מפגש פתיחה למעבדה א' פיזיקה

לקראת הקורס מעבדה א' בפיזיקה, אנו עורכים מפגש פתיחה לפני תחילת הסמסטר. המפגש יתקיים **באולם לב, בתאריך 22.10.2025 בשעות 09:00-13:30**, לפני תחילת סמסטר א'. הנוכחות איננה חובה והקלטה של הכנס תועלה לפני תחילת הסמסטר.

לתשומת ליבכם - מפגשי המעבדה יחלו בשבוע הלימודים הראשון לסמסטר. המפגש נותן בסיס מתמטי להמשך העבודה במעבדות במהלך הסמסטר, **והחומר המועבר בו הינו קריטי לצורך תפקוד בקורס. למייל שנשלח לתלמידים צורף תרגיל בית להגשה במפגש המעבדה הראשון. הציון על תרגיל זה ייכלל בציון הסופי בקורס.**

נושאי המפגש הנם:

- מבוא לקורס המעבדה בפיזיקה, מטרות המעבדה ומבוא לפיזיקה ניסיונית.
- מבנה המעבדה, נהלי המעבדה, סדרי מפגשי המעבדה הראשונים בסמסטר.
- חישוב שגיאות – אופן הביצוע במעבדה והגדרות מתמטיות.

יש לקרוא את חוברת עיבוד הנתונים. חומר הקריאה יהיה זמין גם ב- Moodle של קורס המעבדה, שיהיה נגיש לאחר הרישום לקורס בבידינג.

בברכת סמסטר פורה,

ד"ר שני דניאלי
אחראית מעבדות פיזיקה א'
ביה"ס לפיזיקה ואסטרונומיה

בכל בעיה ושאלה יש לפנות למדריכים הראשיים במייל: labahead@tauex.tau.ac.il

בחינת פטור בקורס" מבוא לתכנות לפיזיקאים"

Computers for Physics

(מס' קורס 0321.1121.01)

בסמסטר א' מתקיים הקורס "Computers for Physics" (מתקיים בשפה האנגלית), שהינו קורס חובה בחלק מתוכניות הלימוד.

על פי תקנות ביה"ס לפיזיקה, וכפי שמצוין בידיעון: "**תלמידים בעלי ידע קודם בתכנות אשר יעברו בהצלחה את הבחינה שתתקיים בתחילת הסמסטר, יקבלו פטור מקורס זה**".

לתשומת לב!

הבחינה נועדה **אך ורק** לתלמידים אשר הקורס "Computers for Physics " הוא **קורס חובה בתוכנית הלימודים שלהם,** ולא נלמד כקורס בחירה.

בחינת הפטור תתקיים **ביום שישי,** בשבוע השני לתחילת הלימודים:

7.11.2025 בשעה 9:00.

תלמידים המעוניינים להבחן, חייבים להירשם בלינק -
https://forms.gle/pccw2kzGNEohvzZW6

וזאת **לא יאוחר מתאריך 17.9.2025** בצירוף פרטים אישיים: שם, מס' ת.ז. וחוג לימוד.

בכל שאלה ניתן לפנות למרצה הקורס פרופ' רועי בק-ברקאי: roy@tauex.tau.ac.il

החומר לבחינה הוא החומר שיילמד בקורס כפי שנפרסם לקראת תחילת השנה.

מבוא לתכנות לפיזיקאים – Computers for physics

מספר קורס: 0321-1121

סטודנט/ית יקר/ה

ברכות על קבלתך ללימודי פיזיקה באוניברסיטת תל אביב.

