Лабораторна робота
Тема: Основи Python

Прості типи даних
По суті все, чим займається комп'ютер як обчислювальна машина, -- це обробка даних. У високорівневих мовах програмування всі дані належать до якихось вбудованих типів. Найпростішими з них є рядки та числа (цілі та дійсні розглядаються окремо, так як реалізуються в комп'ютері по-різному), які просто являють собою текстові фрагменти або числові значення.
Всі вони відображаються в python по-різному і, якщо ви задаєте змінній якесь значення за його зовнішнім виглядом інтерпретатор може визначити тип даних, який мається на увазі. У випадку, коли інтерпретатор не в змозі самостійно правильно визначити тип, програміст може конвертувати значення у необхідний тип даних примусово. Для цього застосовуються вбудовані функції, які, подібно до математичних функцій, приймають аргумент - дані, які потрібно конвертувати, та повертають перетворене значення необхідного типу.
Int -- цілі числа: 1, 2, 0, -10, 9999 і т.д. Відображаються просто як числа. Для перетворення будь-якого значення на ціле число використовується функція int():
```
x_float = 1.0
x_int = int(x_float)
```
Float -- дійсні числа: 1.0, 2.543, 0.0, -0.99999, 123123123.4213 і т.д. Виглядають як 2 числа (ціла і дробова частина), розділені крапкою. Для перетворення будь-якого значення на дійсне число використовується функція float():
```
x_str = '1'
x_float = float(x_str)
```
Str -- рядки: 'High hopes', "The great gig in the sky", "When You're In" і т.д. Виглядають як текстові фрагменти будь-якої довжини -- від пустого рядка і до нескінченності (при цьому, зверніть увагу, що консоль/термінал, у якій ми вводимо рядок може мати власне обмеження довжини команди і просто не дозволить ввести надто великий рядок за 1 раз) -- взяті в одинарні чи подвійні лапки. Для перетворення будь-якого значення у рядок використовується функція str():
```
x_int = 1
x_str = str(x_int)
```
Звичайно, не всі дані можна привести до будь-яких типів. Пропоную вам поекспериментувати із цим самостійно.

Арифметичні дії
Числа
Для чисел (як цілих так і дійсних) передбачено стандартні математичні операції: додавання (+), віднімання (-), множення (*) та ділення (/). А крім того піднесення у степінь (**), ділення націло (//) та взяття остачі (%) від ділення націло. Першим у виразі обчислюється піднесення у степінь, потім операції множення, ділення та остачі від ділення, потім додавання та віднімання. Операції з однаковим пріоритетом обчислюються зліва направо. Для зміни порядку проведення операцій, як і в математиці, використовуються круглі дужки.
```
print (2+2)**2 % 10
```
- виведе 6
При цьому, якщо хоч одне із значень у виразі є дійсним числом, для проведення розрахунків інтерпретатору доведеться конвертувати значення, які з ним взаємодіють, і результат також буде дійсним, навіть за відсутності у нього дробової частини.
```
print (2+2)**2 % 10.0
```
- виведе 6.0
І навпаки, якщо операнди є цілими, результат також буде цілим. Таким чином, ділення двох цілих чисел фактично також є операцією "ділення націло":

```
print 10/3
```
- виведе 3

І ще раз звертаю вашу увагу: перетворення типів при арифметичних обчисленнях виконується оператором лише у разі необхідності -- якщо один з двох операндів дії є дійсним, або ви примусово вказуєте, що необхідно конвертувати тип даних. Тому за цим слід уважно слідкувати:

```
print 3/2 * 1.0
```
- виведе 1.0
```
print 3.0/2 * 1
```
- виведе 1.5

Рядки

Рядки можна додавати між собою -- ця операція називається **конкатенацією**.

```
print 'q' + 'w'
```
- виведе 'qw'

А ще (несподівано) **рядок можна множити на ціле число X**. Результатом цієї операції є початковий рядок, повторений X разів.

```
print 'q' * 3
```
- виведе 'qqq'

Інші арифметичні операції для рядків невизначені.

Математичні функції

Крім арифметичних операцій будь-яка мова програмування містить також деякий набір математичних функцій. Аналогічно до функцій перетворення типів даних, вони приймають числові аргументи та повертають розраховане значення -- результат.

Вбудовані функції

Деякі з них вбудовані в мову і доступні в будь-якому місці програми. Це

abs(x) -- модуль від числа

bin(x) -- переведення числа у двійкову систему числення

hex(x) -- переведення числа у шістнадцяткову систему числення

max(x,y) -- пошук максимуму з 2 чисел, також може приймати будь-яку кількість аргументів

min(x,y) -- пошук мінімуму з 2 чисел, також може приймати будь-яку кількість аргументів

round(x) -- округлення числа

round(x,y) -- округлення числа x із вказаною точністю -- у знаків після коми

Модуль math

Більшість математичних функцій недоступні в програмі з самого початку і для використання вимагають підключення стандартного модуля math -- фактично бібліотеки для інтерпретатора, в якій описано, яким чином вони мають обчислюватися (*як саме це працює, ми розглянемо докладніше на наступних лекціях*). Після підключення math на початку вашої програми ви можете вільно використовувати їх в обчисленнях, викликаючи як math.<ім'я функції>(<аргументи функції>):

```
import math
x = 16
print math.sqrt(x)
```
- виведе 4.0

