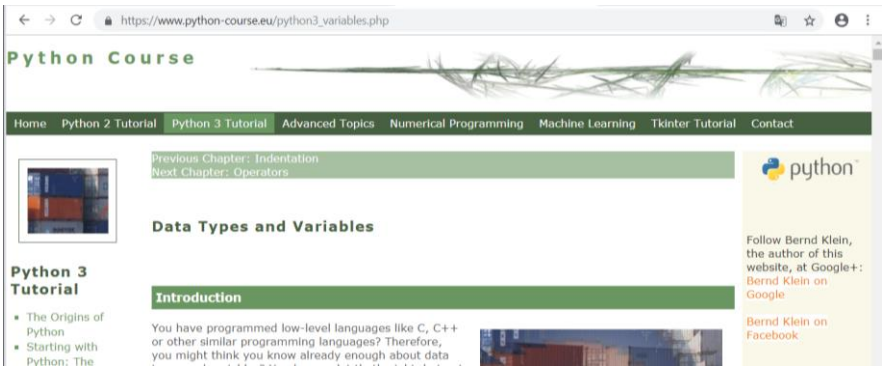
PROJEKT B 17. 10.2018

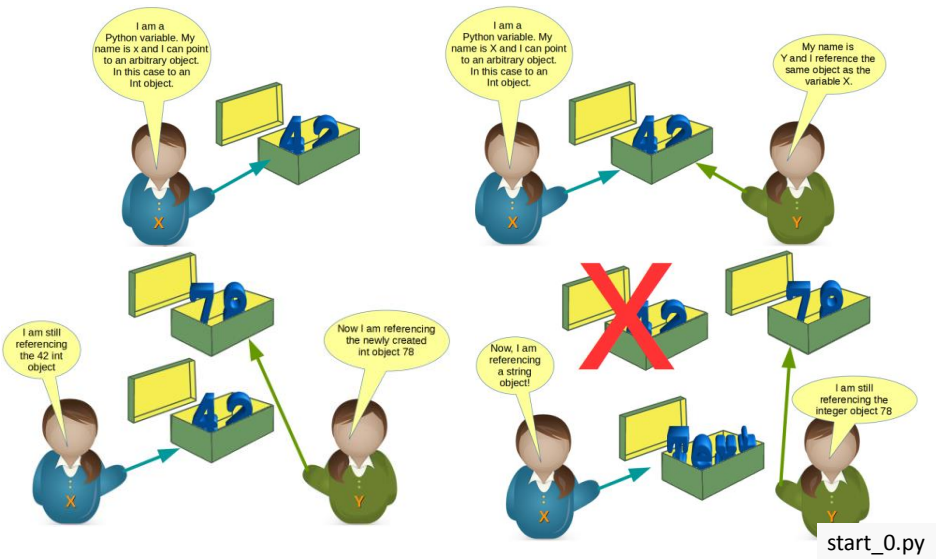
TERMIN ROZLICZENIA 29.10.2018

PYTHON: WPROWADZENIE
TYPY DANYCH, ZMIENNE, PODSTAWOWE WYRAŻENIA



Dynamiczne typy zmiennych

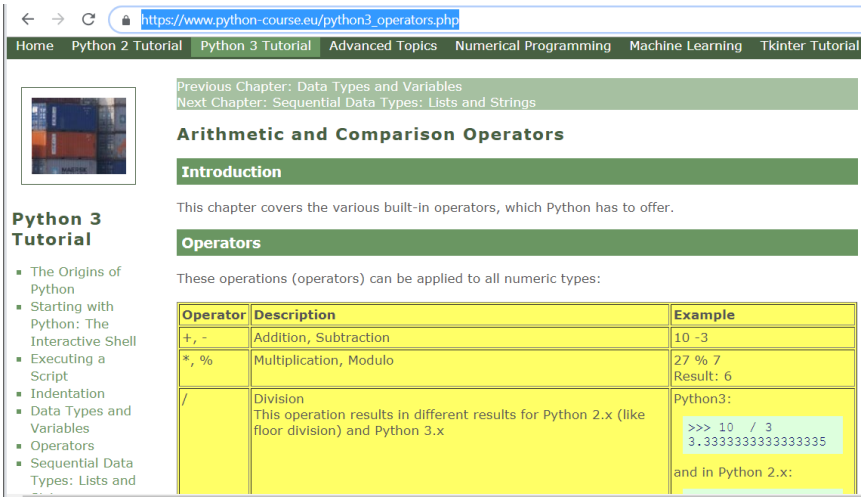
Wartości a nie zmienne mają TYP. Zmienna to (1) nazwa i (2) referencja do danego obiektu, czyli wskaźnik do wartości.



start_0.py

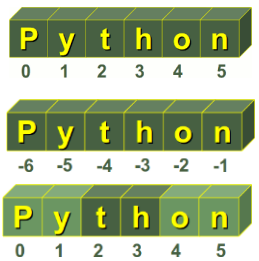
Operatory

https://www.python-course.eu/python3_operators.php



start_1.py

Typy danych



Typy wbudowane: liczby,
łańcuchy napisowe,
kolekcje: listy, krotki, słowniki zbiory

Typy danych mogą być zmienne (modyfikowalne) lub niezmienne (niemodyfikowalne).
Typy niemodyfikowalne to łańcuch czy krotka.

