- ☐ **Python**
  - ☐ **Podstawowe informacje**
    - ☐ Języki nisko- i wysokopoziomowe
    - ☐ *PYTHONPATH*
    - ☐ Python 2 vs 3 (+ pliki *.pyc*)
    - ☐ Biblioteka standardowa + import modułów
    - ☐ Zmienne
      - ☐ Typy zmiennych
        - ☐ Liczbowe: *int, float*
        - ☐ Znakowe: *str*
          - ☐ operacje na *str*
          - ☐ f-string, raw string
        - ☐ Logiczne: *bool*
        - ☐ Kolekcje: *list, dict, tuple, set*
        - ☐ *None*
      - ☐ Mutowalne vs niemutowalne
    - ☐ Operatory arytmetyczne (+, -, *, /, //, %, **)
    - ☐ Operatory przypisania (=, +=, -=, *=, /=, //=, %=, **=)
    - ☐ Operatory porównania (<, >, <=, >=, ==, !=)
      - ☐ Referencje (“==” vs “*is*”)
    - ☐ Instrukcje warunkowe: *if, elif, else*
    - ☐ Operatory logiczne (*and* = koniunkcja, *or* = alternatywa + *not*)
    - ☐ Pętle: *while* i *for* (w jakich przypadkach stosujemy każdą z nich)
      - ☐ *break* i *continue*
      - ☐ Pętle zagnieżdżone
    - ☐ Złożone struktury danych
      - ☐ Listy
        - ☐ Operacje na listach
      - ☐ Tuple
      - ☐ Słowniki
        - ☐ Operacje na słownikach
      - ☐ Zbiory
        - ☐ Operacje na zbiorach
      - ☐ Porównanie cech złożonych str danych
      - ☐ Zagnieżdżanie struktur danych

- ☐ Funkcje
  - ☐ Parametr vs argument funkcji
  - ☐ Sposoby przekazywania argumentów do funkcji
    - ☐ Argumenty kluczowe (nazwane) i pozycyjne (nienazwane)
    - ☐ Argumenty domyślne
    - ☐ **args* i ***kwargs*
  - ☐ Zmienne lokalne i globalne (*global*)
  - ☐ Lambda

- ☐ Praca z plikami
  - ☐ Tryby: r, w, a, x +
    - ☐ r*ead, readline, readlines* ()
      - ☐ Usuwanie białych znaków: *.strip, .splitlines* ()
    - ☐ *write, writelines* ()
    - ☐ Porównanie trybów pod kątem:
      - ☐ nadpisania zawartości
      - ☐ gdzie jest wskaźnik
        - ☐ *seek, tell* ()
  - ☐ Ścieżka względna vs bezwzględna
  - ☐ *open, close* ()
    - ☐ *with open() as* …
  - ☐ ASCII
  - ☐ JSON (*JavaScript Object Notation*)
    - ☐ Parsowanie
    - ☐ Serializacja i deserializacja
      - ☐ *json.dump, load* ()

- ☐ Programowanie obiektowe = OOP (*object-oriented programming*)
  - ☐ Klasy i obiekty, atrybuty i metody
  - ☐ *__init__* i *self*
  - ☐ 4 cechy OOP
    - ☐ Abstrakcja
    - ☐ Hermetyzacja = enkapsulacja
      - ☐ Pola __prywatne, _chronione i publiczne
      - ☐ getter i setter
    - ☐ Polimorfizm
    - ☐ Dziedziczenie
      - ☐ Dziedziczenie wielokrotne

- ☐ Kolejność wywoływania
  - ☐ MRO (*Method Resolution Order*), *mro()*
- ☐ Referencja do obiektu
  - ☐ Kopie płytkie i głębokie
- ☐ Metody magiczne = specjalne = __dunder__
  - ☐ *__str__, __repr__*
  - ☐ Przeładowanie operatorów

