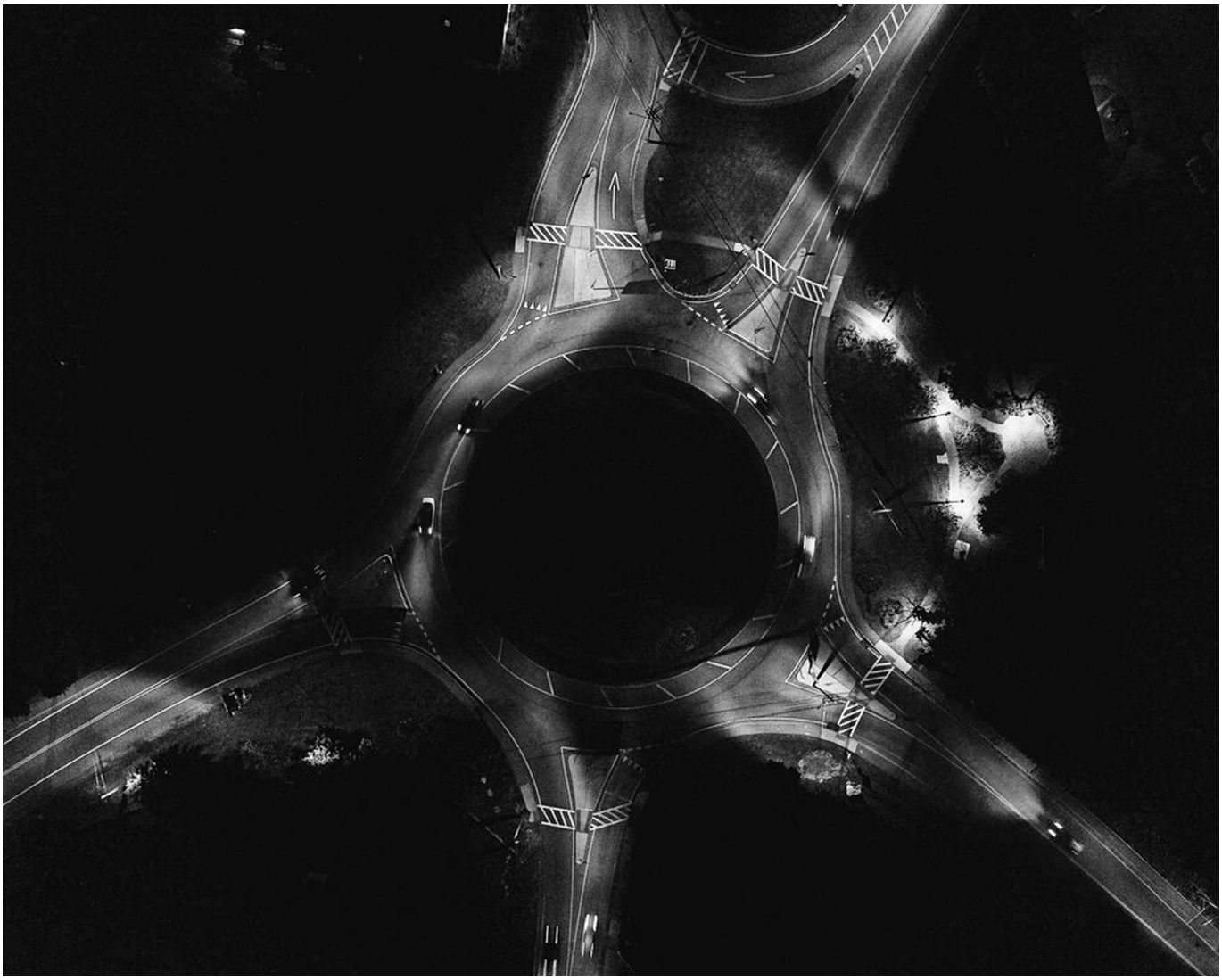


Instrukcja warunkowa w języku Python

Źródło: Osman Rana, domena publiczna.

> W tym e-materiale powtarzamy wiadomości ze szkoły podstawowej.