Модуль math містить:

- 2 константи math.pi та math.e
- функції округлення чисел: math.ceil(x) (округлення "вверх"), math.floor(x) (округлення "вниз")
- степеневі та логарифмічні функції: math.pow(x, y), math.exp(x), math.log(x), math.log(x, y), math.log10(x), math.sqrt(x)
- тригонометричні функції: math.sin(x), math.cos(x), math.tan(x), math.asin(x), math.acos(x) та інші (включаючи гіперболічні)
- функції для переведення градусів та радіан: math.degrees(x) та math.radians(x)
- та деякі інші

Складніший приклад:

$W = \pi t 2^{\frac{3}{2}(n-1)}$

Вводимо змінні n та t, наприклад:

```
n = 10
t = 0.99
```

Розраховуємо значення W, для чого нам знадобиться константа pi з модуля math та піднесення в степінь. Можна використати як звичайну операцію **, так і функцію pow з модуля math. Для того, щоб обчислення показника степеня відбувалося вірно (а не діленням націло), записуємо 3.0 замість 3, уточнюючи таким чином, що інтерпретатор має справу з дійсними числами:

```
W = math.pi * t * math.pow(2, (3.0 * (n - 1) / 2))
print W
```

Завдання для тренування

Вхідні дані: 2 невід'ємних дійсних числа a та b -- аргументи командного рядка. b не дорівнює 0.
Вихідні дані: дійсне число -- результат обчислення формули

$$x = \frac{\sqrt{ab}}{e^a * b} + ae^{\frac{2a}{b}}$$

Приклад
Вхідні дані: 0 1
Приклад виклику: python test.py 0 1
Результат: 0.0
Вхідні дані: 0.5 10
Приклад виклику: python test.py 0.5 10
Результат: 0.688209837593

import math
import sys
a=float(sys.argv[1])
b=float(sys.argv[2])
x=math.sqrt(a*b)/(math.pow(math.e, a)*b)+a*math.pow(math.e,2*a/b)
print(x)

або

```
import math
a=float(input())
b=float(input())
#x=a+b
x=math.sqrt(a*b)/(math.pow(math.e, a)*b)+a*math.pow(math.e,2*a/b);
print(x);
```

Підключення модулів

Поки опускаємо подробиці, але ви вже бачили 2 модуля, які можна підключити до програми: sys та math. Це деякі готові бібліотеки, які надають вам доступ до якогось додаткового

функціоналу і розширюють ваші можливості при написання програми. Їх підключення виглядало як:

```
import sys
import math
```

Будь-яка програма починається з підключення модулів. Можливо не цих, а якихось інших (не в будь-якій програмі потрібні математичні функції). Можливо вам не потрібне розширення функціоналу і ця секція програми просто залишиться порожньою. Але, якщо ви щось підключаєте, намагайтеся робити це на самому початку.

Коментарі

Коментар - це будь-який текст для пояснення роботи програми, який не виконується інтерпретатором. Коментарі можуть бути короткі: відокремлюються від основного коду дієзом і все, що знаходиться в коді програми після дієза і до кінця рядка, інтерпретатором ігнорується. Або довгі, в декілька рядків: перед початком багаторядкового коментаря ставиться троє подвійних лапок, і ще троє в кінці -- все, що знаходиться між ними, інтерпретатор також не виконає.

```
number = 10 # number of variables
"""
Here the program begins. This piece of text will be ignored while executing the
program
"""
number = number + 1
```

Крім тлумачення змінних коментарі можуть застосовуватися будь-де в коді програми для пояснення, що там відбувається. Часто буває, що, повертаючись через деякий час (тижня чи двох достатньо), автор вже не пам’ятає, для чого виконувалася та чи інша дія. А, якщо код програми мають читати сторонні люди (наприклад, програма розробляється кількома програмістами), коментарі тим більше не будуть зайвими.

Алгоритмічні конструкції

Ключовим у прикладі є застосування алгоритмічних конструкцій - умовного розгалуження та циклу. Обидві вони передбачають нелінійне виконання програми, для чого можуть мати вкладені блоки коду - саме ті фрагменти програми, які необхідно виключити з лінійної послідовності виконання дій в програмі.

Нагадую: вкладені блоки коду в python виділяються відступами на початку рядка. Алгоритмічні структури можуть вкладатися одна в одну скільки завгодно разів -- це призводить до більшої вкладеності блоків коду і, відповідно, більших відступів.

```
if n == 0:
  print "n is equal to zero"
else:
  if n == 7:
    print "You're lucky!"
  else:
    if n > 0:
      print 'n is greater than zero'
    else:
      print "n is less than zero"
```

З іншого боку це значить, що та частина коду, яка не передбачає вкладеності, тобто виконується послідовно, не має містити зайвих відступів, плаваючи вліво-вправо - всі команди мають бути гарно написані на одному рівні.

Більшість мов програмування є досить гнучкими в цьому питанні, але містять додаткові оператори або позначки для виділення вкладених блоків. Розробники ж python пішли іншим шляхом: позбавились таким чином зайвих елементів у коді і одночасно змусили програмістів гарно форматувати код.

