Programowanie aplikacji geoinformatycznych (Python)

Działania na podstawowym środowisku Python

1. Stwórz pusty folder i otwórz go w wybranym IDE.
2. Jako interpreter wybierz „czystą” wersję Pythona.
3. W folderze stwórz pierwszy plik z kodem źródłowym o nazwie *skrypt1.py*.
4. Dodaj w nim pierwszą linię kodu *print(“Hello World”)* i uruchom skrypt wskazanym interpreterem.
5. Wypisz wartość funkcji *help(print))*.

Z komentarzem [JS1]: Sposób na uzyskanie pomocy dot. wykorzystywania danej funkcji.

Oczekiwany rezultat:

```
C:\Users\jas\prog-apl-geoinf>C:/Users/jas/AppData/Local/Programs/Python/Python310/pyth
on.exe c:/Users/jas/prog-apl-geoinf/skrypt1.py
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file:  a file-like object (stream); defaults to the current sys.stdout.
    sep:   string inserted between values, default a space.
```

Moduły i przestrzenie nazw

1. Z pakietu *os* zaimportuj jedną funkcję o nazwie *getcwd*.
2. Wywołaj funkcję *getcwd* i jej wynik przypisz do zmiennej *current_path.*
3. *Wypisz wartość zmiennej* current_path.
4. W tym samym folderze stwórz dodatkowy plik o nazwie *czas.py.*
5. W pliku *czas.py* dodaj zmienną i nazwij ją *aktualny_czas*, przypisz do niej wartość funkcji *datetime.now()*, która pochodzi z modułu datetime z pakietu datetime.
6. Zaimportuj moduł *czas* w skrypcie *skrypt1.py*.
7. Wypisz wartość zmiennej *aktualny_czas*.
8. Zaimportuj pakiet *time*.
9. Jako kolejną funkcję w skrypcie użyj *time.sleep(20)*.
10. Ponownie wypisz wartość zmiennej *aktualny_czas*.
11. Przeładuj moduł *czas*.
12. Po raz trzeci wypisz wartość zmiennej *aktualny_czas*.
13. Zwróć uwagę na to, kiedy zmieniła się wartość zmiennej. Zastanów się, kiedy inicjowana jest zmienna *aktualny_czas*.

Z komentarzem [JS2]: Funkcję znajdź w internecie, zwróć uwagę na swoją wersję Pythona.

Oczekiwany rezultat:

```
C:\Users\jas\prog-apl-geoinf>C:/Users/jas/AppData/Local/Programs/Python/Python310/pyth
on.exe c:/Users/jas/prog-apl-geoinf/skrypt1.py
C:\Users\jas\prog-apl-geoinf
2024-03-03 22:17:55.163925
2024-03-03 22:17:55.163925
2024-03-03 22:18:15.175109
```

Podstawowe typy wbudowane

<Pamiętaj, że dobrą praktyką jest importowanie modułów i pakietów na samym początku plików>

<Na różnych etapach tego zadania wykorzystaj funkcję *type*(), która pozwala zauważyć, jak działa dynamiczne typowanie w Pythonie>

1. Stwórz nowy plik o nazwie *skrypt2.py*. Wszystkie kolejne instrukcje wprowadzaj po kolei w tym pliku.

Działania matematyczne

2. Stwórz zmienną *wartosc o wartości 100*
3. Do zmiennej *dodawanie* przypisz wartość dodania do zmiennej *wartosc* liczby 123.15.
4. Stwórz zmienną *potega* i przypisz do niej zmienną *dodawanie* podniesioną do potęgi 12345.
5. Do zmiennej *tekst* przypisz rzutowanie wartości zmiennej *potega* na typ string.
6. Do zmiennej *wartosc_pi* przypisz wartość pi.
7. Do zmiennej *losowa* przypisz losową wartość z listy *[1,2,3,4,5]*.

Łańcuchy znaków

<łańcuchy znaków to typ sekwencyjny, więc można się odwoływać do poszczególnych znaków>

8. Nadpisz zmienną *tekst* następującym wyrażeniem: *tekst = f"Wartosc: {tekst}"*
9. Wyświetl długość tekstu w zmiennej *tekst* i później wykorzystując wycinki wyświetl część zmiennej *tekst* o wartości „art”.
10. Wypisz wartość funkcji *dir(tekst)*.
11. Zmień cały łańcuch znaków w zmiennej *tekst* na wielkie litery, wypisz.
12. Spróbuj zamienić znak na pozycji 2 w łańcuchu w zmiennej *tekst* na znak p.