כמו כל תחום דעת מדעי, גם פיזיקה בעידן המודרני נשענת על שימוש במשאבי מחשב ותכנות. ועל כן, כחלק מהכשרתך הבסיסית בלימודי פיזיקה אנו מקיימים בשנה הראשונה קורס מבוא לרכישת ידע בסיסי בתכנות. קורס התכנות השנה מתרכז **בשפת התכנות Python** אשר הפכה בשנים האחרונות לכלי עזר מרכזי למרבית הפיזיקאים העוסקים בניסוי ותאוריה ברחבי העולם. במהלך הלימודים תתבקשו לבצע משימות תכנות כחלק משעורי הבית בלימודי הקורסים השונים בפיזיקה.

פורמט הקורס שונה משאר הקורסים אותו תלמדו במהלך התואר והוא מבוסס על **למידה עצמאית** עם אתר שנבנה במיוחד לקורס זה. האתר מלמד את עקרונות התכנות ומאפשר לכם להתנסות בכתיבת וקריאת קוד, מציאת בעיות בקוד תוך כדי אפשרות להרצה של הקוד באתר עצמו. קישור לאתר האינטראקטיבי של הקורס ישלח לכם בתחילת השנה ויופיע באתר ה moodle של הקורס.

בשבוע הראשון ללימודים נפגש למפגש פתיחה איתי, פרופ׳ רועי בק-ברקאי, בו אסביר על עקרונות הקורס ויעדיו ואענה על שאלות הבהרה לגבי מהלך הקורס. **הפגישה תתקיים באולם דארך, ביום ראשון 26.10.25, בשעה 17:00.**

אחת לשבועיים / שלושה תתקיים **שעות תרגול** בה תחדדו את אשר למדתם באופן עצמאי עם המתרגל בכיתת המחשבים בבניין דן-דוד. פרטים מדויקים על תאריכי התרגולים יופיעו באתר הmoodle של הקורס.

מי שיודע לתכנת ברמה מספקת ניתנת האפשרות לגשת ל**מבחן פטור בתחילת הסמסטר, ביום שישי השני של הסמסטר (7.11.25)**. פרטים נוספים על הבחינה יינתנו במפגש הראשון של הקורס ובתכתובות באתר הקורס. החומר הנכלל בבחינת הפטור ניתן בסילבוס הקורס.

במהלך הסמסטר יתקיימו **4 מבדקים**. המבדקים יהיו על החומר הנלמד בלמידה העצמאית והתרגולים ועל פי חלוקת השבועות שמופיעה בסילבוס. המבדקים יתקיימו בשעות "השיעור" **בימי ראשון בשעה 17:00. חובת הגעה למבדקים.** תאריכי המבדקים הצפויים הם: 23.11, 14.12, 4.1, 25.1.

שלושת הבחנים הראשונים שווים 20 נקודות והבוחן האחרון 40 נקודות.

בנוסף, עבור אלו שפספסו עד שני בחנים מסיבות מוצדקות (מילואים / מחלה וכד׳) יינתן בוחן השלמות נוסף ב18.1.

בקורס זה לא ניתן ציון מספרי והוא לא נכלל בממוצע הציונים לתואר. **ע"מ לעבור את הקורס אתם נדרשים להיות נוכחים בלפחות שלוש בחנים ולהשיג ציון מצטבר של 60 נקודות.**

נפגש במפגש הפתיחה בו אענה על כל השאלות שיעלו בנוגע לקורס.

בהצלחה,

פרופ׳ רועי בק-ברקאי וצוות הקורס

Course syllabus

- Weeks 1-6 (Basics) give everyone a common vocabulary by adapting the clear, bite-sized notebooks from the Virtual Python Programming site.
- Weeks 7-12 (Advanced) numerical and software-engineering tools that modern research physicists actually use. Optional topics: object-oriented, debuggable, data-driven codebases.

Week 1 — Introduction: Python + Jupyter Basics

- what a programming language is, how source code becomes machine code, and why Python is popular for scientists.
- Installing Python (Anaconda 3.11) and setting up Jupyter Lab / PyCharm.
- Fundamental data-types: int, float, bool, str; arithmetic, floor-division //, modulo %, exponent **.
- Variables & assignment, memory model, using Python Tutor to visualise state.
- Comparisons and Boolean operators (and / or / not).
- Strings: concatenation, indexing & slicing, immutability, key string methods (upper, find, replace …).

