Temat 1:

Architektura Neumanna – John von Neumann stworzył komputery, które przechowywują programy. Składa się ona z:

- **CPU** (procesor) – jednostka centralna, która pobiera dane z pamięci, które są później interpretowane i wykonywane jako rozkazy:
  - **ALU** – jednostka arytmetyczno-logiczna, wykonuje operacje arytmetyczne, operacje logiczne na dwóch liczbach oraz operacje jednoargumentowe (przesunięcie bitów, negacja);
  - **CU** (układ sterowania i synchronizacji) – kontroluje pracę procesora i wytwarza sygnały potrzebne do sterowania niektórymi elementami komputera;
  - **Cache** – pamięć, która umożliwia dostęp do częstotliwości używanych poleceń i informacji.
- **REJESTRÓW:**
  - **PC** (licznik rozkazów) – zawiera adresy komórki pamięci w której przechowywany jest kod rozkazu przeznaczony do wykonania jako następny;
  - **MAR** (rejestr adresowy pamięci) – określa adres w pamięci dla kolejnej operacji odczytu lub zapisu;
  - **MDR/MBR** (rejestr buforowy pamięci) – zawiera dane, które mają być zapisane/odczytane w/z pamięci.

MASS MEMORY (pamięć masowa) – są to różne urządzenia do przechowywania dużych ilości danych:

- SSD, HDD
- Zewnętrzne dyski twarde
- Napędy optyczne
- Pendrive
- Flash memory cards (karty pamięci flash)

Pamięć RAM - pamięć główna komputera o dostępie swobodnym. Przechowuje aktualnie wykonywane programy, dane do tych programów oraz wyniki ich pracy.

DATA TYPE – klasyfikacja elementów danych, reprezentują rodzaj wartości

1. Bool
2. String
3. Float
4. Integer

Zmienna – miejsce do przechowywania wartości.

Operator – specjalne symbole, które wykonują obliczenia arytmetyczne lub logiczne

Sum (+)	Int. Division (//)
Difference (-)	Remainder (%)
Product (*)	Power (**)
Division (/)	Comparison (<>, <=, >=, ==, !=)

Temat 2

Instrukcja IF - ocenia, czy instrukcja jest prawdziwa czy fałszywa, i uruchamia kod tylko w przypadku, gdy instrukcja jest **prawdziwa**.

Instrukcja ELSE – uruchamia kod w przypadku, gdy **instrukcja if** ma wartość false.

Instrukcja ELIF – ocenia więcej niż dwa możliwe wyniki.

Pętla WHILE – wykonuje podany skrypt, póki warunek jest prawdziwy. **Pętla WHILE** może być pętlą nieskończoną, jeśli wpiszemy taki warunek, którego program nie będzie miał możliwości spełnić.

Pętla FOR – wykonuje skrypt wielokrotnie (iteracja) dla podanego warunku.

for **wartość** in **sekwencja**[string, lista etc.]:

skrypt do wykonania

Wartość – w tej sytuacji jest to zmienna, która przyjmuje wartość elementu wewnątrz sekwencji na każdej iteracji. Pętla jest kontynuowana tak długo, aż dotrze do ostatniego elementu sekwencji;

Skrypt do wykonania – jest to jakiś kod jaki chcemy wykonać przy każdej iteracji, dla przykładu może to być suma kolejnych liczb sekwencji.

Temat 3

Obsługa plików – tworzenie, odczyt, aktualizacja, usuwanie plików.

- **open()** – podstawowa funkcja do obsługi plików w Pythonie; ma 2 parametry – *nazwa pliku* i *tryb*:
  - "r" – (odczyt) - wartość domyślna. Otwiera plik do odczytu, błąd, jeśli plik nie istnieje;

    "read()" – metoda, która służy do odczytu zawartości pliku.
  - "a" – (dołącz) - otwiera plik do dołączenia, tworzy plik, jeśli nie istnieje;

    "append()" – metoda, która dodaje elementy do zawartości pliku na końcu.
  - "w" – (zapis) - otwiera plik do zapisu, tworzy plik, jeśli nie istnieje;

    "write()" – metoda, która nadpisuje istniejącą zawartość pliku.
  - "x" – (utwórz) - tworzy określony plik, zwraca błąd, jeśli plik istnieje.

RegEx (wyrażenie regularne) - sekwencja znaków tworzących wzorzec wyszukiwania. **RegEx** może służyć do sprawdzania, czy ciąg zawiera określony wzorzec wyszukiwania.

Funkcje, metaznaki???

Temat 4

Funkcja – jest to blok kodu, który działa tylko wtedy, gdy jest wywoływana. Do funkcji można przekazywać dane, znane jako parametry/argumenty -> zwraca dane

Parametr to zmienna wymieniona w nawiasach w definicji funkcji.

Argument to wartość wysyłana do funkcji, gdy jest ona wywoływana.

Return – zwraca wartość.

Zmienna globalna – zmienna, zdefiniowana na zewnątrz funkcji, co oznacza, że może być używana zarówno wewnątrz jak i na zewnątrz funkcji.

