«ПРОГРАММИРОВАНИЕ НА PYTHON ДЛЯ ТЕХНИЧЕСКИХ ЗАДАЧ»

План практических занятий

Программированию обычно учат на примерах.
Niklaus Wirth

1. Вводное занятие

1.

Заполните анкету по ссылке https://forms.gle/vSMfJso5DANczuTb6 .

2.

По ссылке python.org/downloads/ скачайте и установите интерепритатор Python для своей операционной системы. **Важно:** убедитесь, что при установке установлен флажок **Add Python 3 to the PATH.**

3.

Запустите PowerShell или ее аналог в вашей операционной системе.

4.

В оболочке PowerShell запустите среду Python, набрав команду *python* в командной строке. Если команда не распознается оболочкой командной строки, вернитесь к шагу 2. Вероятно, вы не установили флажок **Add Python 3 to the PATH.** Если все работает, введите команду quit()

5.

Теперь давайте познакомимся с некоторыми консольными командами. Для начала перейдем в каталог, в котором будем работать:

cd ls – это пример, тут должен быть путь к папке в вашей системе.

Создаем новый каталог

mkdir 'Lab_1'

И переходим в него

cd Lab_1

ls выводит содержимое текущего каталога. Если потерялись: команда pwd. cd ~ Возвращает в домашний каталог. Что еще может пригодиться:

pwd	Вывести текущую директорию
mkdir	Создать каталог
cd	Сменить каталог
ls	Вывести содержимое каталога
rmdir	Удалить каталог
cp	Скопировать файл или каталог
type	Выести все содержимое файла
help	справка
exit	Покинуть консоль

Самостоятельно: «походить» по каталогам своего компьютера, просмотреть содержимое. Затем вернуться в созданный рабочий каталог.

6.

Теперь познакомимся с простыми командами Python 3 в консольном режиме, а также базовыми типами данных.

Зачем это вообще? Этот способ интерпретации команд позволяет попробовать нужные нам функции языка, не прибегая к написанию отдельных скриптов для этого.

Запускаем интерпретатор, вводим следующие команды и наблюдаем за результатом.

```
print("Привет")
print("Еще привет") - #просто комментарий
```

Рекомендации по именам переменных:

имя переменной может состоять только из цифр, букв и знаков подчеркивания;

имя переменной не может начинаться с цифры.

-Имя должно описывать суть

-Будьте последовательны

-Уважайте обычай языка

-Следите за длиной

6.1 Числа

```
# int float
>>> num = 13
>>> print(type(num))
<class 'int'>
>>>
>>> num = 13.4
>>> print(num)
13.4
>>>
>>> num = 1.5e2
>>> print(num)
150.0
>>>
>>> num = 150.2
>>> print(type(num))
<class 'float'>
>>>
>>> num = int(num)
>>> print(num, type(num))
150 <class 'int'>
>>>
>>> num = float(num)
>>> print(num, type(num))
150.0 <class 'float'>
>>> int('100')
100
>>> int('100', base=2)
4
>>> int(0b100)
```

```
4
>>>


# Комплексные числа
>>> num = 14 + 1j
>>> print(type(num))
<class 'complex'>
>>> print(num.real)
14.0
>>> print(num.imag)
1.0
>>>

# Арифметические операции
x=5
y=2

x+y
x-y
x*y
x**y или pow(x, y)
x/y
x%y # остаток
x // y #неполное частное
abs(x)
>>> c_num = complex(x,y) # комплексное число
(5+2j)
>>> c_num.conjugate()  # комплексно-сопряженное
(5-2j)

>>> divmod(x, y) # пара (x // y, x % y)
```

Для преобразования целых чисел в другие системы счисления есть встроенные функции

```
>>> num = 8
>>> bin(num)
'0b1000'
>>> hex(num)
'0x8'
>>> oct(num)
'0o10'

# Методы типа int

>>> num.bit_length()
4
>>> num.bit_count()
1
```

И еще много других, которые при необходимости смотрим в документации.

6.2 Логический тип данных и булевы операции

В логическом (булевом) типе данных имеется 2 значения: True («правда») и False («ложь»). Считаются подтипом integer.

Наберите в консоли:

```
print(3 > 4)
print(3 <= 3)
print(6 >= 6)
print(6 < 5)
print(6==5)
print(6 != 5)
```

Логические операторы: `not, or, and.`
Свойства:
1) Коммутативность
A and B== B and A
A or B== B or A
2) Ассоциативность
A and (B and C) == (A and B) and C
A or (B or C) == (A or B) or C
3) Дистрибутивность
A and (B or C) == (A and B) or (A and C)
A or (B and C) == (A or B) and (A or C)
4) Закон де Моргана
not(A and B) == not(A) or not(B)
not(A or B) == not(A) and not(B)

Построить таблицы истинности для логических операторов

Если применить функцию bool() к большинству объектов в Python, то получим True, исключая следующие случаи:
1) bool(False)
2) bool(None)
3) bool(0)
4) bool('') или bool([]) или bool({})

Приоритет булевых опреаций: not, and, or .

```
x, y, z = True, False, True
result = not( x and y or z )
print(result)
```

В чем смысл дистрибутивности логических выражений? Дистрибутивность связана с раскрытием скобок в логических выражениях. Она определяется следующими правилами:

```
True and (False or True) == (True and False) or (True and True)
True or (False and True) == (True or False) and (True or True)
```

Самостоятельно: определить, является ли год високосным. Год является високосным, если он кратен 4, но при этом не кратен 100, либо кратен 400. С использованием логических выражений решений реализуется в три строчки.