Oczekiwany rezultat:

```
Dlugosc zmiennej tesk: 30, wykorzystanie wycinkow: art
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__
', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__
', '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt
__', '__mod__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr
__', '__rmod__', '__rmul__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__
', 'capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'f
ind', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', '
isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle',
 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefi
x', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rs
trip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate',
 'upper', 'zfill']
Po zmianie na wielkie liter: WARTOSC: 5.578044006646277E+31
Traceback (most recent call last):
  File "c:\Users\jas\prog-apl-geoinf\skrypt2.py", line 19, in <module>
    tekst[2] = "p"
TypeError: 'str' object does not support item assignment
```

Listy

<elementy wewnątrz listy nie muszą być tego samego typu; listy mogą być zagnieżdżone>

13. Działania na listach:
    - Stwórz zmienną o nazwie *lista*, przypisz do niej rzutowanie na listę zmiennej *tekst*.
    - Wykorzystując wycinki zrób tak, żeby lista zawierała jedynie litery słowa WARTOSC i później dwukropek.
    - Do listy dodaj kolejny wyraz, niech będzie to kolejna lista [1,2,3,4,5].
    - Z listy usuń wyraz, który jest dwukropkiem.
    - Wypisz zmienną *lista*.
14. Listy składane (list comprehension):

Z komentarzem [JS3]: Pojawił się błąd przy uruchomieniu? Zmień wartość potęgi na 12.

Z komentarzem [JS4]: Odpowiednią funkcję możesz znaleźć w internecie.

Z komentarzem [JS5]: Odpowiednią funkcję możesz znaleźć w internecie.

Z komentarzem [JS6]: Wyszukaj w internecie i spróbuj zrozumieć jak działają formatted string.

Z komentarzem [JS7]: Wyświetla ona wszystkie dostępne atrybuty dla tej zmiennej - to może być przydatne jeśli będzie Ci brakować wiedzy co można zrobić z danym typem zmiennej.

Z komentarzem [JS8]: To nie powinno się udać, powinien pojawić się błąd: TypeError: 'str' object does not support item assignment -> należy zwrócić uwagę, że nie wszystkie typy wspierają różne działania.

Z komentarzem [JS9]: Do wykonania tych funkcji wykorzystaj operacje specyficzne, możesz je wylistować używając funkcji dir().

- Stwórz zmienną jak tutaj: *lista2 = [1,2,3,"banan",100].*
- Jako zmienna lista3, wykorzystaj składnię listy składanej, przeiteruj po każdym wyrazie z listy, do nowej listy zapisz wartość podniesioną do potęgi 2, jeśli wartość jest równa "banan" to ją pomiń.
- Stwórz *lista4*, wykorzystaj funkcję *range()*, ma ona zawierać co drugą liczbę od 2 do 16.
- Wypisz zmienne *lista2*, *lista3* i *lista4*.

Oczekiwany rezultat:

```
['W', 'A', 'R', 'T', 'O', 'S', 'C', [1, 2, 3, 4, 5]]
lista2: [1, 2, 3, 'banan', 100], lista3: [1, 4, 9, 10000], lista4: [2, 4, 6, 8, 10, 12, 14, 16]
```

Słowniki

<obiekty typu słownik posiadają argumenty, każdy z nich musi mieć klucz oraz wartość>

15. Stwórz pusty słownik o nazwie *ja*.
16. Niech będzie to reprezentacja Twojej osoby, dodaj do niego klucze *imie*, *nazwisko*, *wiek*, *rodzice* (rodzice mają być reprezentowani przez listę z 2 zagnieżdżonymi słownikami o 2 kluczach: *imie* i *wiek*.
17. Wypisz wartość klucza *rodzice*.
18. Wypisz jedynie imię pierwszego z rodziców.
19. Wypisz wszystkie klucze naszego słownika.
20. Sprawdź czy nasz słownik posiada klucz *rodzenstwo*, wypisz zmienną typu boolean.

Oczekiwany rezultat:

```
[{'imie': 'Elzbieta', 'wiek': 48}, {'imie': 'Jan', 'wiek': 51}]
Elzbieta
dict_keys(['imie', 'nazwisko', 'wiek', 'rodzice'])
False
```

Krotki

<bardziej znana jest raczej nazwa angielska – tuples>

<krotki są jak takie listy, ale bez możliwości modyfikacji>

21. Do zmiennej *krotka1* przypisz wartość (1,2,"3",4,2,5).
22. Wypisz długość zmiennej i pierwszy wyraz.
23. Sprawdź, ile razy występuje wartość 2 i wypisz.
24. Spróbuj zmienić pierwszy wyraz na wartość 2.

Z komentarzem [JS10]: Służy do tego jedna z operacji specyficznych.

