第2回　Pythonの基礎

AI概論での勉強を進める上で、必要最低限のPythonの文法を学びます。

情報ネットワーク工学科　204132　鶴田晃矢

Pythonとは

まずは復習をしましょう

変数

Pythonでは、変数に様々な値を入れることができます。 変数に値を入れる場合は、次のように記述します。

変数名 = 値

以下は様々な変数です。
変数名には、数字や_（アンダーバー）を使うこともできます。

プログラム１（変数）

In [2]:

```
a = 123                         # 整数
b_123 = 123.456                 # 小数
hello_world = "Hello World!" # 文字列
print("aの値は",a,"です")  # 整数aを出力しよう
print("b_123の値は",b_123,"です") #b_123を出力しよう
print(hello_world) #hello_worldを出力しよう
```

```
aの値は 123 です
b_123の値は 123.456 です
Hello World!
```

＃の後に書いた文字は、コメントとして扱われます。
コメントはプログラムとして認識されないので、コードの中にメモを書きたい場合はコメントを使います。

演習１：プログラム１を参考にして以下のコメントに合うようにプログラムを書きましょう

In [7]:

```
num = 100  #整数numに１００を代入
PAI = 3.141592  #小数PAIに3.141592を代入しましょう
name = "Koya Tsuruda"  #文字列nameに自分の名前を代入しましょう
print = (num)#numを出力しよう
print = (PAI)  #PAIを出力しよう
print = (name)  #nameを出力しよう
```

演算子

演算子を使って様々な演算を行うことができます。
以下では、様々な演算の結果を変数に格納し、printで表示しています。

プログラム２

In [1]:

```
a = 3
b = 4

c = a + b  # 足し算
print(c)

d = a - b  # 引き算
print(d)

e = a * b  # 掛け算
print(e)

f = a / b  # 割り算
print(f)

g = a ** b  # べき乗（aのb乗）
print(g)
```

```
7
-1
12
0.75
81
```

演習2-1：底辺の長さ（teihen）を５、高さ（高さ）を６とする三角形の面積(s)を求めましょう

In [2]:

```
#練習２-1のプログラムを作り、面積を表示しましょう。（実行までしてください）
```

演習2-2：円の面積sと円周l(エル)の長さを求めましょう。ただし、円の半径rは10cmとする。

In [3]:

```
r=10
PAI=3.141592
l=2*PAI*r
print(l)
s=PAI*r**2
print(s)
```

62.83184
314.1592

+= や -= などの演算子を使うことで、変数自身に値を足したり, 引いたりすることができます。

プログラム３

In [4]:

```
h = 5
h += 3  # h = h + 3と同じ。
print(h)

i = 5
i -= 3  # i = i + 3と同じ
print(i)
```

8
2

演習３-1：+＝を使って、xの値に10を足して、その値を表示しましょう。また、 yの値から10を引いて、その値を表示しましょう。

ただしｘの初期値は５、yの初期値は100とします。

In [1]:

```
x = 5
x += 10
print(x)
y = 100
y -= 10
print (y)
```

15
90

*= や /= などの演算子も同様に、変数自身に値を掛けたり、割ったりすることができます。

演習３-２：＊＝を使って、 j の値を２倍にして、その値を表示しましょう。

jの初期値は５とします。

#練習3-2のプログラムを作り、jの値を表示しましょう。（実行までしてください）

In [6]:

```
j=5
j*=2  #j=j*2
print(j)
```

10

演習3-3：＊＝を使って、 j の値を3倍にして、その値を表示しましょう。

In [2]:

```
j=10
j*=3
print(j)
```

30

演習4：利率0.1%で100万円を5年間預金するといくらになるか計算しましょう。（繰り返し処理は次回以降勉強します。）

In [ ]:

```

```

リスト

リストは、複数の値をまとめて扱う場合に使用します。リストは全体を [] で囲み、各要素は , で区切ります。

In [3]:

```
a = [1, 2, 3, 4]
print(a)
print(a[0])
print(a[1])
print(a[2])
print(a[3])
```

```
[1, 2, 3, 4]
1
2
3
4
```