Умовне розгалуження -- if..else

Логічно, описує умову, за якої відбувається перехід на одну гілку чи іншу, та включає 2 вкладених блоки коду:

```
if n == 0:
  print ("n is equal to zero")
else:
  print ("n is not equal to zero")
```

Якщо умова справджується, виконується перший блок. Якщо ні -- другий. Іноді буває, що необхідно щось робити лише у випадку, коли умова вірна -- в такому випадку другу частину конструкції можна опустити:

```
if n == 0:
  print ("n is equal to zero")
print ("this will be printed anyway")
```

В якості умови найчастіше застосовуються порівняння змінних із значеннями або іншими змінними. Для цього вам доступні <, >, <=, >=, ==. Для перевірки “не дорівнює” є аж два позначення: != та <>. Зверніть увагу: для перевірки, чи є значення або змінні рівними використовується подвійне “дорівнює” ==, одинарне ж застосовується для присвоєння значень.

Форма if .. elif .., яка дозволяє побудувати розгалуження більш ніж на 2 гілки, не створюючи для цього додаткових рівнів вкладеності в коді програмі.

```
n = int(raw_input('Input n: '))
if n > 100:
    print ('so large number!')
elif n > 10:
    print ('ok, not bad')
elif n == 0:
    print ('tricky!')
else:
    print ('hey, what are you doing?')
```

Цикл for

Це цикл із наперед визначеною кількістю ітерацій. Не заглиблюючись у подробиці, ви можете підставити будь-яке N та повторити необхідну дію або послідовність дій (блок коду) N разів. При цьому змінна-лічильник циклу набуватиме на кожній ітерації значення від 0 до N-1. Наприклад, ми можемо вивести перші N натуральних чисел:

```
n = int(raw_input('input N:'))
for counter in range(n):
  print (counter+1)
```

Або скласти між собою 10 членів арифметичної прогресії a1+a2+...+a10 з a0=1 та d=5:

```
a_0 = 1                    # zero member
a_previous = a_0           # previous member
a_i = 0                    # current member
d = 5
sum = a_0
for counter in range(10):
  a_i = a_previous + d     # calculate the new member
  sum = sum + a_i
  a_previous = a_i         # save it as previous
print (sum)
```

Поки що він виглядав приблизно так, і це, умовно кажучи, був цикл для того, щоб повторити дію n разів:

```
n = 10
for i in range(n):
    print (i)
```

Але, після розгляду списків та функції range(), стає зрозуміло, що призначення циклу for трохи більш загальне - це обхід послідовностей, наприклад списків. Або рядків. Так, рядки являють собою впорядковані послідовності літер і працюють схожим чином.

Отже нам не обов'язково обмежуватись числами від 0 до N. Ми можемо підготувати заздалегідь будь-який список або рядок і перебрати поелементно. При цьому на кожній ітерації змінна-лічильник циклу набуватиме значення наступного елемента послідовності.

```
# letter послідовно набуває значення кожного символу рядка
for letter in 'qwertyuiopasdfghjkl':
    print letter
list_to_iterate = [1, 2, 3, 4, 8, 2, 43, 12, 31, 20]
# element послідовно набуває значення кожного елементу списку
for element in list_to_iterate:
    print element
# element послідовно набуває значення кожного другого елементу списку
for element in list_to_iterate[::2]:
    print element
```

Якщо вам необхідно саме перебрати значення послідовності, або ви просто знаєте наперед кількість необхідних ітерацій циклу, цикл for підійде якнайкраще. Крім того, що він призначений саме для цього, його важче впустити у нескінченне повторення, так як будь-яка послідовність має бути скінченною.

Цикл while

Цикл while дозволяє так само перебирати послідовності:

```
n = 10
i = 0
while i < 10:
    print (i)
    i = i + 1
```

або

```
n = 10
list_to_iterate = [1, 2, 3, 4, 8, 2, 43, 12, 31, 20]
i = 0
while i < len(list_to_iterate):
    print (list_to_iterate[i])
    i = i + 1
```

Але він є більш універсальним і може працювати із будь-якою умовою. Коли ви очікуєте якоїсь події, коли вам потрібно повторювати розрахунки до досягнення заданої точності -- в усіх випадках, коли ви не знаєте наперед кількість ітерацій:

```
sum = 0
i = 0
while sum < 100:
    i = i + 1
    sum = sum + i
```

Цикл while продовжує повторювати блок коду, поки значення умови рівне True. І ви завжди повинні бути впевнені, що колись умова змінить своє значення і цикл скінчиться. Бо пропустити збільшення лічильника або просто помилитися в умові набагато простіше ніж “випадково” розтягнути скінченну послідовність до нескінченної.

Список

Список – складний тип даних, впорядкована послідовність значень будь-яких типів.

Це значить, що змінна-список містить не одне значення, як число або логічна змінна, а одразу декілька. Причому це можуть бути будь-які значення (рядки, числа, логічні змінні, інші списки) і їх порядок чітко визначений, отже у кожного значення в списку є порядковий номер – індекс, за яким можна звернутися до окремого елемента списку, щоб прочитати або записати значення. Для визначення списку його елементи беруться в квадратні дужки.

```
example_list = ['one', 1, 2.0, True, [0.1, 0.2, 0.3]]
example_list[0] == 'one'
example_list[1] == 1
example_list[4][1] == 0.2
example_list[5] = '5!' # присвоїли значення новому елементу
```

Найцікавішим є те, що до списків можна застосовувати різні вбудовані функції, призначені для роботи з ними. Наприклад, визначити довжину списку:

```
print (len(example_list))
```

Зрізи

З індексами все досить просто і зрозуміло: якщо значення складається з інших значень, то необхідний якийсь доступ до його складових.