Oczekiwany rezultat:

```
Dlugosc: 6, pierwszy wyraz: 1
Wartosc 2 wystepuje 2 razy
```

Zbiory

<dosyć podobne do pozostałych, można je porównać do listy kluczy słownika>

<ich ciekawą funkcjonalnością jest to, że w zbiorach nie mogą się pojawiać duplikaty?

25. Stwórz dwa zbiory o nazwach *X* i *Y*, nadaj im wartości odpowiednio: *set("kalarepa") oraz set("lepy").*

26. Wyświetl część wspólną obu zbiorów - można na nich wykonywać podobne operacje jak na zbiorach matematycznych.

Oczekiwany rezultat:

```
{'e', 'p', 'l'}
```

Instrukcje

1. Napisz program, który iteruje przez listę imion używając pętli *for* oraz funkcji *enumerate()*, aby wyświetlić indeks każdego imienia wraz z samym imieniem.
2. Stwórz przykłady dla testów *if*:
    a. Gdzie wystąpią dwa warunki - napisz program sprawdzający czy dana liczba jest dodatnia i parzysta. Jeśli tak, program powinien wydrukować "Liczba jest dodatnia i parzysta".
    b. Gdzie wykorzystane zostanie zaprzeczenie *not* lub =! - napisz program, który sprawdza, czy wprowadzona przez użytkownika liczba nie jest równa zero. Jeśli nie jest, wydrukuj "Liczba jest różna od zera".
    c. Gdzie wykorzystane będzie słowo *in* - napisz program, który sprawdza, czy wprowadzony przez użytkownika owoc znajduje się na liście dostępnych owoców (np. ['jabłko', 'banan', 'pomarańcza']). Jeśli tak, program powinien wydrukować "Owoc jest dostępny".
3. Stwórz przykład z pętlą *while* - Stwórz program, który będzie ciągle prosił użytkownika o wprowadzenie liczby. Program powinien sumować wprowadzone liczby i kończyć działanie, gdy suma przekroczy 100. Po zakończeniu pętli, program powinien wydrukować sumę wprowadzonych liczb.

"Dziwactwa"

<w tych przypadkach, zapoznaj się z kodem, wyzwól go i zastanów się co się dzieje>

1. Przypisanie tworzy referencje, a nie kopie

```
L = [1,2,3,4]
M = [1,2,3,L,4]
print(f"Wartość zmiennej M przed zmianą L: {M}")
L[1] = "woooow"
print(f"Wartość zmiennej M po zmianie L: {M}")
```

2. Powtórzenie dodaje jeden poziom zagłębienia

```
L = [4,5,6]
X = L * 4
Y = [L] * 4
print(f"X: {X}, Y: {Y}")
L[1] = "wow"
print(f"X: {X}, Y: {Y}")
```

```
L = [4,5,6]
Y = [list(L)] * 4
L[1] = "wow"
print(f"Y: {Y}")
Y[0][1] = "wow"
```

Z komentarzem [JS11]: Tak wygląda operacja wyciągająca część wspólną: X & Y. Po inne można sięgnąć do dokumentacji.

Z komentarzem [JS12]: Tutaj można wykorzystać funkcję input().

```
print(f"Y: {Y}")
```

Praca z plikami

<przy pracy z plikami warto stosować managery kontekstu (za pomocą słowa with), które dbają o poprawne otwarcie i zamknięcie zasobów>

```
with open('example.txt', 'r') as file:
  content = file.read()
```

Zadania sprawdzające

<w tym zadaniu przydać się zdecydowanie mogą operacje specyficzne dla różnych typów>

Zadanie 1

Wczytaj jako słownik plik z rozszerzeniem JSON (przydatny może okazać się pakiet *json*). Zapisz do zmiennej połączone wszystkie teksty z pliku. Zmodyfikuj następująco ten tekst:

- Zamień wszystkie duże litery na małe,
- Podziel go na wyrazy - będzie to najprawdopodobniej lista,
- Usuń znaki interpunkcyjne – w tekście występują jedynie kropki i przecinki,
- Zmodyfikuj tak każdy wyraz, żeby w każdym ostatni znak był w formacie dużej litery np. wyraz KozA,
- Z listy usuń wyrazy, które nie posiadają w sobie znaku *a* lub *A* - można wykorzystać składnię list składanych,
- Stwórz zmienną, które będzie przechowywać wszystkie unikatowe wyrazy - można wykorzystać zbiory,
- Stwórz zmienną, która będzie przetrzymywać ilość wystąpień dla każdego ze słów występujących w tekście - można wykorzystać słowniki.

Zapisz stworzone zmienne do pliku JSON, wartości kluczy wybierz samodzielnie.