Wiesz już, czym charakteryzują się instrukcje warunkowe. Informacje na ten temat znajdziesz w e-materiale Instrukcja warunkowa.

W tym e-materiale zapoznasz się z ich implementacją w języku Python.

Implementacje w innych językach programowania:

- Instrukcja warunkowa w języku C++,
- Instrukcja warunkowa w języku Java.

Więcej zadań? Sięgnij do: Instrukcja warunkowa – ćwiczenia.

> **Twoje cele**

- Przeanalizujesz składnię instrukcji warunkowej w języku Python.
- Wyjaśnisz, jak zapisać warunki wielokrotnie złożone.
- Wykorzystasz instrukcję warunkową do rozwiązywania prostych problemów.

Przeczytaj

Instrukcję warunkową można zapisać przy pomocy następującego zdania: **`jeśli (warunek), to wykonaj (instrukcja)`**.

W języku Python będzie to wyglądało następująco:

```
if warunek:
    instrukcja1
    instrukcja2
    ...
```

Definiowanie warunków

Warunek może zwracać dwie wartości: prawda (`True`) lub fałsz (`False`).

Instrukcje znajdujące się po warunku zostaną wykonane tylko wtedy, kiedy warunek zwróci `True`.

Gdy w programie wykonujemy operację porównania, zwracane są wartości logiczne `True` albo `False`, np.:

```
x = 5

print(x == 5)       # zwróci True
print(x != 5)       # zwróci False
print(x == 7)       # zwróci False
print(x > 2)        # zwróci True
print(x < 5)        # zwróci False
print(x <= 5)       # zwróci True
print(x >= 5)       # zwróci True
```

> **Ważne!**

> Do przypisania zmiennej pewnej wartości używa się operatora =, natomiast do porównania ze sobą dwóch zmiennych służy podwójny znak równości ==. Operator „różne od” zapisujemy jako !=.

Przykładowa instrukcja warunkowa i jej wynik może wyglądać tak:

```
x = 5

if x < 10:
    print("Liczba jest mniejsza od 10")
```

Warunki złożone

Warunki możemy zagnieżdżać:

```
x = 5

if x < 10:
    if x > 3:
        print("Liczba jest większa od 3 i mniejsza od 10")
```

Możemy uprościć kod, wykorzystując operatory logiczne, na przykład:

```
x = 5

if x > 3 and x < 10:
    print("Liczba jest większa od 3 i mniejsza od 10")
```

> **Ważne!**
>
> Operatorem koniunkcji jest `and`, natomiast operatorem alternatywy jest `or`.

Rozszerzona postać instrukcji warunkowej

Możemy w łatwy sposób zdefiniować instrukcje (jedną lub kilka), które zostaną wykonane, kiedy warunek nie zostanie spełniony. Służy do tego słowo kluczowe `else`.

```
x = 5

if x > 3 and x < 10:
    print("Liczba jest większa od 3 i mniejsza od 10")
else:
    print("Liczba jest mniejsza od 3 lub równa 3 lub większa od 1
```

W języku Python istnieje również słowo kluczowe `elif`, który jest połączeniem operatorów `else` i `if`. Możemy za jego pomocą konstruować złożone instrukcje warunkowe, na przykład:

```
x = 5

if x > 3 and x < 10:
    print("Liczba jest większa od 3 i mniejsza od 10")
elif x < 3:
    print("Liczba jest mniejsza od 3")
elif x == 3:
    print("Liczba jest równa 3")
else:
    print("Liczba jest równa 10 lub większa od 10")
```

Po słowie kluczowym `else` została umieszczona instrukcja, która powinna zostać wykonana, jeśli żaden z warunków nie jest spełniony.

Przykładowy problem

Przyjrzyjmy się bardziej rozbudowanemu przykładowi. Wyobraźmy sobie taką sytuację: pewna fundacja przygotowuje informacje dla domów tymczasowych na temat tego, jak powinno się karmić koty. Po konsultacji z weterynarzem ustalono, że kot powinien zjadać około 60 kcal na każdy kilogram ciała. Jednak weterynarz wskazał, że sytuacja ma się inaczej w przypadku kotek, które są kotne lub karmią już młode.