Більш цікавою можливістю списків є взяття “зрізу”: крім того, щоб звертатися до окремих елементів, можна вибирати цілі фрагменти послідовності, утворюючи за необхідності нові списки.

- example_list[i:j] -- вибере всі елементи списку з i-го (включно) по j-й (виключаючи),

- example_list[i:] -- вибере всі елементи списку з i-го (включно) до кінця,
- example_list[:j] -- вибере всі елементи списку з початку по j-й (виключаючи).

Якщо i чи j від'ємні, відлік для них буде проводитися з кінця послідовності (до речі, те саме стосується від'ємних індексів). Перевірте, що виведуть print example_list[-1], example_list[1:4], example_list[-1:13], example_list[4:1], example_list[:-1], example_list[-2:].

Також операцію зрізу можна застосовувати і з 3 аргументами:

- example_list[i:j:k] -- вибере кожний k-й елемент списку з i-го (включно) по j-й (виключаючи),
- example_list[i::k] -- вибере кожний k-й елемент списку з i-го (включно) до кінця,
- example_list[:j:k] -- вибере кожний k-й елемент списку з початку по j-й (виключно),
- example_list[::k] -- вибере кожний k-й елемент списку.

k також може бути від'ємним, що призведе до формування нового списку із зворотним порядком елементів. Перевірте самі: example_list[::2], example_list[:4:3], example_list[1::-1], example_list[3:0:-1].

Є ще одна функція, яка працює схожим чином, - це вже частково знайома вам range(), яка по суті створює список-зріз послідовності цілих чисел:

- range(i) -- повертає список з числами від 0 (включно) до i (виключаючи, тобто до i-1),
- range(i,j) -- повертає список з числами від i (включно) до j (виключаючи, тобто до j-1),
- range(i,j,k) -- повертає список, що містить кожне k-те число від i (включно) до j (виключаючи, тобто до j-1).

Аналогічно, i, j, k можуть бути від'ємними числами.

Корисні функції

len(x) - повертає довжину послідовності x (списку або рядка).

```
print (len('qweqweqweqw qwe qr qr '))
print (len([123.1, 32, 66, 23, 'q', 55, 2342])
```

range(x) - генерує список цілих значень від 0 до x-1.

range(x, y) - генерує список цілих значень від x до y-1.

range(x, y, z) - генерує список цілих значень від x до y-1 з кроком z.

```
print (range(1, 5, 2)  # [1, 3])
print (range(10, -5, -3) # [10, 7, 4, 1, -2])
print (range(10)  # [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
print (range(66, 68)  # [66, 67])
```

x.**append**(y) - додає значення y в кінець списку x

```
x = []
x.append('first!')
print (x) # ['first!']
```

x.**reverse**() - змінює порядок елементів списку x на зворотний

```
x = [1, 2, 3]
x.reverse()
print (x) # [3, 2, 1]
```

x.**upper**() - змінює регістр всіх символів рядка x на верхній.

```
print ('String'.upper()) # 'STRING'
```

x.**lower**() - змінює регістр всіх символів рядка x на нижній.

```
print ('String'.lower()) # 'string'
```

x.**replace**(substring_old, substring_new) - замінює всі входження фрагменту substring_old в рядку x на substring_new

x.replace(substring_old, substring_new, count) - замінює перші count входжень фрагменту substring_old в рядку x на substring_new

```
message = "Good morning, man. What are you doing?"
print (message.replace("ng", "n'"))
# "Good mornin', man. What are you doin'?"
```

x.**find**(substring) - повертає позицію входження (індекс першого символу) фрагменту substring в рядку x або -1, якщо фрагмент не знайдено.

x.find(substring, start_pos) - повертає позицію входження (індекс першого символу) фрагменту substring в рядку x починаючи з позиції start_pos, або -1, якщо фрагмент не знайдено.

```
str1 = "this is string example....wow!!!"
print (str1.find("exam"))
print (str1.find("exam", 10))
```

Рядки можуть містити спеціальні символи, які при виведенні рядка обробляються спеціальним чином - вони називаються escape-послідовностями. Найбільш часто використовуються наступні:

- \n - перехід на новий рядок
- \t - вставка табуляції
- \" - подвійні лапки (корисно, якщо необхідно вставити подвійні лапки в рядок, оточений подвійними лапками)
- \' - одинарні лапки (корисно, якщо необхідно вставити одинарні лапки в рядок, оточений одинарними лапками)

Завдання

1. **Складіть програму для завдання, що знаходиться під номером, що відповідає Вашому порядковому номеру у списку групи.**

Дано *x, y, z.* Вичислити *a, b,* якщо

1. $a = \dfrac{\sqrt{|x-1|} - \sqrt[3]{|y|}}{1 + \dfrac{x^2}{2} + \dfrac{y^2}{4}}$,

2. $b = x(\operatorname{arctg}(z) + e^{-(x+3)})$;