- ☐ **Informacje bardziej zaawansowane**
  - ☐ *RegEx*
  - ☐ Listy składane (*list comprehension*)
  - ☐ Generatory
    - ☐ yield
  - ☐ Dekoratory
  - ☐ Wyjątki
  - ☐ Wielowątkowość
    - ☐ Proces, wątek
    - ☐ GIL (*Global Interpreter Lock*)
    - ☐ Zliczanie referencji (*Garbage Collector*)
    - ☐ *Multithreading* vs *Asyncio*: https://www.devs-mentoring.pl/asyncio-vs-multithreading/
  - ☐ Czysty kod i dobre praktyki
    - ☐ PEP8 (*Python Enhancement Proposals*)
    - ☐ Dokumentacja
    - ☐ *DRY (Don't Repeat Yourself), KISS (Keep It Simple, Stupid)*
    - ☐ *SOLID (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, Dependency Inversion)*

☐ Testy

- ☐ Poziomy testów
  - ☐ Jednostkowe = modułowe
    - ☐ *Unittest, Pytest*
  - ☐ Systemowe
  - ☐ Integracyjne
  - ☐ Akceptacyjne = *E2E* (*end-to-end*)
- ☐ Typy testów
  - ☐ Manualne
    - ☐ Funkcjonalne = *black box* (*jak funkcjonuje aplikacja?*)

- ☐ Automatyczne
  - ☐ Strukturalne = *white box* (*testowanie kodu*)
  - ☐ Niefunkcjonalne (*jak działa system?*)
    - ☐ Wydajnościowe, obciążeniowe, przeciążeniowe, użyteczności, pielęgnowalności, niezawodności i przenaszalności
  - ☐ Regresywne (*ponowne testowanie po modyfikacjach*)
- ☐ TDD (*Test-driven development*)
  - ☐ Cykl: faza czerwona, zielona, niebieska
- ☐ Dobre praktyki:
  - ☐ Konwencje nazywania testów
  - ☐ Testy jednostkowe wg F.I.R.S.T (*Fast, Isolated, Repeatable, Self-validating, Timely*)

☐ Frameworki webowe

- ☐ **Flask**
  - ☐ Funkcjonalność:
    - ☐ micro-framework: prosty, lekki i elastyczny
    - ☐ WSGI (*Web Server Gateway Interface*)
    - ☐ Biblioteki zewnętrzne:
      - ☐ Integracja z bazą danych: SQLAlchemy
      - ☐ Uwierzytelnianie: JWT (*JSON Web Token*), session-based authentication
  - ☐ Projekt we Flask
- ☐ **Django**
  - ☐ Funkcjonalność wbudowana:
    - ☐ Panel administratora
    - ☐ Integracja z bazą danych, system migracji i ORM (bazy danych: PostgreSQL, MySQL, SQLite oraz Oracle)
    - ☐ Zarządzanie użytkownikami i sesjami, zabezpieczenia CSRF itd.
    - ☐ Wzorzec projektowy *MVT* (*Model - View - Template*) i *MVC*
  - ☐ Projekt w Django

☐ Bazy danych

- ☐ Bazy relacyjne vs nierelacyjne
  - ☐ SQL (*Structured Query Language*)
    - ☐ Bazy relacyjne np. PostgreSQL, MySQL, SQLite, Oracle
      - ☐ Tabele i powiązań między nimi
        - ☐ *Primary Key/Foreign Key*

- ☐ Rodzaje powiązań
  - ☐ jeden do jednego
  - ☐ jeden do wielu (1 do n)
  - ☐ wiele do wielu (n do m)
- ☐ Diagramy baz danych
- ☐ CRUD (*create, read, update, delete*)
- ☐ Zapytania SQL (operacje *DML, DDL, DCL, DQL*)
- ☐ ORM (*Object-Relational Mapping*)
- ☐ Transakcje i model ACID (*atomicity, consistency, isolation, durability*)

☐ Bazy nierelacyjne = NoSQL np. MongoDB, Redis

- ☐ Model BASE (*Basically Available, Soft state, Eventual consistency*)