リスト名の直後に [インデックス] をつけると、リストの要素を取り出すことができます。
インデックスは、要素の先頭から0、1、2、3とつけます。

In [21]:

```
b = [4, 5, 6, 7]
print(b[2]) # 先頭から0、1、2、3とインデックスをつけた場合の2のインデックスの要素を出力
```

```
6
```

演習５： リストｂの最後の要素を出力しましょう

In [6]:

```
print(b[3])
print(b[-1])
```

```
7
7
```

appendでリストに要素を追加することができます。
追加された要素は、リストの一番最後に配置されます。

In [8]:

```
c = [1, 2, 3, 4, 5]
c.append(6)
print(c)
```

```
[1, 2, 3, 4, 5, 6]
```

演習6： リストｂに８と９を追加しましょう

In [23]:

```
b=[4,5,6,7]
b.append(8)
b.append(9)

print(b)
```

```
[4, 5, 6, 7, 8, 9]
```

リストの中にリストを入れて、2重のリストを作ることも可能です。

In [9]:

```
d = [[1, 2, 3], [4, 5, 6]]
print(d)
d.append([7,8,9])
print(d)
```

```
[[1, 2, 3], [4, 5, 6]]
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

dの全ての要素を出力してください。

if文

if文は、条件分岐のために使用します。
以下は、ifの直後の条件が満たされていれば、その直後のブロックの処理を行う例です。

In [10]:

```
a = 5
if a>3:           # aが3よりも大きければ
    print(a + 2)  # インデント（半角スペース4つ）を先頭に挿入する
else:             # a>3を満たしていなければ
    print(a - 2)
```

```
7
```

比較するための演算子には、上記の > （大きい）の他に、 < （小さい）、 >= （以上）、 <= （以下）、 == （等しい）、 != （等しくない）があります。

主な演算子をまとめると以下の通りです。

算術演算子	+	足し算
	-	引き算
	*	かける
	/	割る（小数）
	//	割る（整数）
	%	余り
	**	べき乗
比較演算子	<	小さい
	>	大きい
	<=	以上
	>=	以下
	==	等しい

	!=	等しくない
論理演算子	and	両者を満たす
	or	どちらか片方を満たす
	not	満たさない

In [24]:

```
b = 7
if b==7:          # aが7と等しければ
    print(b + 2)
else:             # a==7を満たしていなければ
    print(b - 2)
```

9

演習７： scoreが６０以上であれば、合格と表示、６０未満であれば不合格と表示しましょう。

In [ ]:

```

```

整数nが2で割り切れたら偶数と割り切れなかったら奇数と表示しましょう。

In [ ]:

```

```

来週の内容を予習しておきましょう。

In [ ]:

```
for a in [4, 7, 10]:  # リストを使ったループ
    print(a + 1)
```

以下は、rangeを使ったループの例です。
rangeは、0から(指定した値-1)までの範囲を指定します。

In [ ]:

```
for a in range(5):  # rangeを使ったループ 0から4までの範囲
    print(a)
```

関数

関数を用いることで、複数行の処理をひとまとめにすることができます。
関数は def と書いて、その後に関数名を記述します。

In [ ]:

```
def my_func_1():  # my_func_1が関数名
    a = 2
    b = 3
    print(a + b)

my_func_1()  # 関数の呼び出し
```

関数は、**引数**と呼ばれる値を関数の外部から受け取ることができます。
引数は、関数名の直後の( )の中に設定します。
引数は、,で区切って設定することができます。

In [ ]:

```
def my_func_2(p, q):  # p、qが引数
    print(p + q)

my_func_2(3, 4)  # 引数として3と4を渡す
```

関数は、**返り値**と呼ばれる値を関数の外部に渡すことができます。
返り値は、関数の最後にreturnと書いて、その直後に設定します。

In [ ]:

```
def my_func_3(p, q):  # p、qが引数
    r = p + q
    return r  # rが返り値

k = my_func_3(3, 4)  # 返り値として受け取った値をkに入れる
print(k)
```

演習:

以下のセルに、引数で与えられた値を2倍して返す関数double を書いてみましょう。受け取った値を出力しよう

In [ ]:

```

```