3. $a = \dfrac{3 + e^{y-1}}{1 + x^2 |y - \operatorname{tg} z|}$,

4. $b = 1 + |y - x| + \dfrac{(y-x)^2}{2} + \dfrac{|y-x|^3}{3}$;

5. $a = (1+y)\dfrac{x + y/(x^2+4)}{e^{-x-2} + 1/(x^2+4)}$,

6. $b = \dfrac{1 + \cos(y-2)}{x^4/2 + \sin^2 z}$;

7. $a = y + \dfrac{x}{y^2 + \left|\dfrac{x^2}{y + x^3/3}\right|}$,

8. $b = (1 + \operatorname{tg}^2 \dfrac{z}{2})$;

9. $a = \dfrac{2\cos(x - \pi/6)}{1/2 + \sin^2 y}$,

10. $b = 1 + \dfrac{z^2}{3 + z^2/5}$;

11. $a = \dfrac{1 + \sin^2(x+y)}{2 + |x - 2x/(1 + x^2 y^2)|} + x$,

12. $b = \cos^2(\operatorname{arctg} \dfrac{1}{z})$;

13. $a = \ln\left|(y - \sqrt{|x|})(x - \dfrac{y}{z + x^2/4}\right|$,

14. $b = x - \dfrac{x^2}{3!} + \dfrac{x^5}{5!}$.

15. Визначити відстань між двома точками з координатами (x_1,y_1) та (x_2,y_2).

16. Трикутник заданий координатами своїх вершин. Знайти периметр трикутника.

17. Трикутник заданий координатами своїх вершин. Знайти площу трикутника.

18. Знайти площу сектору, Радіус якого 13.7, а дуга містить задане число радіан φ .

Дано дійсне число *a.* Не використовуючи жодні арифметичні операції крім множення отримати:

19. a^7 за чотири операції;

20. a^{13} за п’ять операцій;

21. a^{15} за п’ять операцій;

22. a^{21} за шість операцій;

23. a^{28} за шість операцій.

24. Дані два дійсних числа *a i b.* Отримати їх суму, різницю та добуток.

25. Дані дійсні числа *x* і *y.* Отримати $\frac{|x|-|y|}{1+|xy|}$.

26. Дано довжину ребра куба. Знайти об'єм та площу бокової поверхні.

27. Дані два дійсних додатних числа. Знайти середнє арифметичне та середнє геометричне цих чисел.

28. Дані два дійсних числа. Знайти середнє арифметичне та середнє геометричне їх модулів.

29. Дані катети прямокутного трикутника. Знайти його гіпотенузу та катети.

30. Визначити периметр правильного *n*-кутника, описаного навколо кола радіусом *r*.

2. Складіть програму для завдання, що знаходиться під номером, що відповідає Вашому порядковому номеру у списку групи.

1. Дані дійсні числа *x*, *y*. Отримати max (*x*, *y*).

2. Дані дійсні числа *x*, *y*. Отримати min (*x*, *y*).

3. Дані дійсні числа *x*, *y*, *z*. Отримати max (*x*, *y*, *z*).

4. Дані дійсні числа *x*, *y*, *z*. Отримати min (*x*, *y*, *z*).

5. Дані дійсні числа *x*, *y*, *z*. Обчислити max (*x* + *y* + *z*, *xyz*).

6. Дані дійсні числа *x*, *y*, *z*. Обчислити $\min^2$ (*x* + *y* + *z/2, xyz*) +1.

7. Дані дійсні числа *a*, *b*, *c*. Перевірити чи справедлива нерівність $a < b < c$.

8. Дані дійсні числа *a*, *b*, *c*. Подвоїти ці числа, якщо $a \geq b \geq c$, і замінити їх абсолютним значенням, якщо це не так.

9. Дані дійсні числа *x*, *y*. Обчислити *z*:

$$z = \begin{cases} x - y, & \text{якщо x>y,} \\ y - x + 1 & \text{в іншому випадку.} \end{cases}$$

10. Дані два дійсних числа. Вивести перше число, якщо воно більше другого, і обидва числа, якщо це не так.

11. Дані два дійсних числа. Замінити перше число нулем, якщо воно менше чи рівне іншому, і залишити без змін в іншому випадку.

12. Дано три дійсних числа. Вибрати з них ті, котрі належать інтервалу (1, 3).

13. Дані дійсні числа *x, y (x≠ y).* Менше з цих чисел замінити їх пів сумою, а більше – їх подвоєним добутком.

14. Дано три дійсних числа. Піднести до квадрату ті з них, котрі не є від'ємними

Дано дійсне число *a.* Обчислити(*a*), якщо

15. $$f(x)=\begin{cases} x^2 & при\ -2\le x<2, \\ 4 & у\ інших\ випадках \end{cases}$$

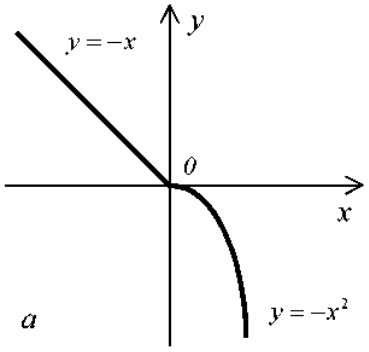
16. $$f(x)=\begin{cases} 0 & \text{при } x\le 0, \\ x & \text{при } 0<x\le 1, \\ x^4 & \text{в інших випадках} \end{cases}$$