Zapotrzebowanie kaloryczne kotek, które liczą ponad dwa lata oraz są w ciąży jest dwa razy większe. Podobnie kotek, które karmią jednego kociaka. Trzy razy większe zapotrzebowanie mają kotki, które karmią od dwóch kociąt. Te, które karmią od trzech do czterech, powinny zjadać cztery razy więcej kalorii. Pięciokrotność zapotrzebowania powinny zjadać zaś kotki, które karmią więcej niż pięć kociąt lub mają mniej niż dwa lata.

Korzystając z samej instrukcji warunkowej, musielibyśmy stworzyć szereg warunków:

```
kcal = waga_kg * 60

if wiek_lata > 2:
    if ciaza == true:
        kcal *= 2

    if ciaza == false:
        if mlode == 1:
            kcal *= 2

        if mlode == 2:
            kcal *= 3

        if mlode > 2
            if mlode < 5:
                kcal *= 4
            if mlode >= 5:
                kcal *= 5

if wiek_lata =< 2:
    kcal *= 5

```

Wykorzystując operatory logiczne, możemy znacznie uprościć kod obliczający, ile powinna zjeść kotka:

```
kcal = waga_kg * 60

if wiek_lata > 2 and ciaza == true or mlode == 1:
    kcal *= 2

if wiek_lata > 2 and mlode == 2:
    kcal *= 3

if wiek_lata > 2 and mlode > 2 and mlode < 5:
    kcal *= 4

if mlode >= 5 or wiek_lata < 2:
    kcal *= 5
```

Wykorzystując klauzule `else` oraz `elif`, otrzymamy taki kod:

```
kcal = waga_kg * 60

if wiek_lata > 2 and ciaza == true or mlode == 1:
    kcal *= 2
elif wiek_lata > 2 and mlode == 2:
    kcal *= 3
elif wiek_lata > 2 and mlode > 2 and mlode < 5:
    kcal *= 4
else:
    kcal *= 5
```

Słownik

> **operator logiczny**
>
> operator pozwalający na wykonywanie podstawowych operacji algebry Boole'a takich jak koniunkcja, alternatywa czy negacja

Schemat interaktywny

Polecenie 1
Uczniowie przygotowali program, który na podstawie podanych warunków sprawdza, czy osobie przysługuje zniżka.

Spróbuj napisać program przedstawiony w schemacie blokowym za pomocą języka Python. Przetestuj jego działanie dla użytkownika, który ma 25 lat, legitymację Honorowego Dawcy Krwi, ale już nie studiuje.

Specyfikacja:

Dane:

- *wiek* - liczba naturalna
- *student* - zmienna logiczna
- *honorowy_dawca_krwi* - zmienna logiczna

Wynik:

Program wyświetla komunikat informujący o tym, czy zniżka przysługuje.

Przykładowe wyjście:

```
Czy zniżka przysługuje? Tak.
```

Polecenie 2

Matka i syn planują zakup biletów lotniczych. Podstawowa cena biletu oferowanego przez linie lotnicze wynosi `cena` zł. Dopłata za przewożenie bagażu cięższego niż `limit_masy` kg wynosi `doplata_nadbagaz` zł. Ubezpieczenie bagażu to dodatkowa kwota w wysokości `ubezpieczenie_bagaz` zł – jeśli jednak pasażerowie dopłacają już za nadbagaż, to ubezpieczenie kosztuje ich tylko `ubezpieczenie_nadbagazu` zł. Osoby podróżujące klasą biznesową mają cenę zarówno nadbagażu jak i ubezpieczenia wliczoną w koszt biletu, którego cena jest wyższa o `doplata_biznes` zł od podstawowej. Rodzina nie leci klasą biznes. Matka bierze ze sobą `x` kg bagażu. Syn bierze ze sobą jedynie `y` kg bagażu. Oboje decydują się na ubezpieczenie.