☐ Front end

- ☐ **HTML** (*Hypertext Markup Language*)
  - ☐ Budowa: *html, head, body*
    - ☐ *h, p, img, a, href, scr, alt, style, title …*
    - ☐ *table, tr, th, td, ul, ol, li …*
    - ☐ *div, span, class, id …*
    - ☐ Semantyczny HTML
      - ☐ *header, nav, section, article, aside, footer …*
  - ☐ Dobre praktyki stylistyczne
- ☐ **CSS** (*Cascading Style Sheets*)
  - ☐ CSS w HTML: *style="...", <head><style>...</style>* lub *href="style.css"*
  - ☐ box model, grid, flex, media queries, prefixy, praktyczne użycia
  - ☐ Preprocesory CSS (SaSS)
- ☐ **Bootstrap**
- ☐ **JavaScript**
  - ☐ DOM (*Document Object Model*)
  - ☐ Frameworki js: *React JS, Angular JS, Node.js, Vue.js*
- ☐ RWD (*Responsive Web Design*)

☐ GIT

- ☐ Zalety systemu kontroli wersji
- ☐ Struktura: *Working Directory –> Staging Area –> Repository*
- ☐ GitHub
  - ☐ *git config / log / status*
  - ☐ *git clone / branch / checkout*

- ☐ *git init / add / commit / push /*
- ☐ *git fetch / pull / merge / revert / rebase / stash*
- ☐ *README*
- ☐ *.gitignore*

☐ Inne

- ☐ **Angielski**
  - ☐ swobodne czytanie dokumentacji
  - ☐ komunikacja z współpracownikami / klientami itd.

- ☐ **Algorytmika**
  - ☐ Czym są dane i jak je przechowujemy?
  - ☐ Struktury: kopiec (stóg), kolejka (FIFO), stos (LIFO), grafy i ich reprezentacje, drzewa
  - ☐ Cykl i ścieżka Eulera, cykl Hamiltona
  - ☐ Podstawy złożoności obliczeniowej (w tym notacja-O)
  - ☐ Algorytmy sortowania:
    - ☐ insert sort – przez wstawianie,
    - ☐ selection sort – przez wymianę
    - ☐ quick sort – szybkie,
    - ☐ bubble sort – bąbelkowe,
    - ☐ heap sort – stogowe
    - ☐ counting sort – przez zliczanie,
    - ☐ merge sort – przez scalanie
  - ☐ Programowanie dynamiczne (problem plecakowy)
  - ☐ Rekurencja
  - ☐ Tablica haszująca

- ☐ **Software-development**
  - ☐ Metody
    - ☐ Programowanie zwinne: manifest *Agile*
      - ☐ *Scrum*
      - ☐ *DevOps*
    - ☐ Model kaskadowy: *Waterfall*
  - ☐ Cykl
    - ☐ *Development*
    - ☐ *CI/CD* (*Continuous integration/Continuous Delivery*)
      - ☐ *Travis*
    - ☐ *Deployment*

- ☐ ***Docker***
  - ☐ VM vs konteneryzacja
  - ☐ *Docker Image, Docker Container, Dockerfile*
    - ☐ *docker run / start / stop / -p / exec -it / build -t / -d*
  - ☐ *Docker Compose*
    - ☐ *docker-compose up / down / ps / images*

- ☐ **Internet, sieć, serwery**
  - ☐ *REST (Representational state transfer)*
  - ☐ *API (Application Programming Interface)*
  - ☐ *HTTP (Hypertext Transfer Protocol)*
    - ☐ Zapytanie = *HTTP request*
    - ☐ Metody *GET, POST, PUT, DELETE, HEAD, PATCH, OPTIONS*
    - ☐ Kod odpowiedzi = *HTTP response*
  - ☐ Model *TCP/IP (Transmission Control Protocol, Internet Protocol)*
  - ☐ Model OSI

- ☐ **Linux**
  - ☐ podstawowe komendy niezbędne do obsługi terminala