17. $$f(x)=\begin{cases} 0 & \text{при } x\le 0, \\ x^2-x & \text{при } 0<x\le 1, \\ x^2-\sin \pi x^2 & \text{в інших випадках} \end{cases}$$

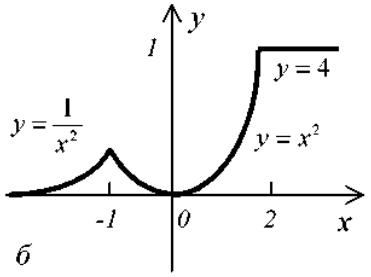
18. $$f(x)=\begin{cases} x^2+4x+5 & \text{при } x\le 2, \\ \dfrac{1}{x^2+4x+5} & \text{в іншому випадку} \end{cases}$$

Дано дійсне число *a*. Для функцій *f*(*x*), графіки котрих представлені на рис. вичислити *f*(*a*).

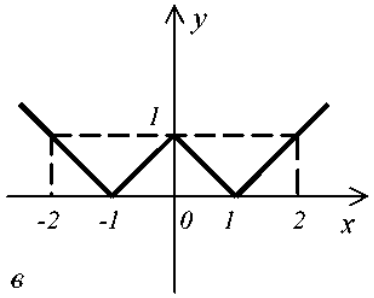
19.



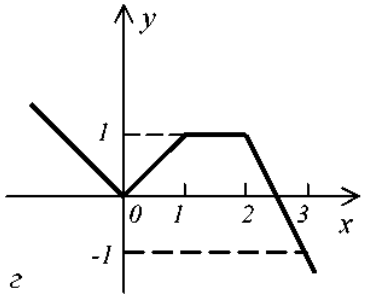
20.



21.

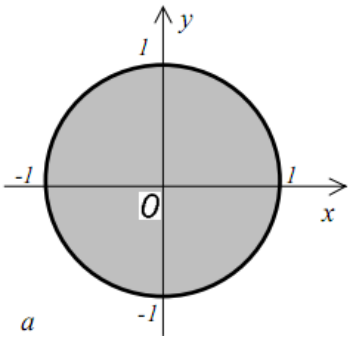


22.

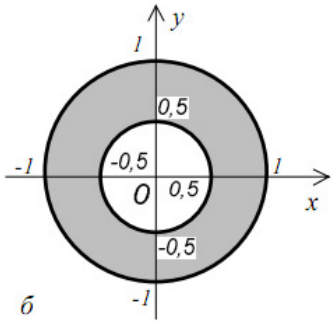


Дані дійсні числа *x*, *y*. Визначити чи належить точка з координатами *x*, *y* заштрихованій частині площини.

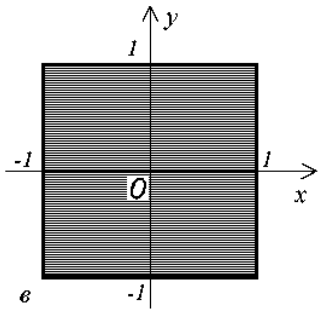
23.



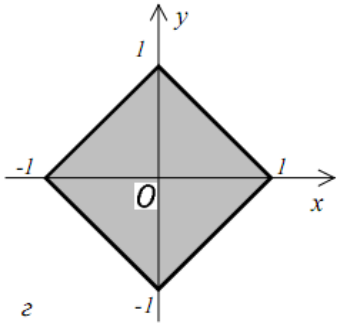
24.



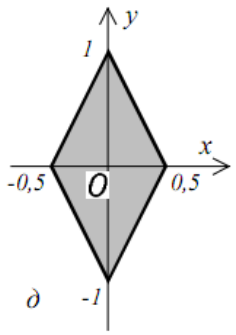
25.



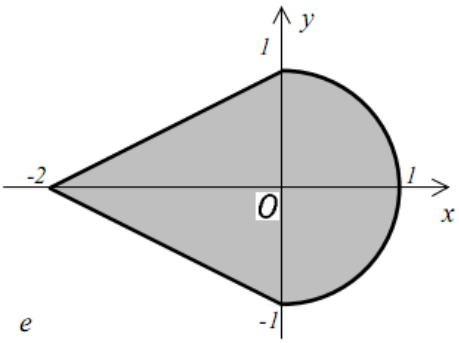
26.



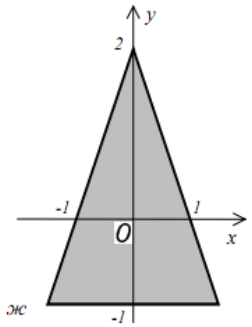
27.



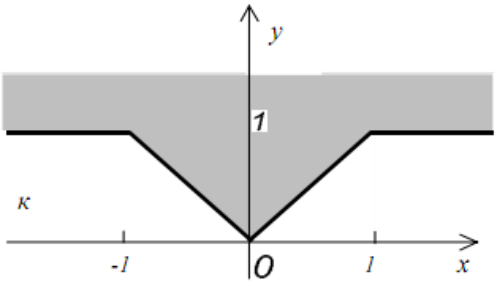
28.