Решение:

```
year = 2022
```

```
is_leap = year % 4 == 0 and (year % 100 != 0 or year % 400 == 0)
print(is_leap)
```

6.3 Строки

```
example_string = "ПРОГРАММИРОВАНИЕ НА PYTHON ДЛЯ ТЕХНИЧЕСКИХ ЗАДАЧ"
print(example_string)

print(type(example_string))

"Можно" + " просто " + "складывать" + " строки"

# Срез строки
example_string[0:15]
```

Что еще можно делать со строками? Мы видели, что стандартные, встроенные в интерпритатор типы данных по сути являются классами. У классов есть свои методы. Вот, например, некоторые методы строк:

```
>>> ex_str='1111'
>>> ex_str.isdecimal()
True

>>> ex_str = 'testCase'
>>> ex_str.removeprefix('test')
'Case'
>>> print(ex_str)
testCase # Обратите внимание: хотя метод вернул строку без префикса, сама исходная строка переписана не была, т.к. строки – это неизменяемый тип.
>>>

>>> ex_str = 'CaseTest'
>>> ex_str.removesuffix('Test')
'Case'
```

Все методы с примерами доступны в документации по ссылке: https://docs.python.org/3/library/stdtypes.html

Оператор in позволяет проверить наличие подстроки в строке:

```
"3.14" in "Число Пи = 3.1415926"

>>> ex_str = 'Привет'
>>> for letter in ex_str:
...             print('Буква', letter)
...
Буква П
Буква р
Буква и
Буква в
Буква е
Буква т
```

```
# К строкам можно применить конвертацию типов. Например, мы можем преобразовать вещественное число в переменную типа str:

num = 999.01
num_string = str(num)
print(type(num_string))
num_string


######### Форматирование строк ############

Зачем? Чтобы поместить переменную в любую текстовую строку.

day = 'понедельник'

author = 'Christopher Thompson'

print(f'Иногда лучше остаться спать дома в {day}, чем провести всю неделю в отладке написанного в {day} кода. {author}')

term = 'Простота'
print("{} — залог надежности.".format(term))


print("""
Что такое?
Почему трое кавычек?!
Мы можем набрать сколько угодно строк текста
хаха
""")
```

7.

Начнем знакомство с инструментами разработки. По ссылке www.atom.io скачайте и установите дистрибутив редактора кода Atom. Если для вас привычен какой-либо другой редактор, используйте его. Даже если это просто Блокнот.

Для тех, у кого уже есть редактор кода: чтобы не скучать, решите задачу:
Используя input() для ввода с клавиатуры числа ступенек, нарисовать лесенку.

```
    #
   ##
  ###
 ####
#####
```

```
num_steps = int( input('ступенкек? '))
i = 0
while i < num_steps:
        print(" " * (num_steps - i - 1), "#" * (i + 1))
         i += 1
```

8.

Создаем в редакторе кода новый документ. Напишем программу для расчета индекса массы тела, оформив ее как отдельный файл-сценарий (скрипт).

```
print('Сколько тебе лет?', end = ' ')
age = input()

print('Какой у тебя рост?', end = ' ')
height = input()

print('Сколько ты весишь?', end = ' ')
weight = input()

BMI = float(weight) / (float(height)**2)

print(f'Well, you are {age} years, your height is {height} cm and your weight is {weight}')
print(f'Your  body mass index is { BMI }')
```

Назовем его Bmi.py и запустим из командной строки

```
python Bmi.py
```

Так значительно интересней, правда? Теперь расширим программу, добавив в нее еще условий.

Например: для расчета индекса рост нужно вводить в м, а не в см. И это легко проверить:

```
if int(height)>100:
    print('Что-то ты высоковат! Рост нужно ввести в метрах', end = ' ')
    height = input()
```

Или можно выдать оценку:

```
if BMI<19:
    print('Нужно больше кушать!')
elif (BMI>=19) and (BMI>=25):
    print('Шикарно! Так держать!')
elif BMI > 25:
    print('Надо бы заняться бегом')
else:
    print('Что-то странное')
```

Вариантов много, тут следует поэкспериментировать и посмотреть, что получится.

Задание на дом:
По ссылке www.anaconda.com/products/distribution скачайте дистрибутив Anaconda для своей операционной системы и установите его на свой компьютер.