Przetestuj program dla następujących danych:

```
cena = 25
limit_masy = 6
doplata_nadbagaz = 70
ubezpieczenie_bagaz = 40
ubezpieczenie_nadbagazu = 25
doplata_biznes = 150
x = 9
y = 4
klasa_biznes = FAŁSZ
ubezpieczenie = PRAWDA
```

Ile zapłacą za bilety?

Specyfikacja:

Dane:

- *cena* – liczba naturalna
- *limit_masy* – liczba naturalna
- *doplata_nadbagaz* – liczba naturalna

- *ubezpieczenie_bagaz* – liczba naturalna
- *ubezpieczenie_nadbagazu* – liczba naturalna
- *doplata_biznes* – liczba naturalna
- *x* – liczba naturalna
- *y* – liczba naturalna
- *klasa_biznes* - zmienna logiczna
- *ubezpieczenie* - zmienna logiczna

Wynik:

- *do_zaplaty* – całkowity koszt podróży

Sprawdź się

Pokaż ćwiczenia: 

Ćwiczenie 1

Napisz program, który sprawdzi, czy liczba naturalna x jest mniejsza od 10. Przetestuj działanie tego programu dla liczby `x = 7`.

Specyfikacja:

Dane:

- *x* – liczba naturalna

Wynik:

- Program wypisuje słowo *TRUE*, gdy liczba *x* jest mniejsza od 10 lub *FALSE* w przeciwnym przypadku.