29.



30.



3. Складіть програму для завдання, що знаходиться під номером, що відповідає Вашому порядковому номеру у списку групи та для завдання (N+10), де N – Ваш порядковий номер в списку групи. Для одного із завдань використайте цикл for, а для іншого while.

1. Дано натуральне число n. Вичислити 2^n.

2. Дано натуральне число n. Вичислити $n!$.

3. Дано натуральне число n. Вичислити $\left(1+\frac{1}{1^2}\right)\left(1+\frac{1}{2^2}\right)\ldots\left(1+\frac{1}{n^2}\right)$.

4. Дано натуральне число n. Вичислити $\frac{1}{\sin 1}+\frac{1}{\sin 1+\sin 2}+\ldots+\frac{1}{\sin 1+\ldots+\sin n}$.

5. Дано натуральне число n. Вичислити $\underbrace{\sqrt{2+\sqrt{2+\ldots+\sqrt{2}}}}_{\text{n коренів}}$.

6. Дано натуральне число n. Вичислити $\frac{\cos 1}{\sin 1}\cdot\frac{\cos 1+\cos 2}{\sin 1+\sin 2}\cdot\ldots\cdot\frac{\cos 1+\ldots+\cos n}{\sin 1+\ldots+\sin n}$.

7. Дано натуральне число n. Вичислити $\sqrt{3+\sqrt{6+\ldots+\sqrt{3(n-1)+\sqrt{3n}}}}$.

8. Дано дійсне число a, натуральне число n. Вичислити a^n.

9. Дано дійсне число a, натуральне число n.

Вичислити $a(a+1)\ldots(a+n-1)$.

10. Дано дійсне число a, натуральне число n.

Вичислити $\frac{1}{a}+\frac{1}{a(a+1)}+\ldots+\frac{1}{a(a+1)\ldots(a+n)}$.

11. Дано дійсне число a, натуральне число n.

Вичислити $\frac{1}{a}+\frac{1}{a^2}+\frac{1}{a^4}+\ldots+\frac{1}{a^{2^n}}$.

12. Дано дійсне число a, натуральне число n.

Вичислити $a(a-n)(a-2n)\ldots(a-n^2)$

13. Дано дійсне число *x.* Вичислити

$$x-\frac{x^3}{3!}+\frac{x^5}{5!}-\frac{x^7}{7!}+\frac{x^9}{9!}-\frac{x^{11}}{11!}+\frac{x^{13}}{13!}.$$

14. Дані дійсні числа *x*, *a*, натуральное число *n*. Вичислити

$$\underbrace{((...((}_{\text{n скобок}}x+a)^2+a)^2+...+a)^2+a)^2+a.$$

15. Дано дійсне число *x*. Вичислити $\frac{(x-2)(x-4)(x-8)...(x-64)}{(x-1)(x-3)(x-7)...(x-63)}$.

16. Дано дійсне число *a*. Знайти серед чисел $1,\ 1+\frac{1}{2},\ 1+\frac{1}{2}+\frac{1}{3},\ \ldots$ перше, котре більше за *a*.

17. Дано дійсне число *a*. Знайти таке найменше *n*, що

$$1+\frac{1}{2}+...+\frac{1}{n}>a.$$

18. Дано натуральне число *n,* дійсне число *x*. Вичислити

$\sin x+\sin^2 x+...+\sin^n x$.

19. Дано натуральне число *n,* дійсне число *x*. Вичислити

$\sin x+\sin x^2+...+\sin x^n$.

20. Дано натуральне число *n,* дійсне число *x*.

Вичислити в) $\sin x+\sin\sin x+...+\underbrace{\sin\sin...\sin x}_{n}$.

21. Дано натуральне число *n.* Скільки чифр в числі *n?*

22. Дано натуральне число *n.* Чому рівна сума його цифер?

23. Дано натуральне число *n.* Знайти першу цифру числа *n.*

24. Дано натуральне число *n*, дійсне число *x*. Вичислити $x^{n^2}/2^n$

25. Дано натуральне число *n*, дійсне число *x*. Вичислити $x^{n^3}/3^n$.

26. Дано натуральне число n. Вичислити $\sum_{k=1}^{n} \frac{1}{k}$.

27. Дано натуральне число n. Вичислити $\sum_{k=1}^{n} \frac{1}{k^5}$.

28. Дано натуральне число *n*, дійсне число *x*. Вичислити $\sum_{i=1}^{n} \frac{x^i}{i!}$.

29. Дано натуральне число *n*, дійсне число *x*. Вичислити $\sum_{i=1}^{n} \left(\frac{1}{i!} + \sqrt{|x|} \right)$.