Łańcuch: sekwencja znaków,

- można oglądać od przodu, od tyłu, fragmentem,
- można do nich dołączać inne znaki
- nie można 'edytować'

```
#Zmienna jako referencja:
x= ["ala", "ola", "ela"]

y_1=3*x
y_2=3* [x]

x[1]="ula"
```

Zbadaj efekt?

start_2.py

start_3.py

Sekwencyjne typy danych

- *Lista (list):* sekwencja uporządkowana o różnej zawartości. Elementy listy są dostępne poprzez indeks, mogą być dowolnie zagnieżdżone, mogą zmieniać rozmiar, elementy listy mogą być modyfikowane.
  L=[1234, 'ala', 'ola', 34, ['ola', 34]]

- *Krotka (tuple):* sekwencja uporządkowana (ponumerowana) różnych zawartości, ale niemutowalnych (modyfikowalnych) zawartości. Zdecydowanie szybsza w działaniu niż lista
  T = 1234, 'ala', 'ola' , 34

- *Słownik (dictionary)*: nieuporządkowany zbiór par (klucz: wartość) przy czym klucze muszą być unikatowe.
  D= { 2466 : 'Phyton', "abc" : ' Phyton', ', (1,2,3) : ' Phyton' }

- *Plik (file):* interfejs do zasobów zewnętrznych. Tworzymy funkcją open(), zamykamy close().
  text_file = open(nazwa_pliku, mode)
  mode: "r", "w", "a", "r+", "w+", "a+", "b",

Plik i poprawność przywiązania

```
def czytaj_plik(nazwa_pliku):
    try:
        plik = open(nazwa_pliku, 'r')
        zawartosc = plik.read()
    except IOError:
        print ("\n\nBłąd  dostępu do pliku. STOP? ")
        zawartosc={}
    return zawartosc
```

start_4.py

Poprawność implementacji

Praktycznie, wszystkie implementacje mają błędy. Im szybciej znajdzie się błąd, tym mniej czasu się zmarnuje. Propozycja testu:

assert 2 + 2 == 5, " z dodawaniem nie jest dobrze"

start_5.py

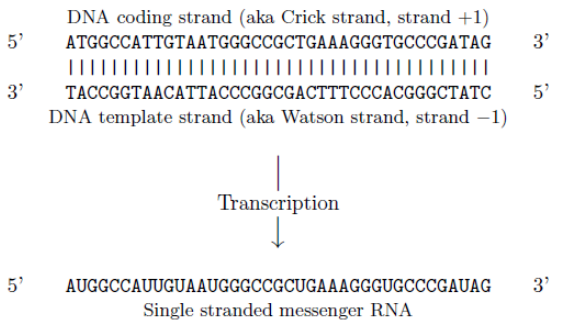
```
def get_AT_content ( dna, ile_cyfr=2):
        dna = dna.replace('N', '')
        length = len(dna)
        A_count = dna.upper().count('A')
        T_count = dna.upper().count('T')
        AT_content = (A_count + T_count) / length
        return round(AT_content, ile_cyfr)

assert get_AT_content("ATGCNNNNNNNNNN") == 0.5, 'error: zle zamienia N'
assert get_AT_content("AAAA") == 1 ,  'error: zle zlicza A '
assert get_AT_content("G") == 0 , 'error: zle zlicza braki'
assert get_AT_content("ATGC") == 0.5,  'error: zle liczy '
assert get_AT_content("AGG") == 0.33,  'error: zle liczy A '
assert get_AT_content("AGG", 1) == 0.3 , 'error: zle przybliza '
assert get_AT_content("AGG", 5) == 0.33333, 'error: zle przybliza '
```

Zasady przygotowanie projektów w Pythonie:

1. Projekt powinien dobrze działać w pracowni komputerowej
2. Ze względu na problemy z kodowaniem polskich liter nie używać polskich liter.
3. Każdą instrukcję umieszczamy w oddzielnym wierszu
4. Funkcje mają być krótkie (jeden ekran) i wykonywać jedną rzecz. Optymalna liczba zmiennych lokalnych to 7 (max 10) zmiennych. Zmiennych globalnych używamy wyłącznie tam, gdzie jest to konieczne.
5. Proszę wstawiać komentarze, które wyjaśniają, co robi program i jak to robi (syntetyczny opis algorytmu), co robi funkcja (syntetyczne czemu służy).
6. Programom zawsze muszą towarzyszyć testy poprawności.

Uwagi dodatkowe:

1. Nie używać słów kluczowych do nazywania zmiennych
2. Używać pustych linii do rozdzielenia funkcji
3. Wstawiać spacje dookoła operatorów i po przecinkach, wyjątek nawias '('.



PROJEKT B 17. 10.2018

1. Dany jest napis wielowierszowy: TEKST
   - Podać sposób obliczenia liczby wyrazów w napisie TEKST (wyraz to ciąg ”czarnych znaków” przedzielonych białymi: spacja, tabulacja , nowa linia)
   - Zbudować napis składający się z prefiksów o rozmiarze k z wyrazów napisu TEKST
   - Zbudować napis składający się z sufiksów o rozmiarze k z wyrazów napisu TEKST
   - Znaleźć najdłuższe i najkrótsze słowo w napisie TEKST
   - Uporządkować wszystkie wyrazy w napisie TEKST
     - Alfabetycznie
     - Pod względem długości słów.
2. Dla wygenerowanej losowo sekwencji nukleotydów o długości n: DNA_strand
   - Utworzyć nić komplementarną
   - Przeprowadzić transkrypcję do messanger RNA , mRNA
   - W oparciu o kod genetyczny przygotować translację mRNA na sekwencje aminokwasów. Rozważyć wszystkie możliwości startu.

Wszystkie wyniki przetwarzania proszę umieścić w plikach tekstowych.