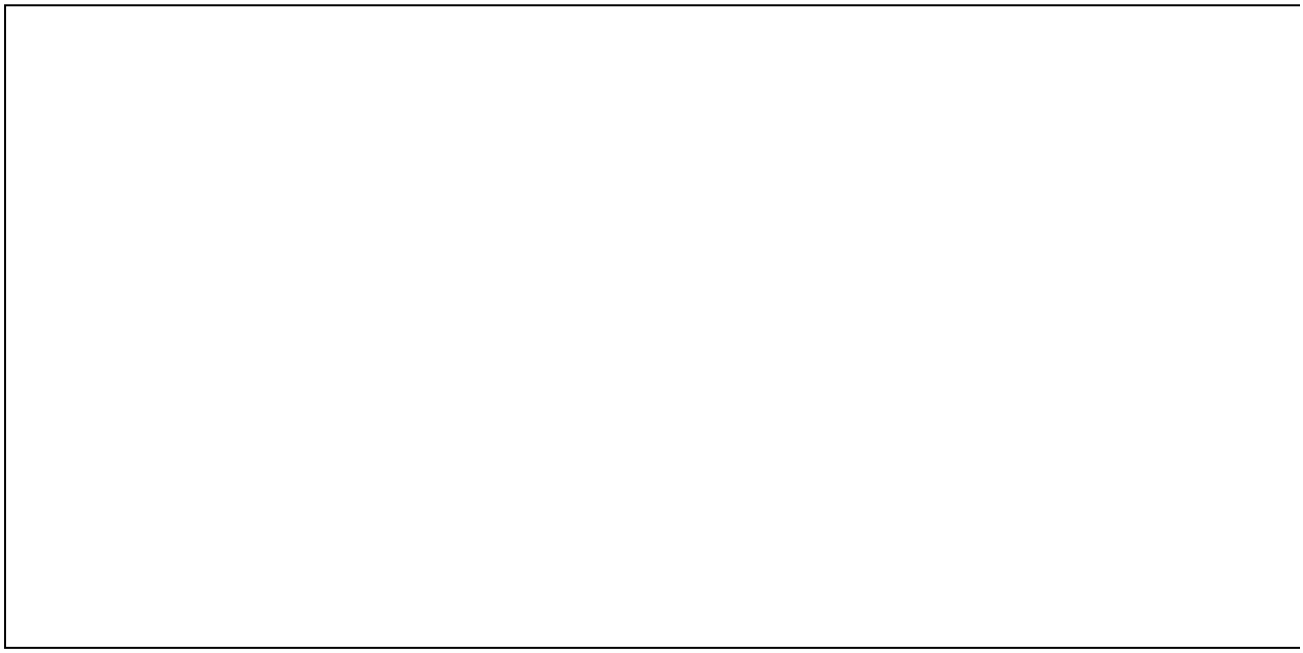
Twoje zadania

1. Utwórz funkcję `czy_liczba_mniejsza_od_10(liczba)`
2. Zwróć `True` gdy argument funkcji spełnia warunki podane w poleceniu
3. Zwróć `False` gdy argument funkcji nie spełnia podanych warunków

```
def czy_mniejsza_od_10(liczba):
    ### tutaj zmodyfikuj kod
    return True

### drukowanie testowego wyniku (dla liczby 123):
x = 7
print(str(czy_mniejsza_od_10(x)))
```

```
1
```



Ćwiczenie 2

Napisz program, który sprawdzi, czy podana liczba naturalna *x* jest parzysta, większa od 50, ale mniejsza od 100. Sprawdź jego działanie dla $x = 73$.

Specyfikacja:

Dane:

- *x* – liczba naturalna

Wynik:

- Program wypisuje słowo *TRUE*, gdy liczba *x* jest parzysta, większa od 50 i mniejsza od 100; *FALSE* w przeciwnym przypadku.

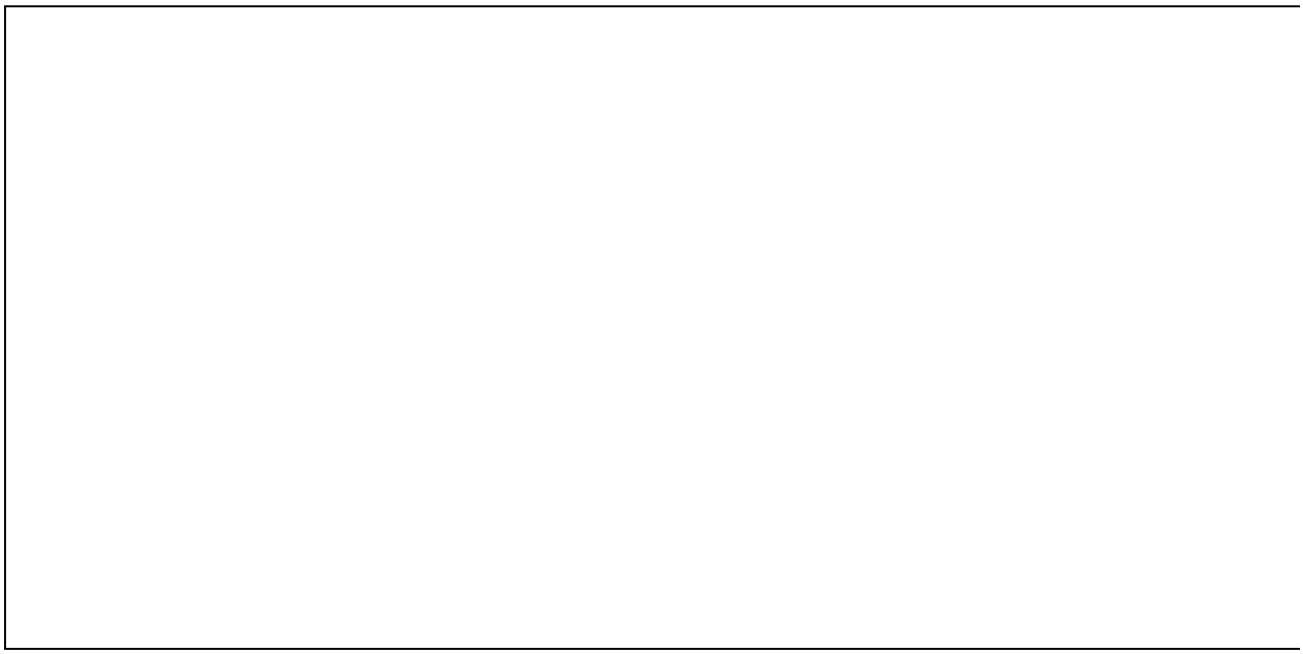
Twoje zadania

1. Utwórz funkcję `test(liczba)`
2. Zwróć `True` gdy argument funkcji spełnia warunki podane w poleceniu
3. Zwróć `False` gdy argument funkcji nie spełnia podanych warunków

```
def test(liczba):
    ### tutaj zmodyfikuj kod
    return True

### drukowanie testowego wyniku (dla liczby 73):
x = 73
print(str(test(x)))
```

```
1
```