30. Дано натуральне число *n*, дійсне число *x*. Вичислити

$$\sum_{i=1}^{n} \frac{x + \cos(ix)}{2^i}.$$

4. Складіть програми для відповідних завдань:

Варіанти:

№ в списку групи	Завдання 1	Завдання 2	Завдання 3
1.	1 a	9 a	18 f
2.	1 b	9 b	18 g
3.	2 a	9 c	18 h
4.	2 b	10 a	19
5.	2 c	10 b	6 a
6.	2 d	11 a	6 b
7.	3	11 b	6 c
8.	4	12	6 d
9.	5 a	13	6 e
10.	5b	14	6 f
11.	6a	15	18 d

12.	6b	16	18 f
13.	6c	17	18 h
14.	6d	18 a	2 a
15.	6e	18 b	2 b
16.	6f	18 c	2 c
17.	7	18 d	2 d
18.	8	18 e	3
19.	1 a	9 a	18 f
20.	1 b	9 b	18 g
21.	2 a	9 c	18 h
22.	2 b	10 a	19
23.	2 c	10 b	6 a
24.	2 d	11 a	6 b
25.	6c	10 b	18 h
26.	6d	18 a	2 d

Завдання:

1. Дана стрічка. Підрахувати:
    a) Скільки разів серед символів зустрічається символ + і скільки разів символ *;
    b) Загальне число входжень символів +, -, * в стрічку.
2. Дана стрічка. Перетворити її, замінивши:
    a) Всі знаки оклику крапками;
    b) Кожну крапку трьома крапками;
    c) Кожну з груп крапок однією крапкою;
    d) Кожну з груп крапок трьома крапками.
3. Дана стрічка. Встановити, чи зустрічається в ній послідовність символів кома і тире (,-).
4. Дана стрічка. Вивести порядковий номер першого входження у стрічку двох символів а (аа). Якщо такої пари символів немає, то вивести 0.
5. Дана стрічка. Відомо, що серед символів є принаймні одна кома. Знайти порядковий номер символу:
    a) Першої коми;
    b) Останньої коми.
6. Дано стрічку. Відомо, що перший символ відмінний від знаку оклику і що серед інших

символів є принаймні один знак оклику. Нехай $s_1, \ldots, s_n$ - символи даної послідовності, що знаходяться перед першим знаком оклику (n – наперед не відоме).

a) Визначити кількість пробілів серед $s_1, \ldots, s_n$;

b) Вияснити, чи входить в послідовність $s_1, \ldots, s_n$ буква «k»;

c) Вияснити,чи серед $s_1, \ldots, s_n$ є всі букви, що входять слово «city»;

d) Вияснити, чи серед $s_1, \ldots, s_n$ є пара сусідніх букв «di» або «id»;

e) Вияснити, чи серед $s_1, \ldots, s_n$ є пара сусідніх однакових символів;

f) Вияснити, чи існують такі натуральні i та j, що $1 < i < j < n$ і що s_i співпадає з s_{i+1}, а s_j з s_{j+1}.

7. Дано стрічку. Видалити із неї всі групи букв $abcd$.
8. Дано стрічку. Перетворити її наступним чином: видалити всі символи * і подвоїти кожен символ відмінний від *.
9. Дано стрічку, серед символів якої є двокрапка.
   a) Отримати всі символи, які знаходяться до першої двокрапки включно;
   b) Отримати всі символи, які знаходяться після першої двокрапки;
   c) Отримати всі символи, які знаходяться між першою та другою двокрапками. Якщо другої двокрапки немає, то вивести всі символи, які знаходяться після єдиної двокрапки.
10. Дано стрічку. Вивести:
    a) Підрахувати найбільшу кількість пробілів, що йдуть підряд;
    b) Встановити, чи в стрічці є п’ять підряд розміщених символів «e».
11. Дано стрічку. Визначити число входжень в стрічку групи букв:
    a) abc;
    b) aba.
12. Дано стрічку. Замінити в ній кожну групу букв “child” на “children”.
13. Дано стрічку. Виключити з неї групи символів, що розміщені між дужками та самі дужки теж. Вважати, що в середині дужок інших дужок немає.
14. Дано стрічку. Перетворити її наступним чином: якщо немає символу *, то залишити без змін, інакше замінити кожен символ, що зустрічається після символу * на символ -.
15. Дана стрічка серед символів якої є хоча б одна крапка. Перетворити її наступним чином: видалити з неї всі коми, що знаходяться до першої крапки і замінити знаком + всі цифри 3, що зустрічаються після першої крапки.
16. Дано стрічку $s_1, \ldots, s_n$. Замінити в ній комами всі двокрапки, що зустрічаються серед

$s_1, \ldots, s_{[n/2]}$ і замінити крапками всі знаки оклику, що зустрічаються серед $s_{[n/2]+1}, \ldots, s_n$.

17. Дано стрічку. Видалити з неї всі групи пробілів,якими починається та закінчується стрічка, а також кожну внутрішню групу пробілів замінити на один пробіл.
18. Дано стрічку. Групи символів, що розділені пробілом (або декількома пробілами) будемо називати словами:
    a) Підрахувати кількість слів в даній послідовності;
    b) Підрахувати кількість букв «а» в останньому слові послідовності;
    c) Знайти кількість слів, що починаються з букви «с»;
    d) Знайти кількість слів, у яких перший і останній символи співпадають;
    e) Знайти будь-яке слово, що починається з букви «а»;
    f) Замінити кожне слово «this» на «then»;
    g) Змінити порядок слів у реченні (наприклад: «я вивчаю Python» потрібно змінити на «Python вивчаю я»);
    h) Знайти довжину найкоротшого слова послідовності.
19. Дана стрічка. Вивести “YES”, якщо стрічка однаково читається зліва направо та справа на ліво, вивести “NO” в протилежному випадку.

5. Оформіть звіт про виконання лабораторної роботи.