Zmienna lokalna – zmienna, zdefiniowana wewnątrz funkcji, która jest do użycia tylko wewnątrz funkcji.

__init__() to specjalna funkcja która wywoływana jest przy tworzeniu instancji klasy - można sobie w niej stworzyć potrzebne w programie zmienne, dodać jakiś kod który ma być wykonany za każdym razem gdy tworzymy nowy obiekt klasy.

self - to taka "zmienna globalna" która (gdy zostanie zdefiniowana) staje się "widoczna" w każdej metodzie klasy (głównie za sprawą tego że jest pierwszym argumentem każdej funkcji/metody w ramach klasy).

Temat 5

Moduł - plik zawierający zestaw funkcji, które chcesz uwzględnić w swojej aplikacji.

dir() - funkcja wyświetlająca wszystkie nazwy funkcji lub zmiennych w module.

nazwa_modułu.nazwa_funkcji - wywoływanie funkcji z modułu.

+ Zalety:

- Mogą być kompilowane niezależnie
- Dokonując zmiany nie musimy modyfikować całości
- Umożliwia pracę wielu użytkownikom nad jednym projektem

Temat 6-7

Programowanie obiektowe –powiązuje dane z czynnościami jakie na tych danych można wykonać

Klasa – zawiera dane ogólne, dotyczące pewnego programu

Obiekt – zawiera dane, które mają przydzielone wartości, może być wiele obiektów do jednej klasy

Metoda – funkcja, powiązana z daną klasą; jest wykonywana na rzecz konkretnego obiektu; może je odczytywać i modyfikować

Obiekt.nazwa - odniesienie do atrybutu

Atrybut – element składni języka **programowania**, który określa konkretną właściwość (znaczenie), nadaną wybranemu elementowi (obiektowi).

Klasa

Cechy	Zachowania
Charakterystyka, np. : • Nazwa • Wzrost • wiek	Funkcja; co robi, jak się zachowuje, np. : • Mówi • Biega • śpi

Temat 8

Python Inheritance (Dziedziczenie)

Dziedziczenie – mechanizm, który pozwala przekazać do poszczególnych klas informacje, że część cech oraz metod jest zdefiniowana w klasie, której są one potomkami./ Pozwala zdefiniować klasę, która dziedziczy wszystkie metody i właściwości z innej klasy.

Parent Class (Rodzic – klasa nadrzędna):

- Klasa dziedziczona
- Klasa bazowa

Child Class (Dziecko – klasa potomna):

- Klasa dziedzicząca po innej klasie
- Klasa pochodna

+ Zalety:

- Zapewnia możliwość ponownego wykorzystania kodu
- Pozwala na dodanie więcej funkcji do klasy bez jej modyfikacji

Dziedziczenie pojedyncze – gdy klasa potomna dziedziczy tylko od jednej klasy nadrzędnej

1 rodzic – 1 dziecko

Dziedziczenie wielokrotne – gdy klasa podrzędna dziedziczy po wielu klasach nadrzędnych.

N rodziców - 1 dziecko

Temat 9

Data stuctures (Struktury danych) - struktury danych to podstawowe konstrukcje, wokół których budujesz swoje programy. Każda struktura danych zapewnia określony sposób organizowania

danych, dzięki czemu można uzyskać do nich skuteczny dostęp, w zależności od przypadku użycia.

Stos – liniowa struktura danych, w której dane dokładane są na wierzch stosu i z wierzchołka stosu są pobierane.

Kolejka – liniowa struktura danych, w której nowe dane dopisywane są na końcu kolejki, a z początku kolejki pobierane są dane do dalszego przetwarzania.

Lista – przechowuje wiele elementów w jednej zmiennej. Można ją modyfikować. Może zawierać różne typy danych. (`thislist = ["apple", "banana", "cherry"]`)

`len()` - określa długość listy, tuple, dictionary, set

`list(...)` - tworzy list

`tuple(...)`- tworzy tuple

`set(...)` - tworzy set

Tuple - zbiór danych w jednej zmiennej, który jest uporządkowany i niezmienny. (`thistuple = ("apple", "banana", "cherry")`)

Set - nieuporządkowany i nieindeksowany zbiór danych. Są niezmienne i nie pozwalają na zduplikowane wartości. (`thisset = {"apple", "banana", "cherry"}`)

Dictionary - Słowniki służą do przechowywania wartości danych w parach klucz: wartość. Słownik to zbiór nieuporządkowany, zmienny i nie zezwalający na duplikaty. Do elementów można się odwoływać za pomocą nazwy klucza. Elementami mogą być różne typy danych m. in. List. (`thisdict = {`

```
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
} )
```

`update()` - metoda aktualizuje słownik z elementów z danego argumentu.

`popitem()` - sposób usuwa ostatni wkładany element.

`clear()` - Metoda opróżnia słownik.

del() - usuwa element lub cały słownik.

Wydrukuj po kolei wszystkie nazwy kluczy ze słownika:

```
for x in thisdict:
  print(x)
```

Możesz użyć `keys()` metody, aby zwrócić klucze słownika:

```
for x in thisdict.keys():
  print(x)
```

copy(), dict() - tworzy kopię słownika