Ćwiczenie 3

Uczniowie pisali sprawdzian z informatyki, z którego mogli uzyskać maksymalnie 100 punktów. Punktacja przedstawiała się następująco:

- 0–39 pkt – ndst
- 40–54 pkt – dop
- 55–69 pkt – dst
- 70–84 pkt – db
- 85–98 pkt – bdb
- 99–100 pkt – cel

Napisz program, który wypisze ocenę na podstawie zdobytych punktów. Przetestuj jego działanie dla wyniku 55.

Specyfikacja:

Dane:

- *punkty* – liczba naturalna z przedziału <0, 100>

Wynik:

- Program wypisuje ocenę w zależności od liczby uzyskanych punktów.

Twoje zadania

1. Utwórz funkcję `ocena(punkty)`
2. Zwróć `cel` gdy argument funkcji jest w przedziale 99-100
3. Zwróć `bdb` gdy argument funkcji jest w przedziale 85-98
4. Zwróć `db` gdy argument funkcji jest w przedziale 70-84
5. Zwróć `dst` gdy argument funkcji jest w przedziale 55-69
6. Zwróć `dop` gdy argument funkcji jest w przedziale 40-54

7. Zwróć `ndst` gdy argument funkcji jest w przedziale 0-39
8. Zwróć `ndst` gdy argument funkcji nie znajduje się w żadnym z podanych przedziałów

```
def ocena(punkty):
    ### tutaj zmodyfikuj kod
    return "ndst"

### drukowanie testowego wyniku (dla 55 punktów):
print(ocena(55))
```

```
1
```

Ćwiczenie 4

Napisz program, który znajduje miejsca zerowe funkcji kwadratowej. Do obliczenia pierwiastka wykorzystaj polecenie `math.sqrt(liczba)`.

Specyfikacja:

Dane:

- *a*, *b*, *c* – liczby rzeczywiste

Wynik:

- *x1*, *x2* – miejsca zerowe funkcji $y = ax^2 + bx + c$

Twoje zadania

1. Utwórz funkcję `miejsca_zerowe(a, b, c)`
2. Zwróć `blad` gdy argument a jest równy 0 (czyli kiedy nie jest to funkcja kwadratowa)
3. Zwróć `brak` gdy funkcja nie posiada miejsc zerowych
4. Jeśli funkcja kwadratowa posiada jedno miejsce zerowe, to zwróć tę liczbę
5. Jeśli funkcja kwadratowa posiada dwa miejsca zerowe to zwróć je za pomocą gotowego zapisu `return [x1, x2]`

```
import math

def miejsca_zerowe(a, b, c):

    delta =
    x1 =
    x2 =

    if (delta > 0):
        return [x1, x2]
    elif: #uzupelnij warunek elif
        return x1
```

```
13     else:
```

```
1
```

Dla nauczyciela

Autor: Jakub Kamiński

Przedmiot: Informatyka

Temat: Instrukcja warunkowa w języku Python

Grupa docelowa:

Szkoła ponadpodstawowa, liceum ogólnokształcące, technikum, zakres podstawowy i rozszerzony

Podstawa programowa:

Cele kształcenia – wymagania ogólne

I. Rozumienie, analizowanie i rozwiązywanie problemów na bazie logicznego i abstrakcyjnego myślenia, myślenia algorytmicznego i sposobów reprezentowania informacji.

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera oraz innych urządzeń cyfrowych: układanie i programowanie algorytmów, organizowanie, wyszukiwanie i udostępnianie informacji, posługiwanie się aplikacjami komputerowymi.

III. Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi, w tym: znajomość zasad działania urządzeń cyfrowych i sieci komputerowych oraz wykonywania obliczeń i programów.