Week 2 — Functions & Conditionals

- Functions (A): why we encapsulate code; syntax of def, parameters vs. return; built-in helpers (len, print).
- Writing reusable helpers (e.g. circumference, max2).
- Conditionals: if, if-else, chained elif, nested decisions; logical operators inside tests.
- Practical flow-charts and small decision-making exercises (hat / coat / umbrella, triangle classifier, etc.).

Week 3 — Lists, Ranges & Loops

- Creating and manipulating lists; indexing, slicing, built-in fns (len, sum, sorted).
- Mutability, list methods (append, remove, pop, sort), nested lists.
- range objects and numeric sequences.
- Loops: for over iterables and range; while; loop control (break, continue).
- Nested loops, factorial example, quick intro to algorithmic complexity & timing.

Week 4 — Names, Scope & Functions (B)

- References vs. values: how assignment behaves for mutable and immutable objects.
- More on tuples vs. lists, and pointer aliasing pitfalls.
- Scope rules: formal vs. actual parameters, local vs. global variables.
- Returning information: return, mutating a passed-in mutable, or (discouraged) globals.
- Functions calling other functions; code modularity.
- Default and keyword arguments; higher-order functions & lambda expressions.

Week 5 — Data-Structures Deep-Dive

- Tuples – fixed-length, immutable sequences.
- Dictionaries (dict): key–value access, requirements on keys, common methods (get, keys, items, update), dynamic dictionary views.
- Worked example: character-frequency counter with sorted output.
- Lists, part B: list comprehensions (optional filter + conditional expression) for concise list creation.

Week 6 — Working with Text Files

- Review of modern string-formatting (str.format, f-strings) and numeric precision specifiers.
- Files vs. folders; absolute vs. relative paths; Windows back-slash caveats.
- open(path, mode) with 'r', 'w', 'a'; reading: read, readline, readlines, iteration; writing with write.
- Cleaning text: split, strip, rstrip, lstrip.
- The with … as context-manager for automatic close().
- End-to-end examples: word-frequency printer, writing a list of squares to a file.

Week 7 – NumPy I: Arrays & Vectorization

- Creating ndarrays from lists, generators, or files; dtypes and type coercion
- Elementwise arithmetic, slicing, views vs. copies, broadcasting rules
- Timing vectorized operations versus pure-Python loops

Week 8 – NumPy II: Linear Algebra & Random Numbers

- Matrix and vector operations with numpy.linalg (solve, eig, SVD)
- Structured arrays and memory layout tips (order, strides)
- Modern random-number API (numpy.random.Generator), reproducible streams, statistical sampling

Week 9 – SciPy & Matplotlib

- ODE solving (solve_ivp), root-finding, optimization, FFT fundamentals
- Basic and advanced figure layout, annotations, exporting to high-resolution formats
- Best practices for reproducible plotting scripts (style cyclers, tight-layout, labels)

Week 10 – Pandas for Data Analysis

- Series vs. DataFrame anatomy, indexing, Boolean masking
- groupby aggregation, joins/merge for heterogeneous datasets
- Time-series resampling and rolling statistics; efficient IO (CSV, Parquet, HDF5)

Week 11 – Object-Oriented Programming for Simulations

- Classes, __init__, encapsulation, properties, dunder methods for operator overloading
- Inheritance vs. composition; designing Particle, ForceField, Integrator hierarchies
- Packaging a small library (directory structure, __init__.py, setup.cfg) for reuse or publication

Week 12 – Unit Testing & Quality Assurance

- Test-Driven Development workflow and the value of small, isolated tests
- pytest syntax: test discovery, assertions, parametrization, fixtures
- Measuring coverage, mocking expensive dependencies, basic CI concepts (GitHub Actions exemplar)