construir y depurar programas que utilizan las bibliotecas estándar disponibles con un lenguaje de programación elegido [Familiarizarse]</li> </ul>
<b>Lecturas :</b> [Gut13], [Zel10]	

## 8. PLAN DE TRABAJO

### 8.1 Metodología

Se fomenta la participación individual y en equipo para exponer sus ideas, motivándolos con puntos adicionales en las diferentes etapas de la evaluación del curso.

### 8.2 Sesiones Teóricas

Las sesiones de teoría se llevan a cabo en clases magistrales donde se realizarán actividades que propicien un aprendizaje activo, con dinámicas que permitan a los estudiantes interiorizar los conceptos.

### 8.3 Sesiones Prácticas

Las sesiones prácticas se llevan en clase donde se desarrollan una serie de ejercicios y/o conceptos prácticos mediante planteamiento de problemas, la resolución de problemas, ejercicios puntuales y/o en contextos aplicativos.

## 9. PLANIFICACIÓN

FECHA	HORARIO	TIPO DE SESIÓN	DOCENTE
Consultar en EDU	Consultar en EDU	Consultar en EDU	Consultar en EDU

## 10. SISTEMA DE EVALUACIÓN

\*\*\*\*\* EVALUATION MISSING \*\*\*\*\*

## 11. BIBLIOGRAFÍA BÁSICA

- [BB19] J. Glenn Brookshear and Dennis Brylow. *Computer Science: An Overview*. Ed. by PEARSON. Global Edition. Pearson, 2019. ISBN: 1292263423. URL: <http://www.pearsonhighered.com/brookshear>.
- [Gut13] John V Guttag. . *Introduction To Computation And Programming Using Python*. MIT Press, 2013.
- [Zel10] John Zelle. *Python Programming: An Introduction to Computer Science*. Franklin, Beedle & Associates Inc, 2010.