Treści nauczania – wymagania szczegółowe

I. Rozumienie, analizowanie i rozwiązywanie problemów.

Zakres podstawowy. Uczeń:

1) planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komputacyjnego (określenie problemu, definicja modeli i pojęć, znalezienie rozwiązania, zaprogramowanie i testowanie rozwiązania).

3) wyróżnia w problemie podproblemy i charakteryzuje: metodę połowienia, stosuje podejście zachłanne i rekurencję;

4) porównuje działanie różnych algorytmów dla wybranego problemu, analizuje algorytmy na podstawie ich gotowych implementacji;

5) sprawdza poprawność działania algorytmów dla przykładowych danych.

Zakres rozszerzony. Uczeń spełnia wymagania określone dla zakresu podstawowego, a ponadto:

3) objaśnia dobrany algorytm, uzasadnia poprawność rozwiązania na wybranych przykładach danych i ocenia jego efektywność;

II. Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych.

Zakres podstawowy. Uczeń:

1) projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych; w szczególności programuje algorytmy z punktu I.2);

2) do realizacji rozwiązań problemów prawidłowo dobiera środowiska informatyczne, aplikacje oraz zasoby, wykorzystuje również elementy robotyki;

Kształtowane kompetencje kluczowe:

- kompetencje cyfrowe;
- kompetencje osobiste, społeczne i w zakresie umiejętności uczenia się;
- kompetencje matematyczne oraz kompetencje w zakresie nauk przyrodniczych, technologii i inżynierii.

Cele operacyjne (językiem ucznia):

- Przeanalizujesz składnię instrukcji warunkowej w języku Python.
- Wyjaśnisz, jak zapisać warunki wielokrotnie złożone.
- Wykorzystasz instrukcję warunkową do rozwiązywania prostych problemów.

Strategie nauczania:

- konstruktywizm;
- konektywizm.

Metody i techniki nauczania:

- dyskusja;
- rozmowa nauczająca z wykorzystaniem multimedium i ćwiczeń interaktywnych;
- ćwiczenia praktyczne.

Formy pracy:

- praca indywidualna;
- praca w parach;
- praca w grupach;
- praca całego zespołu klasowego.

Środki dydaktyczne:

- komputery z głośnikami, słuchawkami i dostępem do internetu;
- zasoby multimedialne zawarte w e-materiale;
- tablica interaktywna/tablica, pisak/kreda.

Przebieg lekcji

Przed lekcją:

1. **Przygotowanie do zajęć.** Nauczyciel loguje się na platformie i udostępnia e-materiał: „Instrukcja warunkowa w języku Python". Uczniowie zapoznają się z treściami w sekcji „Przeczytaj".

Faza wstępna:

1. Przedstawienie tematu i celów zajęć.

Faza realizacyjna:

1. **Praca z multimedium.** Nauczyciel wyświetla zawartość sekcji „Schemat interaktywny". Uczniowie rozwiązują Polecenie 1 oraz Polecenie 2.
2. Chętne lub wybrane osoby prezentują swój kod na forum klasy. Uczniowie dyskutują na jego temat.

Faza podsumowująca:

1. Uczniowie w parach wykonują ćwiczenia nr 1–3 z sekcji „Sprawdź się".
2. Nauczyciel zadaje pytania podsumowujące, np.
    - w jakich sytuacjach instrukcja warunkowa nie jest potrzebna?
    - w jakich sytuacjach instrukcja warunkowa jest konieczna?
3. Na koniec zajęć nauczyciel prosi uczniów o rozwinięcie zdania: „Na dzisiejszych zajęciach nauczyłam/łem się jak…".

Praca domowa:

1. Wykonaj ćwiczenie nr 4 z sekcji „Sprawdź się".

Materiały pomocnicze:

- Oficjalna dokumentacja techniczna dla języka Python 3 (lub nowszej wersji).

- Oficjalna dokumentacja techniczna dla oprogramowania PyCharm lub IDLE.

Wskazówki metodyczne:

- Treści w sekcji „Schemat interaktywny” można wykorzystać na lekcji jako podsumowanie i utrwalenie wiedzy uczniów.